cadence®

Virtuoso Layout Suite XL: Basic Editing User Guide

Product Version ICADVM20.1
October 2020

© 1990-2020 Cadence Design Systems, Inc. All rights reserved.
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.

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

Preface . . . 45
Scope . . . 46
Licensing Requirements . . . 46
Related Documents . . . 46
What's New and KPNS . . . 46
Installation, Environment, and Infrastructure . . . 46
Technology Information . . . 47
Virtuoso Tools . . . 47
Relative Object Design and Inherited Connections . . . 47
SKILL Documents . . . 47
Additional Learning Resources . . . 48
Video Library . . . 48
Virtuoso Videos Book . . . 48
Rapid Adoption Kits . . . 48
Help and Support Facilities . . . 49
Customer Support . . . 49
Feedback about Documentation . . . 50
Typographic and Syntax Conventions . . . 50

1
Layout Editor Basics . . . 53
The Virtuoso Design Environment . . . 55
Basics of the Virtuoso Design Environment . . . 55
Starting Layout XL . . . 55
Starting Layout XL from a Schematic View . . . 56
Starting Layout XL from a Layout View . . . 69
Starting Layout XL from a Physical Configuration View . . . 70
Automatic Constraint Transfer during Layout XL Startup . . . 71
Specifying the Constraint Group . . . 72
Default Parameter and Property Names . . . 72
Layout XL Desktop . . . 73
Customizing the Desktop . . . 74
Layout XL Window . . . 75
Virtuoso Layout Launch Menu . . . 75
Using the Library Browser . . . 76

© 2020 All Rights Reserved.

Cadence Licensing . . . 76
Working with Designs . . . 77
Opening a Design . . . 77
Making a Design Read Only . . . 82
Making a Design Editable . . . 82
Saving a Design . . . 83
Closing a Design . . . 84
Command Function Basics . . . 85
Starting Commands . . . 85
Canceling Commands . . . 86
Undoing Commands . . . 86
Redoing Commands . . . 86
Repeating Commands . . . 86
Nesting Commands . . . 86
Discarding All Edits . . . 87
Handling Unexpected Results While Using Commands . . . 87
Using Infix . . . 88
Menu Access Keys . . . 88
Using Bindkeys to Run Commands . . . 89
Setting Defaults . . . 93
Startup Files . . . 93
Setting Default Values with Menu Commands . . . 93
Setting the Default Application . . . 94
Setting the Background Color of the Layout Editor Window . . . 94
Selecting the Workspace . . . 94
Using Environment Variables . . . 105
Setting Layout Editor Options . . . 105
Display and Selection Controls . . . 105
Setting Display Colors . . . 105
Editing Entry Layers . . . 106
Selecting Objects in Layout XL . . . 109
Deselecting Objects in Layout XL . . . 110
Viewing Multiple Cellviews in Layout XL . . . 110
Displaying Options Forms Automatically . . . 110
Printing to the Command Interpreter Window . . . 110
Using the Technology File . . . 111

© 2020 All Rights Reserved.

Layout Editor Strokes . . . 111
Loading and Unloading Strokes . . . 111
Drawing Strokes . . . 112
Using Forms . . . 114
Displaying Forms . . . 114
Using Form Buttons . . . 114
Using Context-Sensitive Menus . . . 115
Performing Fast GDS Export Using XStream Out . . . 116
Using the Allegro PCB/Package Menu . . . 117

2
Viewing the Design . . . 119
Zooming In or Out . . . 120
Using the Magnifier . . . 120
Using Dynamic Zoom . . . 122
Panning to a Point . . . 122
Panning Across the Cellview . . . 123
Redrawing the Display . . . 123
Setting an Area Display . . . 123
Viewing Coordinates . . . 124
Viewing Angles . . . 129
Viewing Selected Object Information . . . 134
Saving or Restoring a View . . . 135
Saving a Zoom or Pan Image . . . 135
Restoring a Zoom or Pan Image . . . 135
Displaying a Background Cellview . . . 136
Reading Status Information . . . 137
Creating a Reference Window . . . 140
Using a Reference Window to Zoom or Pan . . . 141
Using a Reference Window to Create or Edit Objects . . . 141
Using the World View Window . . . 141
Setting Display Options . . . 142
Setting Up the Design Objects to Appear . . . 142
Displaying Instance or Master Cell Names . . . 143
Displaying Details or Outlines of Arrays . . . 143

© 2020 All Rights Reserved.

Setting Display Levels 144
Setting Up Grid Controls 145
Setting Up Snap Grid 145
Setting Up Snap Grid Spacing 146
Setting Up Snap Mode 146
Setting Up Dimming Options 148
Saving, Loading, and Deleting Display Settings 155
Displaying Object Information 156
Controlling Information Balloons and Dynamic Measurements 157
Customizing Object Display 160
Using a Custom SKILL Function 164
Saving the Display Information Configuration 164
Loading the Display Information Configuration 165
Saving a Configuration as Default 166
Deleting a Display Information Configuration 166
Displaying Preselect and Selected Object Information 166
Cycling Through Preselected Objects 175
Cycle Selection in Partial Selection Mode 175

3
Palette Assistant 177
Features of Palette Panels 177
Window Context and Layer Sets 178
Window Context 178
Layer Sets 178
About Synchronized Palettes 179
Displaying and Hiding the Single Palette Assistant 179
Palette Title Bar Buttons 179
Palette Containers 180
Single Assistant Container 180
Multiple Assistants Container 181
Single Window Container 182
Setting the Palette Type 183
Layers Panel 184
Displaying and Hiding the Layers Panel 184

© 2020 All Rights Reserved.

User Interface . . . 185
Using Mouse Functions to Control Layer-Purpose Pairs . . . 213
Tasks in the Layers Panel . . . 215
MPT Support . . . 237
Objects Panel . . . 253
Displaying and Hiding Objects Panel . . . 254
User Interface . . . 255
Grids Panel . . . 259
Displaying and Hiding Grids Panel . . . 259
User Interface . . . 259
Determining Visibility and Selectability in Objects and Grids Panels . . . 263
Switching between Modes . . . 265

4
Using Design Tools . . . 267

Using Find/Replace . . . 268
Specifying the Search Criteria . . . 268
Finding Objects . . . 269
Finding Objects in Instances . . . 272
Finding Instances . . . 272
Replacing Instances . . . 273
Finding and Replacing Text of Labels . . . 275
Using Area and Density Calculator . . . 278
Using Rulers . . . 280
Creating Rulers . . . 280
Saving Rulers . . . 292
Deleting Rulers . . . 292
Re-mastering Instance Views . . . 293
Replacing Abstract Views with Layout Views . . . 293
Re-Mastering Examples . . . 295
Checking Terminals . . . 296
Creating Pins from Labels . . . 298
Using the Tap Command . . . 300
Using the Tap Command on the Tools Menu . . . 300
Using the Tap Section in the Layout Editor Options Form . . . 300

© 2020 All Rights Reserved.

Changing Objects on a Layer . . . 303
Making Changes to Objects on Layers . . . 303
How Sizing Converts Objects to Polygons . . . 304
Making Boolean Edits by Layer . . . 304
Generating Abstracts . . . 306
Working with Markers . . . 307
Explaining Markers . . . 307
Finding Markers . . . 307
Deleting Individual Markers . . . 308
Deleting All Markers . . . 309
Pad Opening Info Command . . . 312
Generating a Standard Report . . . 312
Generating a Customized Report . . . 316
Generating Labels for Pads . . . 320
Promoting Labels to the Top Level . . . 321

5
Working With Hierarchical Designs . . . 323

Hierarchical Designs . . . 324
Master Cells and Cell Instances . . . 324
Checking for Instance Recursion . . . 324
Changing the Origin of a Cellview . . . 325
Hierarchical Editing . . . 326
Descending into a Cellview . . . 326
Returning to a Different Hierarchy Level . . . 327
Editing a Cell in Place . . . 328
Editing in Place into an Instance Array . . . 328
Using the Tree Form . . . 333
Using the Print Hierarchy Tree Command . . . 334
Making Hierarchy Editable . . . 335
Making Cells . . . 335
Flattening Instances . . . 337
Copying and Cutting Through the Hierarchy . . . 339
Embedded Module Hierarchy Designs . . . 340
Uniquifying the Cellview . . . 340

© 2020 All Rights Reserved.

Listing the Data In Cellviews . . . 341
Refreshing Memory after Editing . . . 342

6
Creating Objects . . . 345
Entering and Deleting Points . . . 347
Entering Points in the CIW . . . 347
Entering Points with the Keypad . . . 348
Using the Enter Points Form . . . 348
How Creation Commands Work with ROD Objects . . . 349
Creating Rectangles . . . 352
Creating Polygons . . . 354
Creating an Arc . . . 355
Creating Paths . . . 356
Controlling the Display of Paths . . . 362
Creating Circles . . . 363
Creating Ellipses . . . 364
Creating Donuts . . . 365
Creating Geometric Wires . . . 365
Creating Instances . . . 366
Creating Pins . . . 367
Creating Strongly Connected Pins . . . 367
Creating Weakly Connected Pins . . . 369
Placing Pin Arrays . . . 371
Setting Pin Label Text Style Options . . . 372
Support for Orthogonal WSP Grid . . . 374
Creating Labels . . . 375
Configuring Label Creation Options . . . 375
Label Creation Methods . . . 379
Label Creation Modes . . . 385
Labelling Overlapping Shapes . . . 393
Creating Vias . . . 398
Types of Vias . . . 398
Parameter Caching . . . 399
Setting Up Via Creation Options . . . 408

© 2020 All Rights Reserved.

Creating Vias Manually . . . 411
Creating a Via Automatically in Single and Stack Modes . . . 424
Setting the Technology to Create Via . . . 440
Creating Multipart Paths . . . 453
Creating Fluid Guard Rings . . . 453
Creating MPP Guard Rings . . . 453
Creating Slots . . . 453
Slotting Flow for Wide Wires . . . 456
Creating P&R Objects . . . 457
Creating Blockages . . . 457
Creating Rows . . . 460
Creating Custom Placement Areas . . . 466
Creating Clusters . . . 467
Creating Track Patterns . . . 469
Creating P&R Boundary . . . 476
Creating Snap Boundary . . . 477
Creating Area Boundary . . . 478
Creating Cluster Boundary . . . 479
Creating Groups . . . 480

7
Editing Objects . . . 483

Undoing and Redoing Commands . . . 484
Moving Objects . . . 484
Moving Objects by Direct Manipulation . . . 485
Moving Objects By Using the Move Command . . . 486
Moving or Copying Objects to Another Layer . . . 487
Moving Objects to Another Cellview . . . 487
Moving Multipart Rectangles . . . 488
Copying Objects . . . 488
Creating an Array of Copies . . . 493
Copying Objects to Another Cellview . . . 500
Copying Cellviews Across Libraries . . . 501
Stretching Objects . . . 501
Stretching Objects by Direct Manipulation . . . 503

© 2020 All Rights Reserved.

Stretching a Circle . . . 504
Stretching a Via . . . 505
Changing the Angle of Corners . . . 505
Stretching and Moving Objects . . . 506
Stretching Paths . . . 507
Stretching Multipart Rectangles . . . 509
Deleting Objects . . . 511
Deleting Objects, Edges and Corners . . . 511
Using the Delete Form . . . 512
Canceling Delete . . . 512
Copying Objects in Repeat Mode . . . 512
Flipping Objects . . . 516
Rotating Objects . . . 520
Rotating a Geometric Object by Any Angle . . . 522
Rotating Multiple Selected Objects . . . 525
Rotating or Mirroring an Object . . . 528
Rotating Instances and Labels . . . 530
Chopping Objects . . . 531
Chopping Vias . . . 535
Merging Objects on a Layer . . . 537
How Merging Converts Objects . . . 538
How Merging Converts Paths . . . 539
Yanking and Pasting Objects . . . 543
Yanking and Pasting Objects in Pre-Selection Mode . . . 543
Yanking and Pasting Objects in Post-Selection Mode . . . 545
Editing Instance Properties . . . 546
Reshaping Objects . . . 546
Reshaping Polygons . . . 547
Adding a Rectangle to a Polygon . . . 547
Reshaping a Path . . . 548
Splitting Objects . . . 549
Splitting an Object . . . 549
Using Split to Change the Angle of Objects . . . 553
Splitting Multiple Objects . . . 554
Modifying Corners . . . 557
Modifying the Corners of a Polygon . . . 557

© 2020 All Rights Reserved.

Sizing Objects . . . 558
Attaching and Detaching Objects . . . 559
Attaching Objects . . . 559
Detaching Objects . . . 561
Converting Instances to Mosaics . . . 563
Converting Mosaic to Instances . . . 566
Converting Circles, Donuts, and Ellipses to Polygons . . . 566
Setting Conic Sides . . . 568
Converting Shapes Pathsegs . . . 568
Converting Polygons to Paths . . . 568
Changing Origin . . . 570
Aligning Objects . . . 570
By Using the Quick Align Command . . . 571
By Using the Align Command . . . 604
Editing Slots . . . 625
Hierarchical Editing . . . 626
Editing Groups . . . 626
Adding Objects to a Group . . . 626
Adding Objects to a Group By Using the Select Target Group Form . . . 627
Ungrouping All Objects in a Group . . . 628
Removing Objects from a Group . . . 629
Editing Group Contents . . . 629
Selecting Objects . . . 631
Selection Basics . . . 632
Pre-Selection and Post-Selection of Objects . . . 633
Selecting Objects . . . 633
Making Objects Selectable and Unselectable . . . 635
Selecting a Group of Objects . . . 638
Selecting Vias in a Via Stack . . . 641
Selecting Multipart Rectangles . . . 643
Selecting Path Components . . . 644
Selecting Path Segments . . . 646
Deselecting Objects . . . 649

© 2020 All Rights Reserved.

8
Creating and Editing Multipart Paths 651
Creating Multipart Paths 652
About Creating a Multipart Path in a Layout Window 653
About Saving a Multipart Path as a Template 654
Flow for Creating Multipart Paths 655
Defining the Master Path 656
Defining Connectivity for the Master Path 660
Adding a New Subpart 662
Defining Connectivity for a Subpart 672
Editing a Subpart 673
Deleting a Subpart 674
Creating a Multipart Path in a Cellview 676
Saving Multipart Path Values as a Template 677
Loading an ASCII Multipart Path Template File 680
Creating an MPP Guard Ring 681
Setting Multipart Path Environment Variables 682
Setting the mppTemplate Environment Variable 682
Setting the mppASCIIFileName Environment Variable 683
Selecting Multipart Paths 685
Selecting a Whole Multipart Path 686
Selecting Multipart Path Ends 687
Selecting Chopped Subpath Ends in Multipart Paths 688
Selecting Multipart Path Segments 690
Selecting a Multipart Path Vertex 692
Moving Multipart Paths 694
How Moving Affects Multipart Paths with Aligned Objects 694
Stretching Multipart Paths 696
Stretching Chopped Subpath Ends in Multipart Paths 698
Splitting Multipart Paths 701
Reshaping a Multipart Path 704
How Chopping Affects Multipart Paths 706
Chopping a Multipart Path 708

© 2020 All Rights Reserved.

9
Creating and Editing Fluid Guard Rings 711

10
Editing and Defining Properties 713
Understanding Properties 714
About Editing Properties 714
Displaying Properties 714
Displaying Context-Sensitive Menu 717
Editing the Properties of an Object 717
Editing Properties of a Group of Objects 719
Searching For and Replacing Properties 719
Defining a New Cellview Property 720
Defining a New Object Property 721
Modifying a Property 721
Deleting a Property 722
Editing Object Attributes 723
Editing Object Connectivity 723
Editing Cut Patterns 723
Editing Multipart Paths 725
Using Net Expressions and Inherited Connections 729
Inherited Connections 729
Net Expressions in the Layout Editor Environment 730
Creating a Net Expression 730
Editing Net Expressions 731
Viewing Instances Containing Net Expressions 731

11
Using Connectivity 733
Overview of Pin Connectivity 734
Defining Must Connect Pins 740
Defining Strongly Connected Pins 742
Defining Weakly Connected Pins 743
Defining Pseudo Parallel Connected Net 745

© 2020 All Rights Reserved.

Checking Connectivity of Pins . . . 746
Propagating Nets . . . 747
Promoting Net Information . . . 747
Adding and Deleting Shapes on Nets . . . 748
Adding Shapes to a Net . . . 749
Removing Shapes from a Net . . . 749
Assigning and Unassigning Nets to Shapes . . . 750
Assigning Nets to Shapes in Pre-Selection Mode . . . 750
Assigning Nets to Shapes in Post-Selection Mode . . . 750
Unassigning Nets from Shapes in Post-Selection Mode . . . 751
Selecting Shapes Based on Nets . . . 751
Tracing Nets Using the Mark Net Command . . . 753
Specifying Via, Connected, and Stop Layers for Tracing a Net . . . 756
Tracing Nets . . . 758
Saving Marked Nets . . . 767
Unmarking Nets . . . 768
Probing Nets, Instances, Pins, and Shapes . . . 768
Pre-Selection Probing . . . 769
Post-Selection Probing . . . 773

12
Microwave Commands . . . 777

About Microwave Layouts . . . 778
Editing Transmission Lines . . . 778
Creating Transmission Lines . . . 778
Creating Bends . . . 779
Creating Advanced Transmission Lines . . . 780
Creating Tapers . . . 781
Transmission Line Bend Examples . . . 782
Standard Bend Examples . . . 782
Chamfer Examples . . . 785
Radial Examples . . . 787

© 2020 All Rights Reserved.

13
Handling Unexpected Results . . . 789
Design Access Problems . . . 790
Cannot Find a Library . . . 790
Cannot Open a Cellview . . . 790
Cannot Write to a Cellview . . . 790
Cell Instances Are Missing . . . 791
Loading an ASCII Template File Returns a CIW Warning Message . . . 792
Problems with Editing and Entering Points . . . 793
Mistakes Entering Points . . . 793
Entered Points on the Wrong Layer . . . 793
Cannot Move Objects or Enter Points As Required . . . 794
Object Moves Instead of Stretching . . . 794
Object Changes Shape When Resizing . . . 795
Cannot Paste an Object . . . 796
Pcell Parameters Do Not Appear . . . 796
Cannot Edit the Properties of Selected Instances . . . 797
Mouse and Cursor Behavior . . . 797
Right Mouse Button Does not Work . . . 797
Pointer Changes Shape . . . 798
Cursor Sticks to Objects . . . 798
Cursor Snaps to the Wrong Object . . . 798
Cursor Does not Snap to a Point . . . 799
Selection Tips . . . 799
Selectable Layers . . . 799
How to Make a Layer Selectable . . . 799
Selection During Edit In Place . . . 800
How to Select Objects in a Dense Design . . . 800
Instance Selectability . . . 801
Problems Selecting Copied Objects . . . 801
Mouse Buttons for Zooming and Selecting . . . 801
How to Select Invisible Objects . . . 802
Window Display Tips . . . 802
How to Make a Design Layer Visible . . . 802
How to See Instance Contents . . . 802

© 2020 All Rights Reserved.

Displaying Instances You Are Editing 803
How to See What the Search Command Finds 803
Problems with Stretching Parameterized Cells 803
Pcell Stretch Handles Are Not Visible 803
Difficulty Selecting Pcell Stretch Handles 804

14
Dialog Box Messages 805
About Messages 806
Closing Dialog Boxes 806
Using Question Dialog Boxes 806
List of Error Messages 807

15
FinFET Support in Layout Editor 829
Snapping Objects to Local Snap Pattern Shapes 831
Snapping Objects to Snap Patterns 831
Placing Objects Inside the Fin Boundary 831
Controlling the Display of Snap Patterns 833
Turning On Snap Pattern Snapping 835
Snapping Objects 835
Snapping Objects and Shapes as Groups 841
Track Group Support 842
Snapping Objects to Global Snap Pattern Grid 846
Controlling the Display of Global Grid 846
Anchoring the Global Grid 846
Commands Supporting Snap Pattern Snapping 849
Controlling the Display of Snap Patterns in Palette 851
Modes for Object Placement 853
Selecting Placement Mode 853
Abstract Mode Placement 853
Manual Mode Placement Override 855
Create Custom Placement Area 857
Snapping for singleTrackCenter Snap Mode 857
Snapping for Edit Commands 860

© 2020 All Rights Reserved.

Support for the exceptOverlap Attribute 860

16
Width Spacing Pattern Support in Layout Editing 863
Overview 864
Controlling the Display of Width Spacing Patterns 865
Snapping Behavior of Objects in Width Spacing Patterns 871
Pin Snapping 871
Pin Snapping for Overlapping Regions 871
Region Shape Snapping 874
Region Group Snapping 875
Instance Snapping: Abstract Mode Placement 875
Enhancements in Editing Commands 877
Enhancements in Create Commands 879
Creating Pins 879
Display Enhancements for WSP 881
Support for Alpha Blending 881
Alpha Blending Environment Variables 882
Support for Filtering Tracks 882
Interactive Snapping 883
Interactive Snap Pattern Snapping: Create Rectangle 883
Interactive Snap Pattern Snapping: Move and Copy Commands 884
Interactive Snap Pattern Snapping: Stretch Command 885
Interactively Setting Color on a Selected Object 887
Enhancements in the Property Editor 887

A
Layout Editor Forms 891
Add Shape to Net Form 897
Align Form 898
Area and Density Calculator Form 905
Assign Net Form 907
Background Cellview Form 908
Choose Object for Label Creation Form 909
Chop Form 910

© 2020 All Rights Reserved.

Clean Overlapping Contacts Form 914
Copy Form 915
Create Area Boundary Form 918
Create Blockage Form 919
Create Circle Form 922
Create Cluster Boundary Form 923
Create Custom Placement Area Form 925
Create and Edit Clusters Form 927
Create Encore Report Form 929
Create Geometric Wire Form 931
Create Group Form 932
Create Guard Ring Form 933
Fluid Guard Ring Context-Sensitive Menu 934
Create Instance Form 935
Create Label Form 938
From Interconnect 945
From Instance 945
Create Measurement Form 947
Ruler Context-Sensitive Menu 954
Create Multipart Path Form 957
Connectivity Section 959
ROD Subpart Form 962
Create Path Form 971
Path Options 971
Slot Options 972
Create Pin Form 974
Manual Mode Options 974
Auto Mode Options 976
Create Pins From Labels Form 979
Create Polygon Form 981
Create P&R Boundary Form 982
Create Rectangle Form 983
Create Row Form 985
Definition 985
Templates 986
Create Method 986

© 2020 All Rights Reserved.

Create Slot Form 988
Calculating Dimensions Associated with Slots 992
Create Snap Boundary Form 995
Create Via Form 996
Single Mode Options 997
Stack Mode Options 1007
Auto Mode Options 1008
Create Via Options Form 1018
Custom Show Purpose Form 1021
Define Net/Pin Connections Form 1022
Delete Form 1023
Delete All Markers Form 1024
Delete Configuration Form 1026
Delete Marker Form 1027
Descend Form 1028
Descend Options Form 1029
Discard Edits Form 1030
Display Options Form 1031
Dynamic Display Form 1041
Setup Tab 1041
Info Balloon Tab 1042
Dynamic Measurement Tab 1042
Highlight Aligned Edges Tab 1043
Parameter Types for Shapes and Objects 1044
Dynamic Selection Assistant Options Form 1047
Dynamic Selection Context-Sensitive Menu 1048
Edit Properties Form 1050
Add Property Form 1053
Modify Property Form 1054
ROD Property Form 1055
Edit Cellview Properties Form 1057
Edit Cut Pattern Form 1059
Edit Via Array Pattern Form 1063
Edit Instance Properties Form for a Fluid Guard Ring 1064
Attribute Types for Objects 1065
Edit Slot Form 1092

© 2020 All Rights Reserved.

Edit Synchronous Clone Form 1093
Enter Points Form 1094
Export Label Form 1097
Export Image Form 1098
Filter Layers Form 1099
Find Marker Form 1101
Find/Replace Form 1103
Flatten Form 1109
Flip Form 1111
Group Return 1112
Freeze Pcell Instance Form 1113
Guard Ring Form 1114
Guard Ring Chop Options Form 1115
Guard Ring Create Tunnel Options Form 1116
Heal Options Form 1117
Halo Editor Form 1118
Highlight Options Form 1120
Install Guard Ring Form 1122
Instance to Mosaic Form 1123
Layer Generation Form 1124
Layout Editor Options Form 1125
- Editor Controls 1125
- Foundry Constraints 1127
- Gravity Controls 1128
- Tap 1128
- Snapping 1129
- Wire Editing 1130

Load Multipart Path Template Form 1137
Magnifier Options Form 1138
Make Cell Form 1141
Make Readonly/Editable Form 1143
MarkNet Options Form 1144
- Options Tab 1144
- Via Layers Tab 1147
- Connected Layers Tab 1148
- Stop Layers Tab 1148

© 2020 All Rights Reserved.

Marker Text Window 1150
Modify Corner Form 1151
Move Form 1152
Pad Opening Info Form 1155
Search Criteria 1155
Sort 1157
Reporting 1157
Label Control 1157
Promote Label 1158
Palette Options Form 1160
Paste Form 1162
Print Hierarchy Tree Form 1163
Propagate Nets Form 1165
Quick Align Form 1166
Quick Align Context-Sensitive Menu 1168
Refresh Cellviews Form 1169
Remaster Instances Form 1170
Remove Shape From Net Form 1171
Repeat Copy Form 1172
Reshape Form 1174
Restore View Form 1175
Return To Level Form 1176
Rotate Form 1177
Save a Copy Form 1178
Save All Mark Nets Form 1179
Save/Restore Selection Set 1180
Save Configuration Form 1181
Save Via Variant Form 1182
Save View Form 1183
Save Multipart Path Templates Form 1184
Select Area Shape Type Form 1185
Select Array Member Form 1186
Selection Options Form 1187
Selection Protection Highlight Options Form 1192
Set Area View Level Form 1194
Set Default Application Form 1195

© 2020 All Rights Reserved.

Set Pin Label Text Style Form 1196
Show Selected Set Form 1197
Size Form 1198
Split Form 1199
Stretch Form 1200
Summary Window 1204
Tap Form 1207
Track Pattern Editor Form 1209
Transmission Line Bend Form 1211
Transmission Line Form 1212
Transmission Line Taper Form 1213
Transparency Options Form 1214
Tree Form 1216
Uniquify Form 1217
Unmark Net Form 1218
Yank Form 1219

B
Environment Variables 1221

Environment Variable Functions 1221
Checking the Value of an Environment Variable 1221
How the System Evaluates Environment Variables 1222
Graphic Environment Variables 1222
Layout Editor Environment Variables 1223
Copying Environment Variables from the Samples Directory to ~/.cdsenv 1224
Changing Settings for Environment Variables in the CIW 1224
Changing Graphic Environment Variables That Are Cellview Properties 1225
Environment Variables in Forms 1225
Improving Performance Using Environment Variables 1225
List of Layout Editor Environment Variables 1230
Graphic Environment Variables 1231
Layout Editor Environment Variables 1240
abutServerOn 1264
accessEdgesOn 1265
acuteAnglePath 1266

© 2020 All Rights Reserved.

addDesignLppsToTech 1267
alignApplySpacing 1268
alignDirection 1269
alignLayer 1270
alignMatchRefLayer 1271
alignMatchRefSize 1272
alignMethod 1273
alignObjType 1276
alignReference 1277
alignRefMethod 1278
alignRefObjType 1280
alignReverseSort 1281
alignSortMethod 1282
alignSpacingType 1283
alignSpacingValue 1284
alignToolBarReferenceObject 1285
alignUpdateLabelLayer 1287
alignUsing 1288
alignUsingInstanceDrawingMode 1289
allowPartialDelete 1290
alphaBlendingBoundaryFillValue 1291
alphaBlendingBoundaryHaloValue 1292
alphaBlendingBoundaryOutLineValue 1293
alphaBlendingFinFillValue 1294
alphaBlendingFinHaloValue 1295
alphaBlendingFinOutLineValue 1296
alphaBlendingTrackFillValue 1297
alphaBlendingTrackHaloValue 1298
alphaBlendingTrackOutLineValue 1299
anchor 1300
applyBounce 1301
arrayDisplay 1302
askSaveOnReturn 1303
autoCopyTemplateCellViewList 1304
autoCircleCreation 1305
autoExtractSaveEnabled 1306

© 2020 All Rights Reserved.

autoLoadBindKeys 1307
autoModePinSize 1308
autoModePinSizeFromPath 1309
autoNetName 1310
autoPermutePins 1311
autoPinLayerFunctionsList 1312
autoPinRectHeight 1313
autoPinRectWidth 1314
autoRefresh 1315
autoSaveMarkNetSetupFile 1316
autoSetRefPoint 1317
autoStoreSelectionSetOn 1318
autoTapShape 1319
autoTapType 1320
autoViaOverrideVariantParameters 1321
autoZoomMode 1322
autoZoomPan 1323
autoZoomScale 1324
balloonFadeTime 1325
balloonOn 1326
balloonPauseTime 1327
balloonTransparency 1328
bannerFields 1329
blockageAllowPGNet 1331
blockageAttachOwner 1332
blockageAttributeMode 1333
blockageCreateMethod 1334
blockageDerivePoints 1335
blockageIsPushedDown 1336
blockageKeepOwner 1337
blockageMaterial 1338
blockageMaxDensity 1339
blockageSpacingType 1340
blockageSpacingValue 1341
blockageType 1342
bondPadLPP 1343

© 2020 All Rights Reserved.

boundarySnappingLayer 1344
boundarySnapXGrids 1345
boundarySnapYGrids 1346
chainMode 1347
checkAllSPGridsEnclosures 1348
checkHalfWidthPathSeg 1349
checkLargeCoordinates 1350
chopArray 1351
chopCircleJustify 1352
chopCircleRadius 1353
chopLineJustify 1354
chopLineType 1355
chopLineUserWidth 1356
chopPRBDialog 1357
chopRectHeight 1358
chopRectJustify 1359
chopRectWidth 1360
chopShape 1361
chopSpecifySize 1362
chopSpecifySpacing 1363
circleRadius 1364
circleShapeMode 1365
clusterBoundaryAspectRatio 1366
clusterBoundaryUtilization 1367
confirmDelete 1368
contactColumns 1369
convertEachInstanceIntoMosaic 1370
copyColumns 1371
copyConnectivity 1372
copyIncrementInstName 1373
copyIncrementNetName 1374
copyResetRowCol 1375
copyRows 1376
copyToLayer 1377
createAsLabel 1378
createInstResetCDF 1379

© 2020 All Rights Reserved.

createPinLabel 1380
createPinPathSpineAware 1381
createPinSchematicAware 1382
createRectilinearBoundary 1383
createViaAlwaysUsesDefinedCalcMode 1384
cursorShape 1385
defaultNewCellName 1386
defaultNewLibName 1387
defaultNewViewName 1388
defaultOpenCellName 1389
defaultOpenLibName 1390
defaultOpenViewName 1391
defaultToolName 1392
defCreatePinConnModel 1393
delDetachedBlockages 1394
deleteFluidShapes 1395
deleteTarget 1396
descendPromptForView 1397
dimGridLines 1398
dimmingAutoEnabled 1399
dimmingIntensity 1400
dimmingOn 1401
dimmingScope 1402
disableLayerFuncSnapToMfgGrid 1403
disableMissingLayerPopUp 1404
disableRoundPathStyle 1405
discardCVEditOnWindowClose 1406
displayFigGroupName 1407
displayFigGroupShapes 1408
displayInstanceOrientation 1409
displayMissingMasterMarkers 1410
displayOptionsFile 1411
displayPCellOrientation 1412
displayPinNames 1413
displayPRHighestResolution 1414
displayRefPoint 1415

© 2020 All Rights Reserved.

displayResolution 1416
displayRowName 1417
displayRowSite 1418
displayRulerMiddlePoint 1419
displaySchDashText 1420
displaySelectionHighlightWhenDimmed 1421
displaySigDigits 1422
displayStartLevel 1423
displayStopLevel 1424
displayStretchHandles 1425
displayViaShapes 1426
doNotShowChopPRBDialog 1427
dotsOn 1428
dotStyleMajorGrid 1429
drawAxesOn 1430
drawDottedGridOn 1431
drawEngine 1432
drawGridOn 1433
drawInstancePinNames 1434
drawInstancePins 1435
drawLabelsSurround 1436
drawPRBoundary 1437
drawSmallLabelOrigin 1438
drawSurroundingOn 1439
drawTraversedInstBbox 1440
drawTraversedInstName 1441
drdEditSmartSnapAllowedWidthSnap 1442
drfPath 1443
dynamicHilightOn 1444
dynamicMeasurementBackgroundColor 1445
dynamicMeasurementDrawBackground 1446
dynamicMeasurementDisplay 1447
dynamicMeasurementEdge 1448
dynamicMeasurementFontSize 1449
dynamicMeasurementFromCursor 1450
dynamicMeasurementLayerScope 1451

© 2020 All Rights Reserved.

dynamicMeasurementTextColor 1452
dynamicProbe 1453
editInstanceBoundaryMode 1454
editorOptionsFile 1455
eipHighlightColor 1456
enableAlignToolBarReferenceObject 1457
enableCopyAndPaste 1458
enableEIPEnhancements 1459
enableSyncCloneEditing 1460
expandPinBusExpression 1461
extendPathLengthCheck 1462
extractAfterGenerateAll 1463
extractCellviewIllegalConnectionLimit 1464
extractCellviewOpenViolationLimit 1465
extractCellviewShortViolationLimit 1466
extractCellviewUnnecessaryPseudoParallelConnectionLimit 1467
extractConnectUnassignedInstTerms 1468
extractConsiderAllCutLayersInVias 1469
extractEnabled 1470
extractEnableVirtualConnections 1472
extractHierExtractSearchDepth 1473
extractKeepShapeAssignment 1474
extractNetOpenViolationLimit 1475
extractNetShortViolationLimit 1476
extractReinitOnStopLevelChange 1477
extractRemoveStickyProperty 1478
extractRunOnInit 1479
extractStopLevel 1480
extractVerifyAnalogSigType 1481
extractVerifyBigMPP 1482
extractVerifyClockSigType 1483
extractVerifyGroundSigType 1484
extractVerifyIllegalHierConnections 1485
extractVerifyIllegalOverlapsForStoppedColoredShapes 1486
extractVerifyIncompleteMustConnectAllPins 1487
extractVerifyInvalidConnections 1488

© 2020 All Rights Reserved.

extractVerifyMustConnectMarkers 1489
extractVerifyOpenViolations 1490
extractVerifyOpticalPortRadiusMatch 1491
extractVerifyPowerSigType 1492
extractVerifyResetSigType 1493
extractVerifyScanSigType 1494
extractVerifySignalSigType 1495
extractVerifyTieHighSigType 1496
extractVerifyTieLowSigType 1497
extractVerifyTieOffSigType 1498
extractVerifyWellSubstrateConnections 1499
extractVerifyUnimplementedInstTerms 1500
extractVerifyUnnecessaryPseudoParallelConnections 1501
extractVerifyWeakMarkers 1502
extractVirtualConnectionSeparator 1503
filterSizeForResolution 1504
fitIntoClusterBoundary 1506
fixedWidthPaths 1507
flattenExcludePcellListFile 1508
flattenMode 1509
flattenPCells 1510
flattenVias 1511
flipAroundCombinedCenter 1512
followRoutingDirection 1513
fullSelectionMode 1514
ghostOfDraggedObjects 1515
gravityApertureInPixels 1516
gravityDistance 1517
gravityEditCellViewOnly 1518
gravityOn 1519
gravitySelectableOnly 1520
gravityType 1521
gravityUse5PixelApertureWhenOff 1522
grEnclosedBy 1523
gridMultiple 1524
gridSpacing 1525

© 2020 All Rights Reserved.

highlightSelProtectedItems 1526
haloBottomOffset 1527
haloLeftOffset 1528
haloRightOffset 1529
haloTopOffset 1530
highlightAlignedEdges 1531
highlightAlignedEdgesColor 1532
highlightAlignedEdgesWidth 1533
iconsOn 1534
ignoreBounce 1535
ignoreParallelMinWidthForSquarePinResize 1536
instanceDrawingMode 1537
instanceSelectionMode 1538
instanceVisibilityMode 1539
instCellName 1540
instColumns 1541
instDelX 1542
instDelY 1543
instHalosEnabled 1544
instLabel 1545
instLibName 1546
instMosaicCreation 1547
instName 1548
instPhysicalOnly 1549
instRows 1550
instViewName 1551
keepCopiedInsideGroup 1552
keepFirstTermName 1553
keepGuardRingEndsConnected 1554
keepWiresConnectedToNonWires 1555
labelAttach 1556
labelAutoConn 1557
labelAutoInst 1558
labelDrafting 1559
labelExpand 1560
labelFontStyle 1561

© 2020 All Rights Reserved.

labelHeight 1562
labelJustify 1563
labelKeepLabel 1564
labelLPP 1565
labelMode 1566
labelOverbar 1567
labelPurpose 1568
labelText 1569
leCheckMissingLayers 1570
leCheckMissingMasters 1571
leChopPreservePathSeg 1572
leDispNetExpr 1573
leWindowBBox 1574
lockAngles 1575
lppVisibilityMode 1576
magAnchorPosition 1577
magConfined 1578
magCursorOffset 1579
magDisplayMode 1580
magFloatPosition 1581
magOffsetCorner 1582
magScale 1583
magSize 1584
maintainConnections 1585
makeCellCellType 1586
markerAccessMode 1587
markerDeleteAccessModeIsEdit 1588
markerDeleteConfirmModeChange 1589
markerDeleteScope 1590
markerDeleteSeverities 1591
markerDeleteSources 1592
markerDeleteSourcesMode 1593
markerSearchBy 1594
markerSearchScope 1595
markerSeverity 1596
markerZoom 1597

© 2020 All Rights Reserved.

markerZoomScale 1598
markNetAutoColorCycle 1599
markNetConnectedLayer 1600
markNetExcludePurposes 1601
markNetHilightObj 1602
markNetNumThreads 1603
markNetOptionsFile 1604
markNetRange 1605
markNetSaveViaInfo 1606
markNetSelectOverlapLayer 1607
markNetStartLevel 1608
markNetStopLayer 1609
markNetStopLevel 1610
markNetThickline 1611
markNetViaLayer 1612
maskLayoutDefaultApp 1613
maxAutoStoreSelectionObj 1614
maxAutoStoreSelectionSet 1615
maxPathSegInSpineAreaSelection 1616
measurementDisplayOn 1617
measurementTextHeight 1618
mergeEnable 1619
mergeScope 1620
mergeVias 1621
mergeXStippleShapes 1622
meshNBThread 1623
modalCommands 1624
modCornerRadius 1625
modCornerType 1626
mppASCIIFileName 1627
mppTemplate 1628
mustConnectAllPins 1629
multiThreadInfo 1630
muteUnstretchableOption 1631
netBasedSelectionNetName 1632
netBasedSelectionOn 1633

© 2020 All Rights Reserved.

netBasedSelectionReverse 1634
netExprDisplay 1635
netName 1636
netNameDisplayDrawBackground 1637
netNameDisplayDrawOnTop 1638
netNameDisplayEnabled 1639
netNameDisplayFilter 1640
netNameDisplayFilterDefaultList 1641
netNameDisplayFilterLayers 1642
netNameDisplayFilterMode 1643
netNameDisplayFontStyle 1644
netNameDisplayMaxLabelSize 1646
netNameDisplayMaxLabelSizeFull 1647
netNameDisplayNetShow 1648
netNameDisplayScope 1649
netNameDisplaySkipInstances 1650
netNameDisplaySkipPcells 1651
netNameDisplaySkipLayers 1652
netNameDisplaySkipVias 1653
netNameDisplayTermShow 1654
netNameDisplayTermSkipInstances 1655
netNameDisplayTermSkipPcells 1656
netNameDisplayUserColor 1657
netNameDisplayUseShapeColor 1658
netsOn 1659
nPtsToTaper 1660
numLevels 1661
numSides 1662
openSavedCellViewInWindow 1663
openToStopLevel 1664
orientation 1665
originMarkersOn 1666
overrideFontName 1667
packetDialogBoxes 1668
padLabelLPP 1669
partialSelect 1670

© 2020 All Rights Reserved.

partialSelectProximityMaxPixel 1671
partialSelectionMode 1672
pasteRef 1673
pathBeginExt 1674
pathCL 1675
pathEndExt 1676
pathJustify 1677
pathOffset 1678
pathStyle 1679
pathWidth 1680
photonicDisplay 1681
photonicPinWidth 1682
photonicPinAngle 1683
photonicPinFacetInPacket 1684
photonicPinFacetOutPacket 1685
photonicPinLabelPacket 1686
photonicPinRadius 1687
photonicPinRadiusLinePacket 1688
photonicPinWidthLinePacket 1689
pinAccessDir 1690
pinDefaultObjectType 1691
pinLabelPlacementMode 1692
pinLayerPurpose 1693
pinLPP 1694
pinMode 1695
pinSelectionMode 1696
pinShape 1697
pinTextCreateAsLabel 1698
pinTextLayer 1699
pinTextLayerNames 1700
pinTextPurpose 1701
pinTextPurposeNames 1702
pinTextSamePurpose 1703
GUI Equivalent 1703
pinTextSameLayer 1704
GUI Equivalent 1704

© 2020 All Rights Reserved.

pinTextOrientation 1705
GUI Equivalent 1705
pinWidth 1706
pinXPitch 1707
pinYPitch 1708
pointSelectAperture 1709
pointSelectCycleMode 1710
polygonArc 1711
preservePins 1712
preservePinFigs 1713
preserveRodObjects 1714
preserveTerminalContacts 1715
preserveTermName 1716
preventDeletionOfLogicalInstance 1717
probeCycleBool 1718
probeFilledInstance 1719
probeFilledInstanceParam 1720
probeHighlight 1721
probeHiliteLppList 1722
probeHiliteLppString 1723
probeInstDisplayRadio 1724
probeNetDisplayCheckInstance 1725
probeNetDisplayCheckPins 1726
probeNetDisplayFlightLines 1727
probeNetDisplayHierarchyLevel 1728
probeNetNameFilterByName 1729
probeNetNameFilterByNameDefaultList 1730
probeNetNameFilterByNameMode 1731
probeNetStopAtDisplayLevel 1732
probePromptBusBit 1733
probeShapeDisplay 1734
probeTermDisplayRadio 1735
probeVisibleLpp 1736
propEditHideTopSection 1737
propEditorHeight 1738
propEditorWidth 1739

© 2020 All Rights Reserved.

propertyEditorAutoResize 1740
propEditStretchableTree 1741
protectedObjHighlightHaloLevel 1742
protectedObjHighlightHaloMode 1743
protectedObjHighlightHaloTrans 1744
protectedObjHighlightHaloWidth 1745
pteAllLayersScope 1746
pteApplyParentState 1747
pteAutoRedrawMode 1748
pteCheckableColumnClickMode 1749
pteCollapseLayersInViewBy 1750
pteColoredLppSortOrder 1751
pteColorMixedSort 1752
pteCompactMPT 1753
pteDeferRedrawOnMouseMove 1754
pteDisplayActiveLppInForeground 1755
pteEnableMouseBindings 1756
pteFindModeOn 1757
pteIncludeCutLayers 1758
pteInfraDefValueObjGrids 1759
pteInfraObjSelFollowVis 1760
pteInfraWirebondProfileColors 1761
pteInverseColorScheme 1762
pteLayerSetActivationMode 1763
pteLayerSetManagerLeftClickMode 1764
pteLayersSelectabilityDepth 1765
pteLayersVisibilityDepth 1766
pteLinkColorVisibilityToShapeVisibility 1767
pteLppGroup 1768
pteLppsDisplayOrderControl 1769
pteLSManagerAsAssistant 1770
pteLSManagerRespectFilters 1771
pteMPTMode 1772
pteMPTScope 1773
pteMovableTools 1774
pteMultiSelectionMode 1775

© 2020 All Rights Reserved.

pteRetainFilterSettings 1776
pteRoutingOnlyScope 1777
pteShowColorControlsForMPT 1778
pteShowMPTScope 1779
pteShowUsedSystemLpps 1780
pteUsedLayersEipScope 1781
pteWindowSynchronizationMode 1782
QAHoldTimeMilliSec 1783
QAMaxPixelsPerMilliSec 1784
quickAlignMode 1785
quickAlignSameLayerTarget 1786
quickAlignSnapTarget 1787
quickAlignSpacingType 1788
quickAlignUserSpacing 1789
realColorSelection 1790
realignLabelToPinCenter 1791
rectangularWSPPinAsSquare 1792
rectBaseMode 1793
rectHeight 1794
rectOnSnapPatternGrid 1795
rectSmartSnapEnable 1796
rectSmartSnapOnActiveLayer 1797
rectWidth 1798
recursionCheck 1799
remasterInstCheckTerms 1800
remasterInstSearchCellText 1801
remasterInstSearchLibText 1802
remasterInstSearchViewText 1803
remasterInstUpdateCellText 1804
remasterInstUpdateLibText 1805
remasterInstUpdateViewText 1806
removeChopShape 1807
removeEmptyNetTerm 1808
removeTerminals 1809
replaceFigs 1810
resetOrientation 1811

© 2020 All Rights Reserved.

reshapeType 1812
retainSelectionSet 1813
rodAutoName 1814
rotateAroundCombinedCenter 1815
rotateInstParam 1816
rowAbutFirstUnit 1817
rowAbutNumberOfUnits 1818
rowAbutUnit 1819
rowBaseName 1820
rowCreateMethod 1821
rowChannelWidth 1822
rowFirstUnitOrient 1823
rowFlipFirstUnit 1824
rowFlipUnit 1825
rowNumberOfRows 1826
rowPartitionBottomOffset 1827
rowPartitionLeftOffset 1828
rowPartitionRightOffset 1829
rowPartitionTopOffset 1830
rowNumberOfSites 1831
rowSiteEnable 1832
rowUnitOrient 1833
rulerColorPacketName 1834
rulerDisplayType 1835
rulerEdgeMode 1836
rulerFontBold 1837
rulerFontSize 1838
rulerLabelBackgroundColor 1839
rulerSegmentModeType 1841
saveACopyLibName 1842
saveRulers 1843
schematicDefaultApp 1844
schematicDefaultTier 1845
schematicSymbolDefaultApp 1846
scrollPercent 1847
segSnapMode 1848

© 2020 All Rights Reserved.

selectCreatedObj 1849
selectHilightPacket 1850
selectionAperture 1851
selectMarkersAtLast 1852
selectObjectLargerThanWindow 1853
selectObjectInEditCvFirst 1854
selectOnlyInstWithSelectableShapes 1855
selectProtectedItems 1856
selectUnderCursorMenuAlphaSorting 1857
selectUnderCursorMenuLimit 1858
setDxDyToZero 1859
setupConstraintGroup 1860
showWhatsNew 1861
slotEdgeToSlotOffset 1862
slotEnable 1863
slotEnableRect 1864
slotEnablePath 1865
slotInterSlotSpacing 1866
slotLength 1867
slotRowSpacing 1868
slotShapeMode 1869
slotSpecifyArea 1870
slotStaggerOffset 1871
slotWarnWhenMoreThanRows 1872
slotWidth 1873
snapDeviceInDetailMode 1874
snapMode 1875
snapPatternDrawingStyle 1876
snapPatternFastDetailModeSnapping 1877
snapPatternGlobalGridAnchoring 1878
snapPatternSnapping 1879
snapPatternSnappingDepth 1880
snapPatternSnappingDepthMethod 1881
snapPatternSnappingMode 1882
snapPatternSnapToXYGridFirst 1883
snapRectToWSPGrid 1884

© 2020 All Rights Reserved.

snapResizeWireObjects 1885
snapToMfgGrid 1886
softFence 1887
spineSelection 1888
startLevel 1889
statusBarXYdXdYLength 1890
statusToolbarFields 1891
stopLevel 1892
stretchPcellApplyToName 1893
stretchHandlesLayer 1894
stretchViaMode 1895
supressFreezePcellUI 1896
tapConnectedFigsOnly 1897
taperStyle 1898
tapLayerFilter 1899
tapPurposeList 1900
tapType 1901
targetColor 1902
targetLineStyle 1903
targetWidth 1904
termDirection 1905
termName 1906
textJustificationOn 1907
tinyInstDetail 1908
transientRulerColorPacketName 1909
transparentGroup 1910
trimEnable 1911
trueColorProbe 1912
trlBendFac 1913
trlBendStyle 1914
trlChamFac 1915
trlNPtsToPl 1916
trlRadFac 1917
trlWidth1 1918
trlWidth2 1919
turnInfixOffWhenSmartSnapping 1920

© 2020 All Rights Reserved.

undoRedoHistoryLimit 1921
updateLabelLPPWithPin 1922
updatePCellIncrement 1923
useAlphaBlending 1924
useDynamicHilightPacket 1925
useOverrideFontName 1926
useRelativeXY 1927
useSeparateLAndXlForms 1928
useTapPurposeList 1929
useTrueBBox 1930
validLppFilterOn 1931
viaAlignCuts 1932
viaAutoFromArea 1933
viaAutoViaAlignment 1934
viaAutoViaCutArrayDirection 1935
viaAutoViaEnableMaxStackDepth 1936
viaAutoViaEnclosureDirection 1937
viaAutoViaHints 1939
viaAutoViaMaxStackDepth 1940
viaAutoViaMinNumCuts 1941
viaAutoViaPreviewShow 1942
viaCheckBlockageTypes 1943
viaCheckMinNumCut 1944
viaCheckPlateMaxWidth 1945
viaCheckPlateMinWidth 1946
viaCheckShapeConn 1947
viaCheckUserSet 1948
viaColumns 1949
viaCompuEnclosures 1950
viaComputeFromMode 1951
viaConstraintGroup 1952
viaCreateInRoute 1953
viaCutSpacingMethod 1954
viaCutClassOrientation 1955
viaEnableUserSkillCallbacks 1956
viaEnableValidPurposes 1957

© 2020 All Rights Reserved.

viaExtendEnclosureBeyondOverlap 1958
viaFastEditFromSelectedVias 1959
viaFillEnclosureOverlap 1960
viaHideEnclosures 1961
viaIgnorePurposes 1962
viaMaximizeCuts 1963
viaMode 1964
viaParamCalcMode 1965
viaPreventDRCOnOverlap 1966
viaPreventDRCWithNeighbors 1967
viaPrintStatisticsReport 1968
viaPrintStatisticsReportByStackDepth 1969
viaRows 1970
viaSelectionMode 1971
viaStackColumns 1972
viaStackRows 1973
viaStackSelection 1974
viaStackSelectionMatchingSizes 1975
viaUseEffectiveCuts 1976
viaUseEntireOverlap 1977
viaUseRowColDefault 1978
viaValidateDRCStandalone 1979
viaValidPurposes 1980
viaWallFromSelectedShapesEnabled 1981
viaWallKeepViasEnabled 1982
viaWallPreviewEnabled 1983
viewNameToTypeList 1984
xSnapSpacing 1985
ySnapSpacing 1986
yankShape 1987
zoomToFig 1988
Environment Variables Used Internally 1989

© 2020 All Rights Reserved.

C
Bindkeys in Virtuoso Layout Suite 1991

D
Objects Criteria Table 1993
Objects Criteria Table 1993

E
Layout Editor Assistants 2015
Annotation Browser 2016
Dynamic Selection 2017
Dynamic Selection Assistant Display 2018
Selecting an Object using the Dynamic Selection Assistant 2021
Palette Assistant 2026
Search 2027
Search Categories 2028
Property Editor 2030
Setting Nets on Instance Terminals 2031
Editing a cdsVia 2032
Editing Via Stacks 2032
Determining Objects in Route 2033
Editing Voltage Information 2034
Navigator 2036
World View 2038

F
Commands and Toolbars in Read-Only Mode 2039
Commands 2039
Toolbar Icons 2042

G
Layout Editor Plug-Ins 2049

© 2020 All Rights Reserved.

Preface

The Virtuoso® Layout Suite is an editing tool that provides the basic creation and editing capabilities to work with layout designs. The layout editor lets you create custom layouts or edit any existing layouts. It lets you work with multipart paths and hierarchical designs. You can also use the layout editor to view the connected shapes in a design.

For information about connectivity-based editing, advanced editing features, and automation of design tasks, refer to the Virtuoso Layout Suite documentation.

This user guide describes how to use the layout editor. It is aimed at developers and designers of integrated circuits and assumes that you are familiar with:

- The Virtuoso design environment and application infrastructure mechanisms designed to support consistent operations between all Cadence® tools.
- The applications used to design and develop integrated circuits in the Virtuoso design environment, notably Virtuoso Layout Suite and Virtuoso Schematic Editor.
- The design and use of parameterized cells.
- The OpenAccess version 2.2 technology file.
- Component description format (CDF)

This preface contains the following topics:

- Scope
- Licensing Requirements
- Related Documents
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

© 2020 All Rights Reserved.

Scope

The functionality described in this guide can be used only in ICADVM20.1advanced nodes and advanced methodologies releases.

Label	Meaning
(ICADVM20.1 Photonics Only)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Photonics_Option license (95550).
(ICADVM20.1 Virtuoso RF Solution Only)	Features supported only in the ICADVM20.1 release releases and which require the Virtuoso_RF_Option (95560) license.

Licensing Requirements

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

Related Documents

What's New and KPNS

- *Virtuoso Layout Suite XL What's New*
- *Virtuoso Layout Suite XL Known Problems and Solutions*

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Virtuoso Design Environment SKILL Reference*
- *Cadence Application Infrastructure User Guide*

© 2020 All Rights Reserved.

Technology Information

- *Virtuoso Technology Data User Guide*
- *Virtuoso Technology Data ASCII Files Reference*
- *Virtuoso Technology Data SKILL Reference*

Virtuoso Tools

- *Virtuoso Layout Viewer User Guide*
- *Virtuoso Layout Suite XL: Connectivity Driven Editing Guide*
- *Virtuoso Layout Suite EXL Reference*
- *Virtuoso Concurrent Layout User Guide*
- *Virtuoso Design Planner User Guide*
- *Virtuoso Multi-Patterning Technology User Guide*
- *Virtuoso Placer User Guide*
- *Virtuoso Simulation Driven Interactive Routing User Guide*
- *Virtuoso Width Spacing Patterns User Guide*
- *Virtuoso RF Solution Guide*
- *Virtuoso Electromagnetic Solver Assistant User Guide*

Relative Object Design and Inherited Connections

- *Virtuoso Relative Object Design User Guide*
- *Virtuoso Schematic Editor User Guide*

SKILL Documents

- The SKILL programming language is documented in the following manuals:
  - *Virtuoso Design Environment SKILL Reference*
  - *Cadence SKILL Language User Guide*
  - *Cadence SKILL Language Reference*

© 2020 All Rights Reserved.

- *Cadence SKILL Development Reference*
- *Cadence SKILL IDE User Guide*

- SKILL access to other applications is provided in the following manuals:
  - *Virtuoso Technology Data SKILL Reference*
  - *Virtuoso Layout Suite SKILL Reference*
  - *Virtuoso Schematic Editor SKILL Reference*
  - *Cadence User Interface SKILL Reference*
  - *Cadence Interprocess Communication SKILL Reference*

Additional Learning Resources

Video Library

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

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

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

Virtuoso Videos Book

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

Rapid Adoption Kits

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

In addition, Cadence offers the following training courses on Virtuoso Layout Suite L:

© 2020 All Rights Reserved.

- Virtuoso Layout Design Basics
- Virtuoso Layout Pro: T1 Environment and Basic Commands (L)
- Virtuoso Layout Pro: T2 Create and Edit Commands (L)
- Virtuoso Layout Pro: T3 Basic Commands (XL)
- Virtuoso Connectivity-Driven Layout Transition
- Virtuoso Schematic Editor

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

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

Help and Support Facilities

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

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

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

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

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

© 2020 All Rights Reserved.

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

- Log on to Cadence Online Support

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

Feedback about Documentation

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

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

You can also submit feedback by using the following methods:

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

Typographic and Syntax Conventions

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

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
|	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.

© 2020 All Rights Reserved.

`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument. Used with brackets to indicate that you can specify zero or more arguments. Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence® SKILL® language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

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

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

1

Layout Editor Basics

The Virtuoso Layout Suite is comprised of various types of tools which are described in the following documents.

- *Virtuoso Interactive and Assisted Routing User Guide*
- *Virtuoso Design Rule Driven User Guide*
- *Virtuoso Parameterized Cell Reference*
- *Plotting Configuration User Guide*

This chapter contains these topics:

- The Virtuoso Design Environment
- Layout XL Desktop
- Layout XL Window
- Working with Designs
- Command Function Basics
- Using Infix
- Menu Access Keys
- Using Bindkeys to Run Commands
- Setting Defaults
- Selecting the Workspace
- Using Environment Variables
- Setting Layout Editor Options
- Setting Layout Editor Options
- Display and Selection Controls

© 2020 All Rights Reserved.

- Using the Technology File
- Layout Editor Strokes
- Using Forms
- Using Context-Sensitive Menus
- Performing Fast GDS Export Using XStream Out
- Using the Allegro PCB/Package Menu

© 2020 All Rights Reserved.

The Virtuoso Design Environment

Basics of the Virtuoso Design Environment

The Virtuoso Design Environment User Guide provides information you need to start using the Virtuoso Layout Suite L layout editor. Listed below are sections that provide a basic introduction to the layout suite environment.

- Starting the Layout Editor

  For information about how to start or quit the software, backup your work, or save your designs, see Getting Started with Virtuoso Software.

- About the Command Interpreter Window

  For information about the CIW window, such as CIW menus, viewing commands, system responses, and typing in SKILL commands, see Using the Command Interpreter Window.

- Working with cellviews

  For information about open and closing cellviews, and setting the default application, see Working with Cellviews.

- Working with Workspaces

  For information about how to select or configure workspaces, see Getting Started with Workspaces and Working with Workspaces.

- The User Interface

  For information about the user interface, see Interacting with the User Interface.

- Session Window and Workspace

  For information about session windows, see Sample Session Window and Workspace.

Starting Layout XL

To use Layout XL, you must have access to one of the following design environments:

- Virtuoso Physical Design Environment (`layoutPlus`)
- Virtuoso Front to Back Design Environment (`virtuoso`)

To start a design environment that includes the Layout XL software:

© 2020 All Rights Reserved.

1. In a terminal window, move into the top-level directory of the design you want to edit.

   The top-level directory is the directory that contains the library definitions (`cds.lib`) file for the design in question. The library definitions file defines the physical locations of all the reference and design libraries. Launching Layout XL loads the library initialization (`libInit.il`) files related to all these reference and design libraries.

2. Type in the executable name of the design environment you want to launch; for example, `virtuoso`.

   The Command Interpreter Window (CIW) is displayed.

   You use the CIW to launch applications on specific cellviews; perform general, non-tool-specific operations; enter SKILL commands; and display messages and warnings.

After you have started the design environment, you can then launch Layout XL from

- A schematic view, with or without a physical configuration view. See Starting Layout XL from a Schematic View.
- A layout view, with or without a connectivity reference. See Starting Layout XL from a Layout View.
- A physical configuration view. See Starting Layout XL from a Physical Configuration View.

Starting Layout XL from a Schematic View

A schematic cellview is the only valid connectivity reference for a Layout XL session. If you do not have a schematic view of your design, you first need to generate one.

When you start Layout XL from a schematic, you need to specify

- Whether to create a new layout view or use an existing one.

© 2020 All Rights Reserved.

- Whether the layout view is opened in the context of a *new* physical configuration view, an *existing* physical configuration view, or an *automatic* physical configuration view.

  Note: If you do not need to use any of the features provided by the *Configure Physical Hierarchy* command, you can choose Automatic mode. This ensures that Layout XL functions the same way as in previous releases, handling all operations related to the physical configuration view automatically.

When Layout XL initializes, the schematic view is re-opened in the context of the physical configuration view being used for the session, which might be different from the configuration the schematic was using previously. The schematic window banner updates to indicate the name of the physical configuration view currently being used.

One effect of this is that you might need to re-extract the schematic view in order to take into account the settings in the physical configuration view. The system will inform you if this extraction is required.

If you close the Layout XL session by closing the layout window only, the schematic view is re-opened with no physical configuration context.

Tip

For more information on physical configuration views, see Configuring the Physical Hierarchy.

For more information on starting Layout XL from a schematic view, see

- Starting Layout XL with an Automatic Physical Configuration View
- Starting Layout XL with a New Physical Configuration View
- Starting Layout XL with an Existing Physical Configuration View

Generating a Schematic View from a CDL Netlist

If your connectivity reference is a CDL netlist, you cannot import it directly into Layout XL. You must first generate a schematic from the netlist.

To generate a schematic view from a CDL netlist,

1. From the CIW, select *File – Import – Spice*.
2. Set the *Netlist Language* to *CDL* and type the *Netlist File* name in the field provided.
3. Click *OK*.

© 2020 All Rights Reserved.

4. Start Layout XL using one of the methods described.

For more information on importing CDL, see Netlist import Using Spice In in the *Design Data Translator's Reference*.

Starting Layout XL with an Automatic Physical Configuration View

If you do not need to use any of the features provided by the Configure Physical Hierarchy command, you should launch Layout XL in Automatic mode. This ensures that Layout XL handles all operations related to the physical configuration view automatically.

To start Layout XL from a schematic view in Automatic mode,

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

   The Open File form is displayed.

2. Specify the *Library* and *Cell* names, choose *schematic* from the *View* cyclic field and click *OK*.

   The Virtuoso Schematic Editor L Editing window appears displaying the cellview you specified.

© 2020 All Rights Reserved.

3. From the schematic window, choose *Launch – Layout XL*.

   Note: *Layout XL* appears in the *Launch* menu only if the Layout XL software is installed. If you do not see *Layout XL* in the menu, check with your system administrator.

   The Startup Option form is displayed.

4. In the *Layout* group box, choose whether you want to create a new layout cellview or open an existing one.

5. In the *Configuration* group box, choose *Automatic* and then click *OK*.

   One of the following happens.

© 2020 All Rights Reserved.

- If you chose to create a new layout view, the New File form is displayed.

The default view name is `layout`.

Choose a *Library* name from the cyclic field and type the *Cell* and *View* names in the fields provided.

Click *OK* to create the new cellview.

The software automatically creates the new layout cellview and looks for an existing physical configuration view called either `physConfig` or `physConfig_`*`schName`*, which might have been created in a previous Layout XL session or when Converting Data to Use the IC 6.1 Layout XL Schema.

If neither view exists, it creates a new one called `physConfig` or `physConfig_`*`schName`*.

© 2020 All Rights Reserved.

- If you chose to open an existing layout view, the Open File form is displayed again.

Specify the *Library* and *Cell* names and choose the *View* you want to open from the cyclic field.

Note: You cannot change setting for the *Open with* option in the *Application* group box, regardless of the value set for the maskLayoutDefaultApp environment variable.

Click *OK* in the form to open the cellview.

If there is an existing physical configuration view associated with the layout cellview, the software opens it in the background. If there is no existing physical configuration, the software creates a temporary, default one called `physConfig` or `physConfig_`*`schName`*.

If the physical configuration view, `physConfig`, associated with a layout cellview is currently open and it is a temporary, default physical configuration view; the physical configuration view cannot be accessed by any other user at the same time. Therefore, if another user wants to open the same design in read-only mode, another temporary physical configuration view, `physConfig_2`, gets created. Likewise, if a third user now wants to open the same design in read-only mode, another temporary physical configuration view, `physConfig_3` gets created.

© 2020 All Rights Reserved.

To conclude, for any incremental views of the same design that require the layout cellview to be opened in read-only mode, a new, temporary physical configuration view gets automatically generated and the name of each new physical configuration view increases incrementally.

The default Layout XL desktop configuration appears: the schematic window, the layout window, the CIW, and the Palette Assistant. For more information, see Layout XL Desktop.

The schematic view is re-opened in the context of the automatic physical configuration view, which might be different from the configuration it was using previously. The schematic window banner updates to indicate the name of the automatic physical configuration view.

Note: One effect of this is that you might need to re-extract the schematic view in order to take into account the settings in the physical configuration view. The system will inform you if this extraction is required. To avoid the need for re-extracting the schematic view, and to have the tool automatically copy the physconfig settings from the previous view, you must copy the cellview using the Library Manager. For more information, see Copying a View.

The Configure Physical Hierarchy window is not displayed in Automatic mode. However, you can invoke it at any time by choosing *Launch – Configure Physical Hierarchy* from the layout window menu bar.

Note: Even in Automatic mode, you still need to use the Configure Physical Hierarchy window if you want to change the switch, stop, and constraint view lists, or make changes to component type for soft block definitions. For more information, see Configuring the Physical Hierarchy.

Tip

If Layout XL does not initialize and instead you see the Layout XL Constraint Group Setting dialog, see Specifying the Constraint Group.

Starting Layout XL with a New Physical Configuration View

To start Layout XL from a schematic view in the context of a new physical configuration view,

1. Complete steps **1** through **4** described in Starting Layout XL with an Automatic Physical Configuration View.

© 2020 All Rights Reserved.

The Startup Options form is displayed.

2. Choose whether you want to create a new layout cellview or open an existing one, make sure that the *Configuration* option is set to *Create New*, and then click *OK*.

One of the following happens.

- If you chose to create a new layout cellview, the New File form is displayed.

Choose a *Library* from the cyclic field and type the *Cell* and *View* names in the fields provided.

© 2020 All Rights Reserved.

- If you chose to open an existing layout view, the Open File form is displayed again.

Specify the *Library* and *Cell* names and choose the *View* you want to open from the cyclic field.

Note: You cannot change setting for the *Open with* option in the *Application* group box, regardless of the value set for the maskLayoutDefaultApp environment variable.

3. Click *OK* to create the new layout cellview or to open an existing one.

© 2020 All Rights Reserved.

The Create Physical Configuration View form is displayed.

The layout *Library* and *Cell* names are seeded automatically.

4. Specify the *View Name* for the physical configuration view.

   To seed the new view with predefined values, check *Use Template* and either choose one of the predefined templates from the drop-down list or specify a file stored in another location. If you do not specify a template, the corresponding environment variables are used to obtain initial values.

   Check the *Open CPH* check box to open the Configure Physical Hierarchy window automatically.

5. Click *OK*.

   The default Layout XL Desktop configuration appears: the schematic window, the layout window, the CIW, and the Palette assistant.

   The schematic view is re-opened in the context of the new physical configuration view, which might be different from the configuration it was using previously. The schematic window banner updates to indicate the name of the new physical configuration view. The Configure Physical Hierarchy window opens to display the physical configuration view you created.

   Note: You might need to re-extract the schematic view in order to take into account the the settings in the physical configuration view. The system will inform you if this extraction is required.

© 2020 All Rights Reserved.

Tip

If Layout XL does not initialize and instead you see the Layout XL Constraint Group Setting dialog, see Specifying the Constraint Group.

Starting Layout XL with an Existing Physical Configuration View

To start Layout XL from a schematic view in the context of an existing configuration view,

1. Complete steps **1** through **4** described in Starting Layout XL with an Automatic Physical Configuration View.

   The Startup Option form is displayed.

2. Choose whether you want to create a new layout cellview or open an existing one and make sure that the *Configuration* option is set to *Open Existing*.

3. Click *OK*.

   One of the following happens.

© 2020
All Rights Reserved.

- If you chose to create a new layout cellview, the New File form is displayed.

Choose a *Library* from the cyclic field and type the *Cell* and *View* names in the fields provided.

© 2020 All Rights Reserved.

- If you chose to open an existing layout view, the Open File form is displayed again.

Specify the *Library* and *Cell* names and choose the *View* you want to open from the cyclic field.

Note: You cannot change setting for the *Open with* option in the *Application* group box, regardless of the value set for the maskLayoutDefaultApp environment variable.

4. Click *OK* to create the new layout cellview or to open an existing one.

© 2020 All Rights Reserved.

The Open Physical Configuration View form is displayed.

5. Specify the *Library*, *Cell*, and *View* names and check the *Open CPH* box to open the Configure Physical Hierarchy window automatically.
6. Click *OK*.

   The default Layout XL desktop configuration appears: the schematic window, the layout window, the CIW, and the Palette assistant.

   The schematic view is re-opened in the context of the specified physical configuration view, which might be different from the configuration it was using previously. The schematic window banner updates to indicate the name of the specified physical configuration view and the Configure Physical Hierarchy window opens.

   Note: You might need to re-extract the schematic view in order to take into account the the settings in the physical configuration view. The system will inform you if this extraction is required.

Starting Layout XL from a Layout View

To start Layout XL from a layout view,

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

   The Open File form is displayed.
2. Specify the *Library* and *Cell* names and choose *schematic* from the *View* cyclic field.
3. Click *OK*.

   The Virtuoso Layout Suite L Editing window appears displaying the cellview you specified.

© 2020 All Rights Reserved.

4. From the layout window, choose *Launch – Layout XL*.

   The default Layout XL Desktop configuration appears: the schematic window, the layout window, the CIW, and the Palette assistant. The schematic view is opened in the context of the physical configuration view being used for the session, which might be different from the configuration it uses when it is opened standalone.

   Note: You might need to re-extract the schematic view in order to take into account the the settings in the physical configuration view. The system will inform you if this extraction is required.

Tip

By default, the schematic editor launches in the XL mode. To launch the editor in the L mode instead, select the *Schematics L* option for the *Open with* field on the Connectivity form.

The corresponding environment variable is lxSchematicDefaultApp.

Notes

- *Layout XL* appears in the *Launch* menu only if the Layout XL software is installed. If you do not see *Layout XL*, check with your system administrator.

Starting Layout XL from a Physical Configuration View

To start Layout XL directly from a physical configuration view,

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

   The Open File form is displayed.

2. Specify the *Library* and *Cell* names and choose *physConfig* from the *View* cyclic field.
3. Click *OK*.

© 2020 All Rights Reserved.

The *Open Configuration or Top Cellview* form is displayed.

4. Set both *Configuration* and *Top Cellview* to *yes* to open both the physical configuration view and the schematic view.

   Note: You can set defaults for this form using the following environment variables.

   ```
   envSetVal("ddserv.he" "hierEditor" 'boolean t)
   envSetVal("ddserv.he" "schEditor" 'boolean t)
   ```

5. Click *OK*.

   The schematic view is opened in a Virtuoso Schematic Editor window and the Configure Physical Hierarchy window is displayed showing the physical configuration view you specified. The system checks out a Layout XL license if there is not one currently checked out.

6. Start Layout XL as detailed in Starting Layout XL from a Schematic View.

Automatic Constraint Transfer during Layout XL Startup

On startup, Layout XL transfers all the constraints defined in the schematic view to the layout view; correctly mapping the constraints and their members between the two views.

Constraints in the schematic are transferred to the top-level layout view. Constraints that have been created or changed in the schematic but not yet saved are also transferred.

Device correspondence information is maintained during the transfer. and physical name mappings are tracked and constraints are updated appropriately to take account of folded instances in the layout.

© 2020 All Rights Reserved.

Specifying the Constraint Group

The Layout XL connectivity extractor requires a constraint group containing information on valid layers, valid vias, and error layers. You specify the default constraint group using a layout environment variable called setupConstraintGroup.

If the specified constraint group does not contain the information required by Layout XL, during initialization the software pops up a dialog inviting you to either:

- Choose the constraint group you want from the ones specified in the technology file.
- Launch XL with the connectivity extractor disabled.

Default Parameter and Property Names

Virtuoso Layout Suite XL requires the following parameter and property names to be reserved for correctly setting up the environment.

Parameter or Property name	Reserved name used for Layout XL set up
Capacitance	c C
Fingers	finger numFingers numFinger

© 2020 All Rights Reserved.

Inductance	l L
Parallel-connected factor	m M
Resistance	r R
Segments	segments
Series-connected factor	s S
Transistor	w

Note: For each parameter, the name that takes precedence is:

1. Parameter name set at the instance level, if an instance override is set
2. Parameter name defined at the cell level, if an instance override is not set
3. The first parameter name defined in CDF (Common Description Format, if different names are set at the instance and cell levels

Layout XL Desktop

When you start Layout XL, the four windows are rearranged into the default window configuration on your desktop. The layout window is on the right with “Virtuoso® Layout Suite XL...” in the banner; the schematic that corresponds to the layout is in the schematic window

© 2020 All Rights Reserved.

on the left. The Palette assistant and the command interpreter window (CIW) are placed to the left and bottom of the screen respectively.

You can move, resize, and iconify any of the four windows.

Customizing the Desktop

You can save and reuse your customized desktop layout in one of the following ways.

- Using the *Options – Save Defaults* command from the CIW.

  When you use this command, Layout XL saves the size and position of the schematic and layout windows as a property on the current layout cellview. It also sends the Palette assistant and CIW window size and location information to the `.cdsenv` file.

© 2020 All Rights Reserved.

When you reopen the layout and schematic cellviews, they appear in the same configuration as they had when you used the *Save Defaults* command.

For more information, see Saving and Recalling Default Settings in the *Virtuoso Design Environment User Guide*.

- By adding the appropriate environment variables to your `.cdsenv` file; for example, to prevent Layout XL from rearranging or resizing windows, add the following line to the `.cdsenv` file:

```
envSetVal("layoutXL" "autoArrange" 'boolean nil)
```

Note: If you save the current window positions and form settings to a file other than `.cdsenv` (for example, `.envFileName`), you can restore the settings by adding the following line to your `.cdsinit` file or typing it in the CIW.

```
envLoadFile("layoutXL" ".envFileName")
```

Layout XL Window

The Layout XL window comprises a menu bar, a canvas, in which you can display and manipulate your design graphically, and a number of toolbars and dockable assistants to help you with your design tasks. This guide describes only those toolbars and assistants as they relate to Layout XL design tasks.

- For information on the individual components of the schematic window, see the *Virtuoso Schematic Editor L User Guide*.
- For detailed information on the new design environment, see the *Virtuoso Design Environment User Guide*.
- For an overview of the connectivity-driven features supported by Layout XL and a lab database and workshop instructions to help you explore how these features can bring productivity gains for a design, view or download the rapid adoption kit, *Introduction to Connectivity-Driven Design in VLS XL*.

Virtuoso Layout Launch Menu

The Virtuoso Layout launch menu enables you to launch the applications listed below:

- ADE L
- ADE XL
- ADE GXL
- ADE Explorer

© 2020 All Rights Reserved.

- ADE Assembler
- Layout XL
- Layout EXL
- Pcell IDE
- Configure Physical Hierarchy
- Plugins
  - Debug Abutment
  - Debug CDF
  - Innovus
  - Parasitics
  - Pcell
  - Simulation
    - NC-Verilog
    - Other
    - Spectre
    - UltraSim
  - VPS-L
  - Voltus-Fi-L
  - Voltus-Fi-XL

Using the Library Browser

For information about the Library Browser, see the Cadence Library Manager User Guide.

Cadence Licensing

For startup information such as setting the installation path, executable, and initialization files, see the Virtuoso Software Licensing and Configuration Guide.

© 2020 All Rights Reserved.

For information about how to run 64-bit software, see Running 64-Bit Versions of Applications in the *Virtuoso Software Licensing and Configuration Guide*.

Working with Designs

- Opening a Design
- Making a Design Read Only
- Making a Design Editable
- Saving a Design
- Closing a Design

Opening a Design

- Opening a New Design
- Opening an Existing Cellview

Opening a New Design

To open a new cellview from the layout editor:

1. Choose *File – New* from an editor window or the CIW.

© 2020 All Rights Reserved.

The *New File* form appears.

2. From the *Library* drop-down list, choose the library in which to store the new cellview.
3. In the *Cell* field, enter a new cell name.
4. In the *View* field, enter a name for the new cellview.
5. In the *Type* drop-down list, choose the view type, such as layout, that is to be associated with the new cellview.
6. In the *Open with* drop-down list, choose the application that the new cellview should automatically open with.
7. Optionally, select the *Always use this application for this type of file* check box to associate the selection made in the *Open with* field with the current *View* type selected. This will save you in future having to select what application should be opened with a selected cellview *Type*.
8. The *Library path file* field displays the location where the new cellview will be stored.
9. Choose whether you want to open the new cellview in a *new tab*, the *current tab*, or in a *new window*.
10. Click *OK*.

© 2020 All Rights Reserved.

Note: You can use an existing layout cellview as a template to auto-create WSP and row region specifications in the design, using the environment variable, autoCopyTemplateCellViewList. Auto-creation works only while creating a new cellview.

Opening an Existing Cellview

To open an existing cellview in the layout editor:

1. Choose *File – Open* from an editor window or the CIW.

   The *Open File* form appears.

2. In the *Library* drop-down list, choose the library that contains the cellview you want to open.

3. In the *Cell* field, specify a cell name using one of the following methods:

   - Type the cell name.
   - Click a cell name in the *Cells* list box on the right.
   - Click *Browse* to select the cell in the *Library Browser*.

4. Choose the *View* to be open from the drop-down list.

© 2020 All Rights Reserved.

The *Type* field will update to display the type of view of the selected cell view.

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

5. From the *Open with* down-down list, choose the application that the cellview should open with.
6. Optionally, select the *Always use this application for this type of file* check box to associate the selection made in the *Open with* drop-down list with the current *View* type selected. This will save you in future having to select what application should be opened with a selected cellview *Type*.
7. Choose to open the cellview in *edit* or *read* mode.
8. The *Library path file* field displays the location where the cellview is stored.
9. Choose to open the cellview in a *new tab* in the current session window, the *current tab* in the current session window, or a *new window* in a new session.
10. Click *OK*.

For more information about cellview tabs, see Working with Cellviews in the *Virtuoso Design Environment User Guide*.

Opening an Existing Cellview

To open an existing cellview in the layout editor:

1. Choose *File – Open* from an editor window or the CIW.

© 2020 All Rights Reserved.

The *Open File* form appears.

2. In the *Library* drop-down list, choose the library that contains the cellview you want to open.

3. In the *Cell* field, specify a cell name using one of the following methods:

   - Type the cell name.
   - Click a cell name in the *Cells* list box on the right.
   - Click *Browse* to select the cell in the *Library Browser*.

4. Choose the *View* to be open from the drop-down list.

   The *Type* field will update to display the type of view of the selected cell view.

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

5. From the *Open with* down-down list, choose the application that the cellview should open with.

© 2020 All Rights Reserved.

6. Optionally, select the *Always use this application for this type of file* check box to associate the selection made in the *Open with* drop-down list with the current *View* type selected. This will save you in future having to select what application should be opened with a selected cellview *Type*.
7. Choose to open the cellview in *edit* or *read* mode.
8. The *Library path file* field displays the location where the cellview is stored.
9. Choose to open the cellview in a *new tab* in the current session window, the *current tab* in the current session window, or a *new window* in a new session.
10. Click *OK*.

For more information about cellview tabs, see Working with Cellviews in the *Virtuoso Design Environment User Guide*.

Making a Design Read Only

To make a design open in read-only mode,

➡ Choose *File – Make Read Only* from the editor window.

If you have made any changes to the current cellview but have not saved them, the *Change Edit Mode* dialog box appears and prompts you to either save the changes or discard them.

After you click *Yes* or *No*, the mode becomes read-only. *Yes* saves edits, and *No* discards your edits. You can click *Cancel* to let the design remain in edit mode.

This command toggles with *File – Make Editable*.

Making a Design Editable

To make a design opened in read-only mode, editable,

- Choose *File – Make Editable* from the editor window.

If you have made no change in read-only mode, the design will switch to the editable mode, and no pop-up window will be displayed.

However, if you have made any changes in read-only mode, and reopen the design in edit mode, then:

- If the design has changed on disk either because of data management, or because it was edited and saved by another user, a warning message will be displayed in CIW indicating

© 2020 All Rights Reserved.

that the design has been refreshed. As a result, changes you have made in read-only mode will be lost.

- If the design has not changed on disk, a dialog box will be displayed informing you that you have changed the design in read-only mode, and you can choose *Yes* to keep the changes or *No* to discard them.

Saving a Design

To save a design:

➡ Choose *File – Save*.

To save a design with a different name:

1. Choose *File – Save a Copy*.

   The *Save a Copy* form appears.

2. Select a library from the first drop-down list at the bottom of the form. This should be the library in which you want to save the cellview.
3. Specify a new cell name in the text field at the bottom of the form.
4. Select the cellview type from the last drop-down list.

© 2020 All Rights Reserved.

5. Select the *Open Saved Copy* check box to open a saved copy of the design.
6. Select read or edit from the *Open for* field.
7. Select the location where you want to open the design from the *Open in* field, *new tab*, *current tab*, or *new window*.
8. Click *OK*.

Note: You can access the *Save a Copy* command if a design is opened in read mode. The command checks out the VLS L license and checks it back in after the save operation is complete.

Thumbnail images, where they have been set up in the current view for pre-viewing cellviews, are updated automatically when you choose to *Save* or *Save a Copy* of that cellview. This happens through an automatic call to the `hiGenerateThumbnails` SKILL command. To disable the auto-generation of thumbnails, you can set the following environment variable in the CIW:

```
envSetVal("ui.thumbnails" "generate" `boolean nil)
```

For more information about thumbnails, see Thumbnail View Functions in the *Cadence User Interface SKILL Reference* and Thumbnail Images of Cellviews in the *Cadence Library Manager User Guide*.

To save hierarchical instances, which have been modified without running the *Edit In Place* or *Descend Edit* command, in a design:

➡ Choose *File – Save Hierarchically*.

For more information about the *Edit In Place* and *Descend Edit* commands, see Working With Hierarchical Designs

Closing a Design

To close an editor window,

➡ Choose *File – Close*.

© 2020 All Rights Reserved.

If you edited the current cellview since the last time you saved your data, the *Save Changes* dialog box appears and prompts you to save the changes, discard them, or cancel the close operation.

- *Yes* saves your edits and closes the window.
- *No* discards your edits and closes the window.

  Note: Use the discardCVEditOnWindowClose environment variable to preserve the edits in the memory when you click *No*.
- *Cancel* cancels the *Close* command but does not discard the changes. It returns you to the cellview you were editing.

Command Function Basics

Starting Commands

To start Layout Suite commands you can,

- Choose a command from pull down menus
- Click a toolbar icon
- Choose a command from the layout editor context-sensitive menu: See Using Context-Sensitive Menus.
- Press a bindkey associated with a Command See Using Bindkeys to Run Commands.
- Draw a stroke in the design window: See Layout Editor Strokes.

© 2020 All Rights Reserved.

Canceling Commands

To cancel a command without changing your data, or to stop a command that automatically repeats, press *Esc* or click *Cancel* on the form.

Note: *Modify Corner* and *Size* commands are exceptions. To cancel these commands, click *Cancel* on the form.

Undoing Commands

To undo a command, choose *Edit – Undo* or Press `u`.

You can undo up to 128 previous commands. You can set the undo limit to either 0 or 128. Set the number of commands by choosing *Options – User Preferences* in the Command Interpreter Window (CIW).

The system keeps an undo history (check points) in a single, global stack. Therefore, when you choose Undo in a cellview, the last edit made to any open cellview is undone.

Note: Once you have saved a cellview, the edits made prior to saving the cellview cannot be undone.

Redoing Commands

To reinstate a change you canceled with *Undo*, choose *Edit – Redo* or press `Shift-u`.

Repeating Commands

Many layout editor commands automatically repeat. For example, each time you finish drawing one rectangle, the *Create Rectangle* command prompts you to draw another one.

By default, all commands on the *Create* menu and the *Create Measurement* command are set to repeat. To set the automatic repeat (on or off), choose *Options – Editor* and set *Repeat Commands* on (or off).

Nesting Commands

You can pause before completing one command to perform a second command. This is called nesting a command. For example, if you move an object and then zoom out, the zoom command would be nested, allowing you to use another command such as moving another object.

© 2020 All Rights Reserved.

You can nest any display command. You cannot nest edit or create commands. As exceptions, edit and create commands can be nested within the *Modify Corner* and *Size* commands.

You can nest up to 20 commands. To set the number of commands you can nest, choose *Options – User Preferences* from the CIW and set the *Nest Limit* field to the number you want.

Discarding All Edits

The Discard Edits Form ignores all edits you made since the last time you saved.

To cancel your edits, choose *File – Discard Edits*. A Discard Edits dialog box appears and asks you to confirm that you want to discard your edits.

Handling Unexpected Results While Using Commands

- The Command Starts Before I Enter Points

  If you press a bindkey to start a command and the command seems to start before you click a point, infix mode is probably on. When infix is on, any command you start with a bindkey uses the current pointer location as its first point.

- Stopping a Command

  - If you cannot stop a command, one of the following might be happening:
    - Each time you finish the command, it restarts. This means you are using a command that automatically repeats.
    - A form appears on screen, and you want to close it without executing the command.

  In either case, cancel the command by pressing `Escape` or click *Cancel* on the form.

  It is not always necessary to cancel a repeating command. If you are running a repeating command and then start a nonrepeating command, the repeating command pauses while the new command runs. After you are finished with the nonrepeating command, the repeating command continues. Also, if you start another repeating command, the first repeating command stops.

- The Command Does Not Repeat and It Should

  By default, editing commands repeat automatically only if you first choose the command, then select the object.

© 2020 All Rights Reserved.

Commands *do not* repeat if you first select the object, then choose the command.

If you chose an editing command before selecting an object, and the command still does not repeat, one of the following might be the cause:

- An object is selected that you cannot see.

  Look at the number next to *Select* in the Status Banner or the Status Toolbar to see if one or more objects is selected. For example, if one object is selected the Status Banner or the Status Toolbar display:

  ```
  Select: 1
  ```

  Zoom out the cellview so you can see all objects.

- The *Repeat Commands* setting for the editor has been turned off.

  To turn the *Repeat Commands* setting on, choose *Options – Editor* and in the Layout Editor Options form, set Re*peat Commands* on.

Using Infix

You can set the layout editor to automatically use the current pointer location as the starting point whenever you start commands with bindkeys. This is called infix mode.

To set infix mode on or off, from the CIW, choose *Options – User Preferences* and turn the *Infix* button on or off.

Menu Access Keys

Menu access keys provide keyboard access to application menus and the functionality they contain without the need to use mouse selections. For example, if you press `Alt-F` on your keyboard, the software displays the contents of the *File* banner menu.

The menu access key for an Layout XL menu, submenu, or command is the underlined letter in the menu command label. For example, to display the Connectivity form using only menu access keys,

1. With the cursor in the Layout XL window, press `Alt-O` to display the contents of the *Options* banner menu.
2. Press `Alt-X` to display the Layout XL Options form.

© 2020 All Rights Reserved.

Using Bindkeys to Run Commands

Many layout editor commands can be run by using bindkeys. If the bindkey settings are loaded, the bindkeys appear to the right of the command name on the menu. The physical representation of the bindkey settings is covered in Bindkeys in Virtuoso Layout Suite.

There is a list of default bindkeys for Layout XL at the following location.

```
your_install_dir/samples/local/lxBindKeys.il
```

Where *your_install_dir* is the directory in which you store Cadence software.

For information about bindkeys, see Bindkeys and Access Keys.

Default Bindkeys Defined for Layout XL

There is a list of default bindkeys for Layout XL at the following location.

```
your_install_dir/samples/local/lxBindKeys.il
```

Where *your_install_dir* is the directory in which you store Cadence software.

This file defines the following bindkeys for Layout XL.

Key	SKILL	Menu
`F10`	nclToggleCAEMode()	*Options – Layout XL – General* tab; *Constraint-aware editing* option
`Ctrl+h`	geDSAFreeze(hiGetCurrentWindow() hiGetPoint(hiGetCurrentWindow()))	*Freeze DSA* and *Unfreeze DSA* from the RMB context menu
`Ctrl+l`	lxProbeRemoveAll()	
`Ctrl+m`	lxHiSwapComps()	*Edit – Advanced – Swap Components*
`Shift+l`	lxHiProbe()	*Connectivity – XL Probe*
`RMB`	lxCmdOptions()	
`Shift+RMB`	lxCmdShiftOptions()	
`Shift+Ctrl+h`	lxToggleShowAllIncompleteNets()	*Show Hide All Incomplete Nets*

© 2020 All Rights Reserved.

Key	SKILL	Menu
`h`	lxToggleShowIncompleteNets()	*Show Hide Current Incomplete Nets*
`Shift+h`	lxShowHideIncompleteNets()	*Show/Hide Selected Incomplete Nets*

Displaying Bindkeys

To display the list of bindkeys for Layout XL,

1. From the CIW menu bar, choose *Options – Bindkeys*.

   The Bindkey Editor form is displayed.

   Note: The *Application Tree* to the left of the form, by default, initially displays all the applications for which bindkeys have been defined.

2. In the *Application Tree*, select the application whose bindkeys you want to view.

   Note: The *BindKey Table*, to the right of the form, will update to list the bindkeys associated with the currently selected application.

3. Optionally, use the *Search* field to filter the bindkey listing.

© 2020 All Rights Reserved.

4. To view the bindkey settings for the currently selected application, either:

- Scroll through the listed bindkeys in the *BindKey Table t*o view the current settings for one or more bindkeys, or

© 2020 All Rights Reserved.

- Select the *Preview* button to display a listing of all of the bindkeys associated with the selected application in a text window.

Layout Application Bindkeys

File Help

```
;;-----------------------------------------------------------------------------
;; Bindkeys for 'Layout'
;; Inherited by:
;;             * Dracula Interactive
;;             * NC-Verilog-MaskLayout
;;             * Other-Layout
;;             * Other-MaskLayout
;;             * Other-Symbolic
;;             * Pcell
;;             * Power IR/EM
;;             * SiP-Layout
;;             * Spectre-Layout
;;             * Spectre-MaskLayout
;;             * Spectre-Symbolic
;;             * UltraSim-Layout
;;             * UltraSim-MaskLayout
;;             * UltraSim-Symbolic
;;             * VLS-GXL
;;             * Virtuoso CE
;;             * Virtuoso XL
;;             * adegxl-maskLayout
;;             * adexl-maskLayout
;;             * parasitics-MaskLayout
;;-----------------------------------------------------------------------------
hiSetBindKeys( "Layout" list(
    list("<DrawThru1>" "leSelBoxOrStretch()"  "geSingleSelectBox()")
    list("<DrawThru2>" ""          "_cmdOptionDrawThru2()")
    list("<DrawThru3>" "hiZoomIn()"            "hiZoomIn()")
    list("<Key>#"      "hiToggleAnchorMagnifier()")
    list("<Key>'"      "hiUpdateMagOptions()")
    list("<Key>."      "leToggleMagnifier()")
    list("<Key>A"      "leHiQuickAlign()")
    list("<Key>B"      "leReturnToLevel()")
    list("<Key>BackSpace"          "deletePoint()"        "deletePoint()")
    list("<Key>C"      "leHiCopy()")
    list("<Key>D"      "geSubSelectPoint()")
    list("<Key>Delete" "leHiDelete()"          "leHiDelete()")
    list("<Key>Down"   "geScroll(nil \"s\"    nil)")
    list("<Key>E"      "leHiEditDisplayOptions()")
    list("<Key>Escape" ""          "cancelEnterFun()")
    list("<Key>F"      "hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)")
    list("<Key>F1"     "hiHelp('window deGetAppInfo(deGetEditViewType(hiGetCurrentWindow()
    list("<Key>F11"    "deToggleAssistants()")
    list("<Key>F2"     "geSave()")
    list("<Key>F3"     "geSaveAs()"            "hiToggleEnterForm()")
    list("<Key>F4"     "geTogglePartialSelect()")
    list("<Key>F5"     "deFileOpen()")
    list("<Key>F6"     "leToggleMaintainConnections()")
    list("<Key>F7"     "leHiEditDRDRuleOptions()")
```

16

© 2020 All Rights Reserved.

Loading Bindkeys

To load a file of bindkey definitions every time you run Layout XL,

➤ Add the following line to your `.cdsinit` file.

```
load(prependInstallPath("path_to_bindkey_file"))
```

For example, the following line loads the list of default bindkeys for Layout XL.

```
load(prependInstallPath("samples/local/lxBindKeys.il"))
```

Setting Defaults

Before you can start working in the layout editor, several startup files must be initiated. Some of the things these files do include setting up your environment, pointing to libraries, and defining your plotters.

Startup Files

The layout editor uses the following startup files:

File Name	Refer To
.Xdefaults	Modifying the .Xdefaults or Equivalent File
.cdsinit	Specifying Environment Settings
.cdsenv	Specifying Environment Settings
.cdsplotinit	Plotting Your Design
display.drf	Display Resource File
cds.lib	Cadence Library Definition File

Setting Default Values with Menu Commands

The following layout editor and DFII commands and forms control most of the default values for your design environment:

Form Name	Description
Display Options Form	Controls display options in layout windows.

© 2020 All Rights Reserved.

Layout Editor Options Form	Controls the options for layout editor behavior.
User Preferences form	Controls your Cadence software environment. These settings affect all Cadence applications, not just the layout editor.
Display Resource Editor (DRE)	Controls the appearance of layers.
Edit Layers form	The *Edit Layers* command is no longer supported.

Setting the Default Application

You can automate the launching of an application when a cellview is opened rather than choosing the application from *Launch* menu each time.

For information, see Specifying Default Application in .cdsenv. and Setting the Default Application for a Layout Cellview.

Setting the Background Color of the Layout Editor Window

You can change the background color of the layout editor window using the environment variable, `defaultEditorBackgroundColor`.

You can change the background color of the current window using the following command:

```
hiGetCurrentWindow()->backgroundColor = "red"
```

```
"red"
```

Selecting the Workspace

You can rearrange the canvas, assistants, and toolbars to suit your individual work preferences. A customized configuration of the toolbars and assistants makes a *workspace*.

Tip

This section describes only the default workspaces defined for Layout XL. For detailed information on workspaces and how you define them, see Getting Started with Workspaces and Working with Workspaces in the *Virtuoso Design Environment User Guide.*

Layout XL ships with the following default workspaces:

© 2020 All Rights Reserved.

- Basic
- CAS
- Classic
- Constraints
- EAD
- Floorplan
- Modgen
- Multipatterning
- VSR

© 2020 All Rights Reserved.

Basic

The *Basic* workspace, shows the canvas, the selected toolbars, the Palette, Navigator, and Property Editor assistants. Notice that the Navigator and the Property Editor assistants are available as tabs on the left panel.

© 2020 All Rights Reserved.

CAS

The *CAS* workspace, shows the CAS tab in the Annotation Browser assistant, in addition to displaying the canvas and the selected toolbars.

© 2020
All Rights Reserved.

Classic

The *Classic* workspace (the default), shows the Palette assistant in addition to the canvas and the selected toolbars.

© 2020
All Rights Reserved.

Constraints

The *Constraints* workspace, shows the canvas, the selected toolbars, and the Palette, Navigator, Property Editor, and the Constraint Manager assistants. Notice that the Navigator and the Property Editor assistants are available as tabs in the left panel.

© 2020 All Rights Reserved.

EAD

The *EAD* workspace, shows the canvas, the selected toolbars, and the EAD Browser, Navigator, and the Palette assistants.

© 2020 All Rights Reserved.

Floorplan

The *Floorplan* workspace, shows the canvas, the toolbars, and the Navigator, Property Editor, and the Annotation Browser assistants.

© 2020 All Rights Reserved.

Modgen

The *Modgen* workspace, shows the canvas, the toolbars, and the Navigator, Property Editor, and the Constraint Manager assistants.

© 2020 All Rights Reserved.

Multipatterning

The *Multipatterning* workspace, shows the canvas, the toolbars, the Palette assistant, and the Track Pattern assistant.

© 2020
All Rights Reserved.

VSR

The *VSR* workspace, shows the canvas, the toolbars, and the Navigator, Palette assistant, and the Virtuoso Space-Based Router toolbar.

© 2020 All Rights Reserved.

Using Environment Variables

For information about how to set environment variables and a list of graphic and layout editor variables, see Environment Variables.

Setting Layout Editor Options

To open the *Layer Editor Options* form, choose *Options – Editor*. You can set the options in this form to define your layout editing environment settings. For information about various options in this form, see:

- Editor Controls
- Gravity Controls
- Tap
- Snapping
- Wire Editing
- Create Via

Display and Selection Controls

This section describes how to customize the display settings for the layout canvas and explains the selection and cross-selection mechanisms in Layout XL.

Setting Display Colors

You can change the colors and characteristics of the display layers that Layout XL uses for the following functionality.

- The Annotation Browser assistant uses the `y0` through `y9` layers to show flight lines.

  For more information, see Highlighting Incomplete Nets.

- The *XL Probe* command uses the (`hilite drawing`) through (`hilite drawing9`) layers to probe pins, nets, and devices.

  For more information, see Highlighting Probes.

- The *Generate Selected From Source* command uses the (`hilite drawing1`) layer to highlight schematic symbols.

© 2020 All Rights Reserved.

- The *Check Against Source* command uses the (`hilite drawing9`) entry layer to identify missing devices.
- The *Highlight* command for placement constraints uses the (`annotate drawing3`) entry layer to highlight in the layout the components bound by a particular placement constraint.

Editing Entry Layers

Tip

Although you can edit layers from the Palette assistant, it is not recommended.

To edit entry layers,

1. In the CIW, choose *Tools – Display Resource Manager*.
2. In the Display Resources Tool Box, click *Edit*.

   The Display Resource Editor window appears.

   For more information, see Display Resource Editor (DRE) Form in the *Virtuoso Technology Data User Guide*.

© 2020 All Rights Reserved.

3. Select *All to* see all the available layers.

Select *All*

4. In the *Layers* column, choose the name of the layer to change.

5. Turn on the *Fill Style*, *Fill Color*, *Outline Color*, *Stipple*, and *Line Style* options you want to use for that layer.

6. Click *Apply*.

The next time you use a command that displays the layer you changed, the color or pattern changes in the design window.

© 2020
All Rights Reserved.

Highlighting Incomplete Nets

To highlight incomplete connections with flight lines, Layout XL cycles through the `y0` through `y9` layers.

Layout XL displays the flight lines of each net in a different color. If there are more than 10 flight lines, the cycle is repeated unless you have specified certain layers for specific nets.

When assigning new colors to the flight lines:

- Use colors that do not closely resemble each other or the colors you use for paths and components.
- Avoid solid fill so you can see what is underneath.

Highlighting Probes

Layout XL uses the (`hilite drawing`) through (`hilite drawing9`) entry layers for probes, which identify equivalent design elements on the schematic and the layout.

The *Check Against Source* command uses the (`hilite drawing9`) layer to identify components in the schematic that are not in the layout (and vice versa).

If you change the colors of the (`hilite drawing`) layers,

© 2020 All Rights Reserved.

- Use thick lines so that you can see device pins.
- Use bright colors to make probes and highlights easy to find.

Tip

Do not make the (`hilite drawing`) layer and the (`hilite drawing2`) layer solid fill because these layers are used to indicate selected components and to manipulate shapes.

Selecting Objects in Layout XL

You can select objects either before or after you start a command.

Preselect Mode

If you select objects before starting a command, the command is said to be operating in *preselect mode*. In preselect mode, the command operates only on the objects that were selected when the command was started. When the command is completed, the selected objects remain in the selected set.

In preselect mode:

- Editing commands do not automatically repeat, even if the *Repeat Commands* option is switched on in the Layout Editor Options form.
- Some editing commands prompt you for a starting point, called the reference point, for the edit.

Postselect Mode

If you select the command and then the objects, the command is said to be operating in *postselect mode*. In postselect mode, you make your selection within the command and when the command is completed, nothing is selected.

In postselect mode:

- Editing commands automatically repeat if *Repeat Commands* is on (the default).
- Editing commands do not necessarily prompt you for a reference point. If you click to select the first object, the editor uses that point as the reference point.

© 2020 All Rights Reserved.

Deselecting Objects in Layout XL

- To deselect an object:

➡ Invoke the *(De)Select Under Cursor* context menu and from the De(select) submenu, hover over the object you want to deselect. The object gets highlighted.

For more information on object selection, see Editing Objects in the *Virtuoso Layout Suite L User Guide*.

Viewing Multiple Cellviews in Layout XL

You can open more than one pair of schematic-layout windows in Layout XL and work on each pair independently.

However, when you open multiple cellviews in Layout XL, the second and subsequent sets of windows are not automatically configured in the same way as the first set.

- If you open a schematic and two copies of the same layout, the *XL Probe* command and the commands in the *Check* and *Update* submenus apply to both layouts.
- If you open a schematic and two different layouts, the *XL Probe* command and the commands in the *Check* and *Update* submenus apply only to the layout from which you selected the command.

Displaying Options Forms Automatically

Options forms let you change command settings while you are running commands. Many options forms do not appear automatically if you have *Options Displayed When Commands Start* turned off in the *Options - User Preferences* in the CIW. In this situation, you must press `F3` to see the command forms.

Printing to the Command Interpreter Window

To print process information to the CIW instead of to a separate Layout XL Info window,

➤ Add the following line to your `.cdsenv` file or type it in the CIW

```
envSetVal("layoutXL" "infoWindow" 'boolean nil)
```


© 2020 All Rights Reserved.

Using the Technology File

Each design library must be associated with a technology library compiled from a technology file. The technology file defines

- Design layers and their properties
- Constraint and constraint groups
- Via definitions
- Site definitions
- Technology devices

For complete details about how to create and edit technology files, refer to the following:

- *Virtuoso Technology Data ASCII Files Reference*
- *Virtuoso Technology Data User Guide*

The layout editor requires the technology file to contain complete definitions of the layers used in your design.

Layout Editor Strokes

A stroke is a unique shape you draw in a cellview by pressing and holding the right mouse button. Cadence ships a set of predefined strokes for the layout editor that you can load.

Note: Before you can use strokes, you must load them from the Cadence installation hierarchy.

Loading and Unloading Strokes

To use the predefined layout editor strokes, you must first load them.

To load the strokes,

➤ Type the following in the text line of the CIW:

```
load(prependInstallPath( "etc/sted/stroke.il"))
load(prependInstallPath( "etc/sted/defstrokes.il"))
hiLoadStrokeFile("def.strokes" "Layout")
```

where, `prependInstallPath` is a SKILL function that uses the value set for the UNIX environment variable `CDS_INST_DIR` (it usually points to *`install_dir`*`/tools/dfII`).

© 2020 All Rights Reserved.

The `stroke.il` file defines mouse key bindings and the applications that recognize strokes. The `defstrokes.il` file lists the Cadence® SKILL language functions used by the predefined strokes. The `def.strokes` file defines the stroke shapes.

You can also add these commands to your `.cdsinit` file, so the strokes are loaded whenever you start the Cadence software.

Note: After you load strokes, you cannot use the right mouse button for any other functions.

To unload the strokes,

1. Quit the current session of the Cadence software and restart the software.
2. Remove the load strokes commands from your `.cdsinit` file, if you have added them to that file.

Drawing Strokes

To start a command with a stroke, press right and draw one of the following strokes:

© 2020 All Rights Reserved.

Stroke	Drawing Direction	Command	Stroke	Drawing Direction	Command
		Copy			Zoom In
		Delete			Zoom In
		Window Fit			Zoom Out
		Move			Pan Right
		Properties			Pan Left
		Undo			Pan Up
		Stretch			Pan Down

The arrows show the direction in which you draw the strokes. There are two ways to start *Zoom In* with strokes.

© 2020 All Rights Reserved.

Using Forms

A form is a window that appears when you use a command.

Displaying Forms

There are two ways to display forms, depending on whether the command has a standard form or an options form.

- Standard forms

  These forms have three dots (...) following the command and appear automatically.

- Option forms

  These forms do not appear automatically. You can display an options form by double-clicking right mouse button or pressing `F3` when a command is active.

To set options forms so they display automatically whenever you choose a command,

➤ From the CIW, choose *Options – User Preferences* and turn on the *Options Displayed When Commands Start* button.

Using Form Buttons

The buttons on a standard form work as follows:

Button	Description
OK	Completes the command and closes the form.
Close/Cancel	Closes the form without executing the command.
Defaults	Resets default values for options on the form.
Apply	Completes the command and keeps the command active and the form on the screen.
Help	Displays information about the form fields.

The buttons on an options form work as follows:

Button	Description
Hide	Minimizes the form and lets you continue with the command.

© 2020 All Rights Reserved.

Cancel	Closes the form and stops the command.
Defaults	Resets default values, if any, for options on the form.

Note: Many forms do not have a *Defaults* button because there are no appropriate default settings for that command. Some commands may have more or fewer buttons than those shown here.

Using Context-Sensitive Menus

Context-sensitive menus display context-sensitive options, based on the object under the pointer at the time of right-click. Context-sensitive menus provide a quick access to object-specific commands.

For example, the layout editor context-sensitive menu lets you start a few commonly used layout commands. Right-click at any free space in the design canvas to display the *Layout* context-sensitive menu and select the required command.

If you use *Edit In Place*, *Descend Edit*, or *Descend Read* commands, the *Layout* context-sensitive menu also displays the *Return* command to ascend the hierarchy.

© 2020 All Rights Reserved.

If multiple objects of different types are selected when you right-click, the *Multiple* context-sensitive menu appears, which allows you to perform basic editing operations on all the objects selected.

However, if all selected objects are of same type, the object-specific, context-sensitive menu appears. For example, if you select a group of rectangles and right-click, the *Shape* context-sensitive menu appears.

Performing Fast GDS Export Using XStream Out

To perform fast GDS export using XStream Out in the layout editor window,

- Choose *File - Export Stream From VM*

© 2020 All Rights Reserved.

Selecting this option will let you export the currently open design using XStream Out. For more information see *Design Data Translator's Reference* guide.

Using the Allegro PCB/Package Menu

In Virtuoso Layout Suite L, the Allegro PCB/Package menu is available that enables you to work on System-in-Package (SIP).

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

2

Viewing the Design

- Zooming In or Out
- Using the Magnifier
- Using Dynamic Zoom
- Panning to a Point
- Panning Across the Cellview
- Redrawing the Display
- Setting an Area Display
- Viewing Coordinates
- Viewing Angles
- Viewing Selected Object Information
- Saving or Restoring a View
- Displaying a Background Cellview
- Reading Status Information
- Creating a Reference Window
- Using the World View Window
- Setting Display Options
- Displaying Object Information

© 2020 All Rights Reserved.

Zooming In or Out

Use the *View – Zoom* commands to control the magnification of the cellview.

- *Zoom In*: Magnifies the image in the cellview window by a factor of 2. You can also scroll up the mouse scroll wheel to zoom in.
- *Zoom Out*: Reduces the image in the cellview window by a factor of 2. You can also scroll down the mouse scroll wheel to zoom out.
- *Zoom To Area*: Prompts you to create a box and then magnifies the image defined by the box to fit in the cellview window.
- *Zoom To Grid*: Reduces the image to the smallest magnification at which the grid is visible.
- *Zoom To Selected*: Increases the image to the largest magnification at which the selected objects can be viewed in the cellview window.
- *Zoom To Fit All*: Redraws the window so that all objects in the cellview appear in the window. This saves time after you zoom or pan and want to see the entire cellview again.
- *Zoom To Fit Edit*: Places the cell you are editing in the center during edit-in-place. If you are not editing in place, *Zoom To Fit Edit* has the same result as *Zoom To Fit All.*

By using the mouse, you can create a box to zoom in on a specific area or zoom out to a specific size.

To zoom in by using the mouse:

➤ Click right, and drag the pointer to create a box around the area you want to enlarge.

To zoom out by using the mouse:

➤ Press `Shift`, click right, and drag the pointer to create a box into which you want the reduced image to fit.

Using the Magnifier

You can use the magnifier to magnify (zoom in) a specific part of the design display area instead of zooming in the entire design display area. You can perform the creating and editing operations in the magnified region.

To use the magnifier:

1. Choose *View – Magnifier* or use the dot bindkey (.) to toggle the magnifier on/off.

© 2020 All Rights Reserved.

A square magnifier appears. It is anchored to the pointer position. You can customize the position of the magnifier with respect to the pointer by using the Magnifier Options Form (*Options – Magnifier* or use the bindkey `'`).

Note: *Magnifier* must be selected from the *View* menu before you can use the *Magnifier Options* form.

2. Place the pointer over the area in the design that you want to magnify.

   The magnifier appears as a square box over the design display area and shows a magnified view of the design under and around the pointer, as shown below.

3. You can move the pointer around to magnify different regions of the design. The magnifier moves along with the pointer, based on the anchorage defined for the magnifier.

4. Press `.` to switch off the magnifier.

You can use the following bindkeys while using the magnifier:

- Press `#` to fix the magnifier position at one place in the design display area.
- Press `Shift` + `Ctrl` + Up arrow key (or press `Shift` + `Ctrl` and scroll the mouse scroll wheel up) to scale up the magnification by a factor of 1.25 in the magnifier.
- Press `Shift` + `Ctrl` + Down arrow key (or press `Shift` + `Ctrl` and scroll the mouse scroll wheel down) to scale down the magnification by a factor of 0.80 in the magnifier.

For information about the SKILL APIs that can be used for design magnification at any time during a `virtuoso` session, see Magnifier Functions in the *Cadence User Interface SKILL Reference*.

© 2020 All Rights Reserved.

Using Dynamic Zoom

You can use the dynamic zoom feature to automatically pan or zoom in the layout canvas based on the objects selected in the Navigator assistant. This lets you more locate objects in a busy design.

To enable the dynamic zoom feature, select the *Dynamic Zoom* option in the *View* menu.

This feature has three modes, *Pan To Selected*, *Zoom To Selected*, or *Zoom To Fit*. To set the mode:

1. Select *Options - Display*.

   The Display Options form opens.

2. In the Zoom/Pan Controls section, ensure that the *Dynamic Zoom* option is set to *on*.
3. Choose the dynamic zoom *Mode* that you want to use, *Pan to Selected, Zoom To Selected*, or *Zoom To Fit*.
4. Move the slider to specify the *Zoom Scale (%)*.
5. Click *OK* to apply the Zoom / Pan Controls updates.

Panning to a Point

To center the image in the cellview window around a point you enter, choose *View – Pan* [`Tab`] and click the point you want to center. The image moves so the point you chose is in the center of the window. Alternatively, you can specify the point around which you want to centre the image in the cellview window using the *Enter Points* form. For more information, see Enter Points Form.

© 2020 All Rights Reserved.

Panning Across the Cellview

The arrow keys on the keypad let you pan across the cellview in any direction.

Redrawing the Display

After you have made many edits to a cellview, the design area can sometimes show portions of lines or objects that you have moved or deleted. Choose *View – Redraw* to refresh the cellview display.

Setting an Area Display

To set the display detail for an area,

1. Choose *View – Area Display – Set* and set the *From* and *To* display levels for the area.
2. Create a box in the cellview around the area in which you want to see the detail of lower levels of the design.

To remove one detail area at a time,

➤ Choose *View – Area Display – Delete* and click the detail area you want to remove.

To remove all detail areas,

➤ Choose *View – Area Display – Delete All.*

© 2020 All Rights Reserved.

Viewing Coordinates

The *Show Coordinates* command displays the coordinates of the selected shape. To view the coordinates of a shape, select it fully or partially. You can select a shape either before (pre-selection mode) or after (post-selection mode) you run the *Show Coordinates* command. For more information about pre-selection and post-selection modes, see Pre-Selection and Post-Selection of Objects.

To view the coordinates of a shape:

1. Choose *View – Show Coordinates.*
2. Select a rectangle as explained below:
   a. Point at or select the complete shape: The coordinate information is displayed for the lower-left and top-right vertices of the rectangle, as shown in the figure below. Notice the diamond-shaped markers identifying the coordinate locations.

   b. Point at or select a vertex: The coordinate information is displayed at the selected vertex of the rectangle, as shown in the figure below.

© 2020 All Rights Reserved.

c. Point at or select an edge: The coordinate information is displayed at the two vertices of the selected edge of the rectangle, as shown in the figure below.

To view the coordinates of all the vertices of a selected shape:

1. Select a shape, say a polygon.

   The coordinate information is displayed for one of the vertices, as shown in the figure below.

2. Press `Spacebar` to display the coordinates of the next vertex.

   You can continue to press `Spacebar` to view the coordinates of all the vertices in succession.

   Note: If you select a shape partially, only the coordinates of the selected vertices are displayed.

To view the coordinates of the vertices when multiple shapes are selected:

© 2020 All Rights Reserved.

1. Select multiple shapes, say a rectangle, a path, and the edge of a polygon.

   The coordinates are displayed for each shape in the selected set one by one, for example, for the selected edge of the polygon, as shown in the figure below.

2. Press `Spacebar` again to view the next set of coordinates—lower-left and upper-right vertices of the rectangle, as shown.

© 2020 All Rights Reserved.

Continue to press `Spacebar` to view the remaining sets of coordinates.

Note: If you first run the *Show Coordinates* command and then select the required shapes, you can repeatedly cycle over the coordinates by pressing `Spacebar`. On the other hand, if you first select the required shapes and then run the *Show Coordinates* command, the command exits after cycling through all the coordinates once.

3. Press `Esc` to finish the command.

© 2020 All Rights Reserved.

The following table shows examples of coordinates displayed for various shapes:

Path		Ellipse	
Circle		Donut	

Circles and ellipses have two sets of coordinates, one for the bounding box and one for the center point.

Rectangles, paths, labels, and vias too have only one set of coordinates. Selecting a label, or via fully or partially displays the same set of coordinates, that of their respective origins. Selecting an instance displays the lower left and upper right coordinates of the bbox and after you press the spacebar the origin is displayed.

© 2020 All Rights Reserved.

A donut has three sets of coordinates—one each for its two bounding boxes and one for the center point—as shown in the figures below.

Outer Bounding Box Coordinates

Inner Bounding Box Coordinates

Center Point Coordinates

You can control the font size and color of the coordinates displayed by using the *Dynamic Measurement* tab of the Dynamic Display Form.

Viewing Angles

The *Show Angles* command displays the angle enclosed at the intersection of two edges. The command displays both the interior and exterior angles. You can view the angles of a shape by selecting it fully or partially. The *Show Angles* command supports both pre-selection and post-selection modes.

Note: The *Show Angles* command does not support paths or pathsegs with two end points and paths or pathsegs with only one of the end points selected.

© 2020 All Rights Reserved.

To view the angle measurement between the edges of a shape:

1. Choose *View – Show Angles.*
2. Select a shape, say a rectangle.

   The interior and exterior angle measurements appear, as shown in the figure below. Notice that the angles are marked with arcs.

To view the angle measurements at all the vertices of a selected shape:

1. Select a shape, say a polygon.
2. Press `Spacebar` to view the angle at the next vertex.

© 2020 All Rights Reserved.

Continue to press `Spacebar` to view the angles at all the vertices of the polygon in succession.

Note: If you select a shape partially, only the angles at the selected vertices are can be viewed.

To view the angle measurements at the vertices of multiple selected shapes:

1. Select multiple shapes, say a rectangle, a path, and the edge of a polygon.

© 2020 All Rights Reserved.

The angle measurements are displayed at one of the vertices of the selected edge of the polygon, as shown in the figure below.

2. Continue to press `Spacebar` to view the angle measurements for the remaining vertices, that is, all the vertices of the rectangle.

© 2020 All Rights Reserved.

Because the *Show Angles* command does not support paths, angle measurements are not displayed for the path.

© 2020
All Rights Reserved.

Note: If you first run the *Show Angles* command and then select the required shapes, you can repeatedly cycle through the angles by pressing `Spacebar`. On the other hand, if you first select the required shapes and then run the *Show Angles* command, the command exits after cycling through all the angles once.

3. Press `Esc` to finish the command.

Viewing Selected Object Information

To display information about a selected set of objects, choose *View – Show Selected Set.*

The Show Selected Set Form opens.

The form shows:

- The *Total Shapes Selected* section lists the selected objects, their layers, the number of points used to create them, and the coordinates of the bounding box that surrounds each object.
- The *Total Instances Selected* section lists all selected cell instances, including each instance name, placement orientation, and the coordinates of its origin.
- The *Total Mosaics Selected* show information about arrays of cell instances are selected.
- The *Total Boundaries Selected* section lists the selected boundaries, their types, the number of edges used to create them, the area, and the boundary names.
- The *Total Blockages Selected* section lists the selected blockages, their types, their layers, their owners, the number of points used to create them, and their areas.

© 2020 All Rights Reserved.

- The *Total Vias Selected* section lists the selected vias, their types (standard or custom), their names, their nets, their directions, and in the case of custom vias their library, cell and view names.

Saving or Restoring a View

The layout editor keeps the last three images you viewed in memory. You can move forward and back through these previously zoomed or panned images when you need to.

To return to the previous image,

➤ Choose *View – Save/Restore – Previous View*.

To go forward to the next image,

➤ Choose *View – Save/Restore – Next View*.

Saving a Zoom or Pan Image

The Save View Form saves the image shown in the window to a file that lasts for the current editing session.

To save a zoom or pan image,

➤ Choose *View – Save/Restore – Save View* and type the name you want to assign the file.

Restoring a Zoom or Pan Image

After you have saved the image, you can later restore it with the *Restore View* command.

To restore a saved image,

➤ Choose *View – Save/Restore – Restore View* and click the name of the file you want to restore.

The image in the window is replaced by the image you saved.

Note: Images are saved only during the current editing session. They are deleted when you exit the software.

© 2020 All Rights Reserved.

Displaying a Background Cellview

The *View – Background* command allows you to display two superimposed cellviews in the same window. Any cellview from any library can be used as a background cellview, except the cellview you are currently editing. You can display layout, abstract, and schematic cellviews in the same window, or you can use a layout cellview as a template to guide your design.

You can open only one background cellview per displayed cellview. The background cellview is visible when you are editing in place. It is not visible when you are descending, but becomes visible when you return to the original hierarchical level.

All display options set in the *Display Options* window are obeyed by both the foreground cellview and the background cellview. The background cellview can be hierarchical.

The cellview you superimpose as a background cellview is purely graphical. It has no influence on any application commands or functions.

- You cannot edit, select, query, or move any objects contained in the background cellview.
- The background cellview has no effect on functions such as pointer gravity movement.
- The *View - Fit* command does not take into account the background cellview.

The background cellview retains all the settings of the Palette Assistant (visibility, layer packet information, etc.) that is attached to the technology file of the background cellview. For example, if the background cellview contains objects on metal2, the objects on metal2 will not display if the visibility of metal2 is turned off in the Palette assistants of the background cellview. If the background cellview has a different technology file from your foreground cellview, the displayed Palette assistants will not control any of the display characteristics of the background cellview.

If the background cellview is edited in another window, you will need to redraw to see the changes made to the background cellview in the foreground cellview.

To display a background cellview,

1. From an open cellview window choose *View – Background*.

   The Background Cellview Form appears.

2. Specify the library, cell, and view name of the cell you want to use as the background cellview.

   Your current edited cellview cannot be used as a background in your open window.

3. Specify the X and Y coordinates where you want the cellview to be placed in the window.

© 2020 All Rights Reserved.

4. Select the rotational angle you want to apply to the background cellview.
5. Specify whether or not you want the background cellview to be displayed or displayed with a reduced luminosity.
6. Click *OK*.

   The background cellview appears behind your opened cellview.

Reading Status Information

The Status Banner at the bottom and the Status Toolbar at the top of the design window provide dynamic information such as pointer movement, object coordinates, and the command currently active.

Status Toolbar

Status Banner

© 2020 All Rights Reserved.

Status Banner

The Status Banner displays by default in a design window. You can select the information you want to view in the Status Banner and change the order in which the items appear by using the bannerFields environment variable.

Status Toolbar

You can enable or disable the Status Toolbar. View the Status Toolbar using one of these methods:

- Select *Window – Toolbars – Status Toolbar.*
- Right-click anywhere in the main toolbar area and select *Status Toolbar.*

The status toolbar displays information related to the objects selected on the canvas.

The first section of the status toolbar displays the following information:

- Selection mode (full select (`F`), full select with spine (`FS`), partial select (`P`), or partial select with spine (`PS`)) and the number of figures selected. For example, *(F)Select:2* signifies two figures are selected in the full selection mode.
- The number of nets selected (*Sel(N):n*)
- The number of instances selected (*Sel(I):n*)
- The number of objects selected (*Sel(O):n*)

The second section of the status toolbar displays the editable toolbar. When you start a command and press the `Tab` key, the *X* and *Y* fields are highlighted and the *dX* and *dY* fields are set to 0, as shown below:

You can use the environment variable to setDxDyToZero to set the *dX* and *dY* fields to 0 on the status toolbar.

You can edit the *X* and *Y* fields. When you press `Enter`, the *X* and *Y* coordinates are used by the currently active command.

Note: When you click the *X* and *Y* fields on the status toolbar with no active command, and press `Enter`, the pointer pans to the coordinates specified in the *X* and *Y* fields.

© 2020 All Rights Reserved.

The *dX* and *dY* fields get highlighted if you press the `Tab` key again. Similar to the *X* and *Y* fields, you can edit the *dX* and *dY* fields. When you press `Enter`, the *dX* and *dY* coordinates are used by the currently active command.

You can control the width of the *X*, *Y*, *dX*, and *dY* fields on the status toolbar by using the environment variable, statusBarXYdXdYLength.

The *X* and *Y* fields are coupled. If you set one, the other is automatically set. The same is true for *dX* and *dY* fields.

The *Dist* field displays the distance the pointer has moved and the *Cmd* field displays the currently active command.

For `Move`, `Copy`, and `Stretch` commands, when you do not specify the values in the *X* and Y fields and press `Enter`:

- If the selection set has a single object with origin, the origin will be considered as the reference point.
- If the selection set has a single object without origin, the lower left of the objects bounding box will considered as the reference point.
- If the selection set has multiple objects, the lower left of the selection sets bounding box will considered as the reference point.

Note: As the data displayed by the Status Toolbar is dynamic, Cadence recommends that you dock the toolbar only horizontally, either to the top or bottom of the design window.

Using the *Window – Workspaces – Save As* command, you can save a configuration to display the Status Toolbar every time the cellview window is opened.

Customizing Status Toolbar Information

You can customize the information that is displayed on the status toolbar using the environment variable, `statusToolbarFields`. You can display fields, such as, cellsize, number of selected objects, environment variable values, shell environment variable values, and window variable values.

The following table lists the information that can be displayed on the status toolbar and an example of the syntax to be used for the environment variable, `statusToolbarFields`.

Display Information	Syntax
Predefined – X, Y, dX, dY	`layout statusToolbarFields string "(X) (Y)"`

© 2020 All Rights Reserved.

Display Information	Syntax
CDS Environment Variables	`layout statusToolbarFields string "(E \"layout\" \"snapMode\")"`
Shell Variables	`layout statusToolbarFields string "(SH \"Process\")"`
SKILL Variables	`layout statusToolbarFields string "(S SomeVersion)"`
Text Strings (fixed)	`layout statusToolbarFields string "(T \"gpdk45\" \"Process\")"`
Separator	`layout statusToolbarFields string "(X) (Y) \\\| (dX) (dY)"`

You can also customize the display format of the information displayed on the status toolbar. You can specify information in bold font and change the color of some information on the toolbar. You can also use an expression format that allows conditional formatting based on values. For example, the information can be displayed in bold font if a command is active or if the number of selections is more than zero, you can customize the status toolbar to display this information in red and bold font.

Creating a Reference Window

You can display a small copy of the whole cellview. This is often called a reference window because you use it to see your whole layout while you edit a zoomed-in portion.

To create a reference window,

1. Choose *Window – Copy Window*.

   A reference window appears, displaying the same cellview.

2. To shrink the new window, click a corner of the new window and drag it towards the opposite corner.

3. In the new window, choose *View – Fit All* [`f`].

© 2020 All Rights Reserved.

Using a Reference Window to Zoom or Pan

When you have two windows displaying the same cellview, you can start a *Zoom* or *Pan* command in your original window and then enter the points to zoom or pan in the reference window. This way, you can pinpoint the area you want to display using the reference window.

To zoom in a reference window,

1. With the pointer in the original window, choose *Zoom – In*.
2. Move the pointer into the reference window.
3. Click to create the zoom box in the reference window.

   The original window zooms in to the area you designated in the reference window, but the reference window remains unchanged.

Using a Reference Window to Create or Edit Objects

When you have two windows displaying the same cellview, you can start an editing command in one and finish it in the other window. For example,

1. In the original window, choose *Create – Path*.
2. Click to begin creating the path.
3. Move the pointer into the reference window, and double-click to finish the path.

Using the World View Window

The World View window shows the complete layout design, its relationship to the main window, and marks the part of the design that is currently displayed in the main graphics window with a view box.

To open the World View window, choose *Window – Assistants – World View*.

- Zooming In or Out

  The view box has a black bar on each side which is a handle used to zoom in or out. when you hold the left mouse button down on a handle of the view box and drag inward or outward, the view box increases or decreases in size and the cellview zooms in or out corresponding to the size of the view box.

- Moving the View Box

© 2020 All Rights Reserved.

To move the view box to a specific area, click the desired area of the design in the World View window with the left mouse button. You can also click the view box and drag it to a new location.

If you use the middle mouse button to click the design in the World View window, a context-sensitive menu opens.

By using the context-sensitive menu, you can:

- Redraw the window
- Zoom In
- Zoom Out
- Fit the entire design into the World View window by choosing *Fit All*
- Select the percentage of the color luminosity of the view box by choosing *Pattern*.

Setting Display Options

The *Display Controls* lets you choose how objects are displayed.

Setting Up the Design Objects to Appear

To control which design objects or attributes appear,

1. Choose *Options – Display*.

   The Display Options Form opens.

2. Click any of the display controls.
3. Click *Apply* to see the result of your changes.

© 2020 All Rights Reserved.

4. When you are satisfied with the settings, click *OK*.

For a description of the options available in the form, see Display Options Form.

Displaying Instance or Master Cell Names

When you suppress cell instance detail (press `Ctrl + F`), one of the following is displayed for each instance:

- The master cell name
- The instance name (usually an incremented number)
- Both master and instance name

To choose what name you want displayed,

1. Choose *Options – Display*.

   The Display Options Form opens.

2. Set *Show Name Of* to *instance* or *master* or *both*.
3. Click *OK*.

Displaying Details or Outlines of Arrays

You can adjust the display details of cell arrays.

To set the array detail,

1. Choose *Options – Display*.
2. Set the *Array Icons*, *Array Display*, and *Display Levels* fields as desired.
3. Click *OK*.

The following table lists the possible options and the results obtained.

Display Option	Results
Array Icons	When *Array Icons* is on and *Display Levels* is set to *From 0 To 0*, outlines of array elements appear with no detail.
Array Display	*Array Display* settings control the number of elements in the array that appear.

© 2020 All Rights Reserved.

Display Option	Results
Display Levels	*Display Levels* controls the level of instance detail to display.

The following figures show sample settings for the display of arrays.

Array Icons: off
Array Display: Full
Display Levels: 0-32

Array Icons: on
Array Display: Full
Display Levels: 0-0

Array Icons: off
Array Display: Border
Display Levels: 0-32

Array Icons: on
Array Display: Border
Display Levels: 0-0

Array Icons: off
Array Display: Source
Display Levels: 0-32

Setting Display Levels

You can control whether details inside of cell instances placed in a cellview appear.

1. Choose *Options – Display*.

   The Display Options Form opens.

2. Set the display levels.

© 2020 All Rights Reserved.

From shows the lowest level of hierarchy that appears. By default, the display begins at the current level, 0.

To shows the highest level of hierarchy that appears. Cell instances inside this cellview are level 1, instances inside level 1 are level 2, and so on.

3. Click *OK.*

To set the display level to 0, press `Control-f` in the design window. To set the display level to 32, press `Shift-f` in the design window.

Setting Up Grid Controls

By default, the cellview window shows a grid of dots. There are two different grids: the minor (small) grid and the major (large) grid.

- Minor grid points are white and appear at every micron.
- Major grid points are green by default and appear at every 5 microns.

To change the visible grid,

1. Choose *Options – Display*.

   The Display Options Form opens.

2. Set the grid controls to the settings you want.
3. Click *OK*.

Setting Up Snap Grid

The grid defines the points at which the pointer (the small square) snaps to the cellview or to objects. With the snap grid set to 0.5 microns, the pointer can snap to each visible grid point or halfway between each grid point.

For example, if your user units are microns, and you want to draw objects at 0.5 micron intervals, you set the spacing for the snap grid to 0.5 microns.

To set the grid, you set the *X Snap Spacing* (distance the pointer can move along the X axis) and the *Y Snap Spacing* (distance the pointer can move along the Y axis) fields. Setting *X Snap Spacing* and *Y Snap Spacing* to 0 produces the same result as turning the environment variable `snapToGrid` off.

Gravity has precedence over snap grid. Turn *Gravity* off while using snap grid.

© 2020 All Rights Reserved.

Note: The snap grid is not the same as the visible grid in the cellview.

Setting Up Snap Grid Spacing

To set the snap grid spacing,

1. Choose *Options – Display*.

   The Display Options Form opens.

2. Type the smallest distance you want between the points you enter along the X axis (*X Snap Spacing*) and Y axis (*Y Snap Spacing*).

   The value you type is in user units (usually microns).

3. Click *OK*.

Setting Up Snap Mode

Setting the Default Snap Modes

To set the default snap mode,

1. Choose *Options – Display*.

   The Display Options Form opens.

2. Set the snap modes.

   - *Create* controls how segments snap as you create objects. You can also access it on the *Options* toolbar using this icon .
   - *Edit* controls how segments snap as you move or stretch objects. You can also access it on the *Options* toolbar using this icon .

3. Click *OK*.

Snap Mode

The snap mode controls how the pointer locks to the grid as you create or edit objects.

© 2020 All Rights Reserved.

- The edit commands use the snap mode to control where you can move all or part of an object. The settings are shown below.

- The create commands use the snap mode to control the shape of segments as you create or reshape objects. *Snap Mode* settings are illustrated below:

The *Synchronized SnapMode* icon on the *Options* toolbar displays the snap mode of the create and edit commands. This icon provides a representation of both the create and edit snap modes. In addition, it provides a single way to change both the create and edit snap modes, simultaneously.

When both the create and edit snap modes are synchronized, the appropriate icon (anyAngle, diagonal, or orthogonal) displays on *Synchronized SnapMode* icon on the *Options* toolbar, as shown in the figure below.

© 2020 All Rights Reserved.

Synchronized Snap Mode icon

Important

You can cycle through the snap modes (anyAngle -> diagonal -> orthogonal) by pressing the `N` bindkey or clicking the *Synchronized SnapMode* icon on the *Options* toolbar.
The *Create* and *Edit* snap modes become unsynchronized if you change either the *Create* snap mode or the *Edit* snap mode in the Display Options Form or on the *Options* toolbar. When the *Create* and *Edit* snap modes are unsynchronized, then the *Synchronized SnapMode* icon on the *Options* toolbar is displayed as a dotted diagonal double-headed arrow, as shown in the figure below.

Setting Up Dimming Options

Dimming lets you change the color luminosity of objects in the cellview. The dimming feature lets you differentiate objects in editable context from objects in other contexts. For example, you can dim everything in the design but the objects in the edit cellview, to better view what is currently editable.

The options are set in the *Display Options* form.

Dimming settings are associated with the window environment.

© 2020 All Rights Reserved.

- Only one dimming intensity can be set per window.
- When you descend the hierarchy or edit-in-place, the dimming settings from the parent environment are used for the new environment.
- When returning from a descend or an edit-in-place, the dimming settings of the parent environment are restored.

To use the dimming options,

1. Select *Options – Display*.
2. Set the *Scope* cyclic field of the *Display Options* to one of the following:
   - *none*: No dimming is applied. This is the default setting. Even if you change the *Dim Intensity* from 0 percent to 100 percent, the objects remain undimmed. The only exception is that the background cellview will dim its *Display* field is set to *dimmed*.
   - *all*: The design is completely dimmed. If you select an object and set the intensity to 100, only the selection box is displayed.
   - *outside*: Upper and lower-level hierarchy and surrounding hierarchy objects dimmed outside of the current context help you distinguish editable objects from the objects that are in other contexts. With the *outside* setting on, if a Pcell exists at the edit level and is visible at the edit level, its contents are not dimmed. If you select the

© 2020 All Rights Reserved.

Via Shapes and *Instance Pins* options, vias belonging to the edit level and instance pins belonging to edit level-1 are not dimmed.

Display Stop Level = 32, Dimming Scope = outside

© 2020
All Rights Reserved.

- *eipSurround*: Only the objects surrounding the EIP hierarchy are dimmed.

Display Stop Level = 32, Dimming Scope = eipSurround

© 2020
All Rights Reserved.

3. Select *Automatic Dimming* to activate dimming irrespective of the dimming *Scope*.

Dimming Scope = none, Automatic Dimming = on

© 2020 All Rights Reserved.

Nothing is selected, highlighted, or probed. So nothing is dimmed.

Dimming Scope = none, Automatic Dimming = on

4. Select *True Color Selection only* to display the selected objects in their original colors and not display their selection highlight.

© 2020 All Rights Reserved.

5. Move the *Dim Intensity* slider to set the color luminosity of the dimmed objects.

Edit in Place with surrounding objects dimmed to 0.

Edit in Place with surrounding objects dimmed to 50.

© 2020 All Rights Reserved.

Edit in Place with surrounding objects dimmed to 100.

6. Click *Apply*.

Note: You can also enable dimming from the *Options* toolbar using this icon. Dim

Saving, Loading, and Deleting Display Settings

You can save, load, and delete the settings in the Display Options form to several locations. Saving the display settings lets people who work on the same design use the same display settings.

The display settings can be saved to the current cellview, library of the cellview, technology library of the cellview, or a specified file.

To save the display settings,

➤ Choose *Save To* and either *Cellview*, *Library*, *Tech Library*, or *File.*

To load the display settings, do one of the following,

■ Start the software.

The display settings are loaded in this precedence, from highest to lowest:

- ❑ Cellview
- ❑ Library of the cellview

© 2020 All Rights Reserved.

- Technology library of the cellview
- File (`~/.cdsenv`)

- Choose *Load From* and either *Cellview*, *Library*, *Tech Library*, or *File.*

To delete the display settings,

➤ Choose *Delete From* and either *Cellview*, *Library*, or *Tech Library*.

Displaying Object Information

- Controlling Information Balloons and Dynamic Measurements
- Customizing Object Display
- Using a Custom SKILL Function
- Saving the Display Information Configuration
- Loading the Display Information Configuration
- Saving a Configuration as Default
- Deleting a Display Information Configuration
- Displaying Preselect and Selected Object Information

There are two types of controls available to display textual information about objects:

- *Info Balloons*

Information balloons display textual information when the pointer is moved over an existing object or shape. They display information about bound and unbound instances. The information displayed for unbound instances when the stop display level ranges from

© 2020 All Rights Reserved.

`0` to `32` is the same as the information for bound instances when the stop display level is set to `0` in the Set Area View Level form.

Note: Info balloons display the information about members of the fig groups when you are editing a fig group or the *Transparent Group* option is selected in the Display Options form.

Note: (ICADVM20.1 Virtuoso RF Solution Only) You can view Information balloons for curvilinear geometries, such as curved paths and curved polygons.

- *Dynamic Measurement*

  Dynamic measurement displays distance measurements when creating or editing objects. The measurement text is displayed next to the pointer.

Note: (ICADVM20.1 Virtuoso RF Solution Only) Dynamic measurement is supported for curved shapes.

Controlling Information Balloons and Dynamic Measurements

1. To display object information, select *Options – Dynamic Display*.

© 2020 All Rights Reserved.

The Dynamic Display Form opens.

2. To display information balloons, set *Show Info Balloon* to on.
3. In the *Info Balloon* tab:
   a. Set the *Pause Time*

      The *Pause Time* is the amount of time before the information balloon will appear when the pointer is over an object. The range is from 0 to 10.

   b. Set *Fade Time* to the desired time in seconds.

      The *Fade Time* sets the amount of time (in seconds) for the information balloon to fade away after the balloon appears. The range is from .001 to 100.

© 2020 All Rights Reserved.

c. Set *Transparency* to the desired value.

Transparency level set to 0.

Transparency level set to 100.

4. To display dynamic measurement, set *Measurement Display On* to on.

5. In the *Dynamic Measurement* tab:

a. Set *Text Height* to the desired size in pixels.

The valid values are from 4 to 50. The default size is 10.

b. Choose the text color from the *Color* cyclic field.

Note: Dynamic measurement supports the *Create*, *Copy*, *Move* and *Stretch* commands. When using the *Move* and *Copy* commands, dynamic measurement will display only in the current cellview. If an object or group of objects are moved or copied to another cellview, the dynamic measurement is not displayed beyond the original cellview.

6. Click *OK*.

© 2020 All Rights Reserved.

Note: (ICADVM20.1 Photonics Only) In Layout EXL, the *Photonics* option is also available in the Dynamic Display form. When the *Photonics* and *Show Info Balloon* options are selected in the Dynamic Display form, information related to the photonics object is displayed as shown below.

Customizing Object Display

The type of information displayed can be customized and is object dependent. To display information about objects or to display the information dynamically when they are being created, moved, or copied, turn on the controls for information balloon and dynamic

© 2020 All Rights Reserved.

measurement and then select the type of information to be displayed in the *Setup* tab of *Dynamic Display* form. For information about the parameters that display in information balloons and those that display dynamically for various objects, see Parameter Types for Shapes and Objects.

To customize the type of information that will be displayed when using information balloons, follow these steps,

1. Select *Options – Dynamic Display*.

   The Dynamic Display Form opens.

2. In the *Setup* tab, expand the *Object Display Setting* tree, by clicking the plus signs next to the *Layout* category.

   Expanding the *Layout* tree displays the categories *Shapes* and *Objects* in which data types are organized.

© 2020 All Rights Reserved.

3. Expand the *Shapes* and *Objects* trees.

4. Select a category, shape, or object by clicking on the category, shape, or object name.

- Select a top-level category by clicking on *Layout*, *Shapes*, or *Objects* to customize the display information for all types of data within the top level category.
- Select individual shapes and objects such as rectangle or instance to customize the display information at the shape or object level.

When clicking on a specific shape or object, the *Contents* portion of the window updates to allow selection of the parameters to be displayed. Only parameters that are relevant to the shape or object are selectable. Parameters that are not relevant to the object are grayed out. See Parameter Types for Shapes and Objects.

© 2020 All Rights Reserved.

5. Click the check box next to each parameter to include the contents of that parameter in the *Info Balloon* display.

You can customize the display by selecting parameters at the top-level category and then select additional parameters to display, or remove from the display, at the individual object or shape level.

You can use the *All* or *None* buttons to select or deselect all available parameters for each shape and object rather than selecting parameters individually.

The following example shows the results of changing the display for *Rectangle*.

Rectangle
X-Length is turned on

Rectangle
All is turned on

6. Click *Apply* or *OK* once you have updated the display contents.

© 2020 All Rights Reserved.

Note: The number of decimal digits that display during the dynamic display of measurements of shapes is controlled using the displaySigDigits environment variable. However, the number of decimal places displayed for the *Area* field is fixed to `6` and is not controlled using this environment variable.

Using a Custom SKILL Function

You can define a custom SKILL extension to specify your own information to be displayed in the Info Balloon box. The information you specify appears below a dashed line in the info balloon box.

The SKILL extension must be defined at the object level. Global levels (Layout, Objects, and Shapes) are not supported. Also, the application must be specified for each SKILL function. Currently only `Layout -XL` is supported.

For example,

```
odcRegShapeRect("Layout-XL" "test custom SKILL function")
```

To turn the display of the SKILL extension on, click the *Custom SKILL Function* button on the *Setup* tab of the *Dynamic Display* form for the object specified in the SKILL function.

For a list of SKILL functions, see Object Display Control SKILL Functions in the *Virtuoso Layout Suite SKILL Functions Reference*.

Saving the Display Information Configuration

Once you have customized the object display for each shape and object, you can save the configuration setting. To customize the type of information that will be displayed when using information balloons, see Customizing Object Display.

© 2020 All Rights Reserved.

To save a configuration, follow these steps,

1. In the *Setup* tab of the *Dynamic Display* form, click *Save*.

   The *Save Configuration* form opens.

2. In the *Select configuration name* field, type a name for the configuration.
3. From the *Select path* field, choose the directory in which to save the configuration.

   The configuration is saved to the `odc.il` in the location of *`selectedPath`*`/dfII/ODC/odc.il`. The path that you choose to save the configuration to will determine the sequence in which a configuration is loaded and is derived from the `setup.loc` file. For more information about the `setup.loc` file or how to edit search order, see Cadence Setup Search File: setup.loc in the *Cadence Application Infrastructure User Guide*.

Loading the Display Information Configuration

To load a configuration, follow these steps,

➤ In the *Setup* tab of the *Dynamic Display* form, choose a configuration from the *Configuration* cycle field.

The order in which configuration files are loaded is determined by the Cadence Setup Search File mechanism (CSF). To find this information, CSF uses the `setup.loc` file. The `setup.loc` file is an ASCII file that specifies the locations to be searched and the order in which they should be searched.

© 2020 All Rights Reserved.

Saving a Configuration as Default

To save a configuration as the default configuration, follow these steps,

1. In the *Setup* tab of the *Dynamic Display* form, choose a configuration from the *Configuration* cyclic field.
2. Click *Default*.
3. Click *OK*.

The configuration you choose as default will be loaded the next time you launch Virtuoso Layout Suite L.

Deleting a Display Information Configuration

To delete a configuration, follow these steps,

1. In the *Setup* tab of the *Dynamic Display* form, click the *Delete* button.

   The *Delete Configuration* form displays.

2. Click the *Configuration* list and select the configuration to be deleted.
3. Click *OK*.

Displaying Preselect and Selected Object Information

You can view object information before (preselection mode) or after (selection mode) selecting the objects.

The preselect and selected object information is displayed by default on the dockable toolbar, *Show Selection Info*, of the design window.

© 2020 All Rights Reserved.

You can enable or disable the *Show Selection Info* toolbar. View the *Show Selection Info* toolbar using one of these methods:

- Select *Window – Toolbars – Show Selection Info.*
- Right-click anywhere in the main toolbar area and select *Show Selection Info.*

The preselect and selected object information is displayed for both fully and partially selected objects. This information is also displayed when a command is active.

© 2020 All Rights Reserved.

The following figure shows the preselect information for a rectangle.

© 2020
All Rights Reserved.

The following figure shows the select information for a rectangle.

(ICADVM20.1 Virtuoso RF Solution Only) You can view preselect and select information for curvilinear geometries, such as curved paths and curved polygons.

The following table lists the preselect and selected information displayed on the Show Selection Info toolbar for various objects:

Object	**Preselect / Select Info**
Instance	`PreSel: Inst Name(InstName) CellName(Cell)` `Select: Inst Name(InstName) CellName(Cell)`
Mosaic	`PreSel: Mosaic Name(MosaicName) CellName(Cell)` `Select: Mosaic Name(MosaicName) CellName(Cell)`

© 2020 All Rights Reserved.

Object	Preselect / Select Info

Shape (includes path, pathseg, polygon, rectangle, circle, ellipse, and donut)	`PreSel: Path Layer(layer purpose) Net(netname)`
	`Select: Path Layer(layer purpose) Net(netname)`
	Note: The abbreviation for purpose (instead of the full purpose name) is displayed. The `Net` field is displayed only if a net is present. For path and pathseg, `Width` is also be displayed. For rectangle, `Width` and `Height` is also be displayed.
	`PreSel: Path Layer(layer purpose) Net(netname)Color LockState`
	`Select: Path Layer(layer purpose) Net(netname)Color LockState`

Label	`PreSel: Label Text(TextName) Layer(layer purpose)` `Select: Label Text(TextName) Layer(layer purpose)`
PR Boundary	`PreSel: PR Boundary` `Select: PR Boundary`
Area Boundary	`PreSel: Area Boundary` `Select: Area Boundary`
Cluster Boundary	`PreSel: Cluster Boundary` `Select: Cluster Boundary`
Snap Boundary	`PreSel: Snap Boundary` `Select: Snap Boundary`

Pin	`PreSel: Pin Term(TermName) Layer(layer purpose)`
	`Select: Pin Term(TermName) Layer(layer purpose)`
	`PreSel: Pin Term(TermName) Layer(layer purpose) Color LockState`
	`Select: Pin Term(TermName) Layer(layer purpose) Color LockState`

FigGroup	`PreSel: FigGroup Name(FigGroupName)` `Select: FigGroup  Name(FigGroupName)`

© 2020 All Rights Reserved.

Object	Preselect / Select Info

Blockages (includes Routing Blockage, Fill Blockage, Slot Blockage, Pin Blockage, FeedThruBlockage, and Screen Blockage)	`PreSel: Placement Blockage Layer(layer)`
	`Select: Placement Blockage Layer(layer purpose)`
	`PreSel: Placement Blockage Layer(layer) Color`
	`Select: Placement Blockage Layer(layer purpose) Color`

Placement Blockage	`PreSel: Placement Blockage` `Select: Placement Blockage`
Halo Blockage	`PreSel: Halo Blockage` `Select: Halo Blockage`
Marker	`PreSel: Marker ShortMsg(short message) Severity (severity)` `Select: Marker ShortMsg(short message) Severity (severity)`
SyncClone	`PreSel: SyncClone  Name(name)` `Select: SyncClone  Name(name)`
Modgen	`PreSel: Modgen` `Select: Modgen`
TextDisplay	`PreSel: TextDisplay Text(Text)` `Select: TextDisplay Text(Text)`
Row	`PreSel: Row  Site(sitedefname)` `Select: Row  Site(sitedefname)`

Via	`PreSel: Via  ViaDef(ViadefName) Net(NetName)`
	`Select: Via  ViaDef(ViadefName) Net(NetName)`
	`PreSel: Via  ViaDef(ViadefName) Net(NetName) LockState`
	`Select: Via  ViaDef(ViadefName) Net(NetName) LockState`

© 2020 All Rights Reserved.

Object	Preselect / Select Info
Multipart Path	`PreSel: MPP  Layer(layer purpose) Net(NetName)` `Select: MPP  Layer(layer purpose)  Net(NetName)`
Ruler	`PreSel: Ruler Length(Length)` `Select: Ruler Length(Length)`

Special Objects

Object	Preselect Info
Fluid Guard Ring	`PreSel: Inst Name(InstName) CellName(Cell)` `Select: Inst Name(InstName) CellName(Cell)`
MPP Guard Ring	`PreSel: MPP  Layer(layer purpose) Net(NetName)` `Select: MPP  Layer(layer purpose) Net(NetName)`

ViaStack	`PreSel: Via  ViaDef(ViadefName) Net(NetName)`
	`Select: Via  ViaDef(ViadefName) Net(NetName)`
	Note: If multiple vias are selected, no object information is displayed.
	`PreSel: Via  ViaDef(ViadefName) Net(NetName) LockState`
	`Select: Via  ViaDef(ViadefName) Net(NetName) LockState`

Wire	`PreSel: PathSeg Layer(layer purpose)Net(netname)`
	`Select: PathSeg Layer(layer purpose)Net(netname)`
	Note: If a wire contains multiple pathsegs and all of them are selected, no object information is displayed.

Bus	`PreSel: PathSeg Layer(layer purpose) Net(netname)` `Select: PathSeg Layer(layer purpose) Net(netname)`

Note: For `Edit In Place` and `Descend Edit` commands, the cellname of the shape one level above in the hierarchy is also displayed in the Show Selection Info toolbar. For example, `PreSel: CellName(Cell) | Rect Layer(layer purpose) Width(n1) Height (n2)`

© 2020 All Rights Reserved.

Note: For a valid selection set, in the preselection mode only one command is performed.

Show Selection Information for Virtual Hierarchies

For virtual groups, the Show Selection Info toolbar displays the virtual group name, the virtual group type—*generated*, *clone*, or *created*, the *PlacementStatus* of the group, and the display depth stop level value.

For virtual groups, the Show Selection Info toolbar displays the pin layer and net on which the pin is created and the pin width and height.

Object	Preselect/Select Info
Virtual Group	PreSel: Virtual FigGroup Name (`figGroupName`) Type (`Generated, Clone, Created`) Fixed StopLevel (`0-32`) Select: Virtual FigGroup Name (`figGroupName`) Type (`Generated, Clone, Created`) StopLevel (`0-32`) **Note:** If the *Placement Status* of the virtual hierarchy is not set to `Fixed`, the Show Selection Info toolbar displays no information about the placement status.
Virtual Pin	PreSel or Select: Virtual Pin Layer (`name purpose`) Net (`netname`) Width Height

The type of object information (preselect or selected) that is displayed in the status line of the design window depends on the number of objects selected and the location of the mouse pointer in the design area.

- When no object is selected in the design area and:
  - The mouse pointer is placed on empty space:

    No object information is displayed in the status line in the design window.
  - The mouse pointer is placed on an unselected object:

    The preselect information of the object on which the mouse pointer is currently placed is displayed in the status line of the design window.
  - The mouse pointer is placed on an unselected object and the left mouse button is clicked:

© 2020 All Rights Reserved.

The selected information of the object is displayed in the status line of the design window until the mouse pointer starts moving.

- When one object is selected in the design area and:
  - The mouse pointer is placed on the selected object:

    The preselect information of the object on which the mouse pointer is currently placed is displayed in the status line of the design window.
  - The mouse pointer is placed on empty space:

    The selected information of the object is displayed in the status line of the design window.
  - The mouse pointer is placed on an unselected object:

    The preselect information of the object is displayed in the status line of the design window.
  - The mouse pointer is placed on an unselected object and the left mouse button is clicked:

    The selected information of the object is displayed in the status line of the design window until the mouse pointer starts moving.
- When multiple objects are selected in the design area and:
  - The mouse pointer is placed on any selected object(s):

    The preselect information of the object is displayed in the status line of the design window.
  - The mouse pointer is placed on empty space:

    No object information is displayed in the status line in the design window.
  - The mouse pointer is placed on an unselected object:

    The preselect information of the object is displayed in the status line of the design window.
  - The mouse pointer is placed on an unselected object and the left mouse button is clicked:

    The selected information of the object is displayed in the status line of the design window until the mouse pointer starts moving.

    Note: In case of overlapping objects, the selected information of the object is displayed in the status line of the design window. When the left mouse button is

© 2020 All Rights Reserved.

clicked again, the selected information of the next object is displayed in the status line of the design window.

Cycling Through Preselected Objects

When the pointer is on overlapping objects in the design window, press the `Space` key to cycle through the preselected objects. In the example below, the pointer is in the area where the three objects, M1, M2, and M3 overlap. You can cycle through the preselected objects (dynamic highlight) using the `Space` key.

The preselect information will be updated simultaneously in the dynamic highlight, *Show Selection Info* toolbar, and Info Balloon. You can cycle through the preselected objects even if the *Dynamic Highlight* option is not selected in the Display Options form. When you use dynamic highlight, the bounding box and label are selected or dynamically highlighted.

Note: If the `Space` key is being used by another command, such as *Quick Align*, *Create Measurement*, or *Create Wire*, the `Space` key will not cycle the preselection inside that command.

Cycle Selection in Partial Selection Mode

You can select and cycle through a shape in the partial selection mode. On the first click, the edge of the inner shape is selected.

© 2020 All Rights Reserved.

On the second click, the edge of the outer shape is selected.

© 2020 All Rights Reserved.

3

Palette Assistant

This chapter contains the following topics:

- Features of Palette Panels
- Window Context and Layer Sets
- About Synchronized Palettes
- Displaying and Hiding the Single Palette Assistant
- Palette Containers
- Layers Panel
- Objects Panel
- Grids Panel

Palette is a tool used to manage the context of layer, object, and grid items by editing their properties, such as visibility and selectability. The context is initialized with the information found in the technology file.

When you open a cellview, Palette is displayed by default, docked to the left of the layout window. It comprises three panels: *Layers*, *Objects*, and *Grids*. You can configure Palette to display the panels in three different types of containers, namely, single assistant, multiple assistants, and single window. By default, Palette opens in the single assistant container. For more information, see Palette Containers.

Features of Palette Panels

The key features of the Palette panels are listed below:

- Can be used to configure the context of layer, object, and grid items by setting their visibility and selectability.
- Provide access to various layer properties, including routing directions, GDS numbers, and technology file properties such as validity, priority, function, description, and mask.

© 2020 All Rights Reserved.

- Can be used to edit the routing direction of layers.
- Provide options to filter layer, object, and grid items.
- Can be used to manage the properties of layers individually and in groups.
- Support both synchronized and independent contexts for palettes attached to different layout windows.
- Provide access to the Options form that can be used to configure Palette default behavior.
- Can be used to configure Palette settings. The Palette configuration settings can be saved as part of a Virtuoso workspace.

Note: Palette is activated by default in the Classic and Basic workspaces.

Window Context and Layer Sets

Use Palette to manage the following:

- Window Context
- Layer Sets

Window Context

A window context determines display properties such as visibility and selectability of layers, objects, and grids in a layout window. Each layout window is controlled by a separate Palette. By default, all palettes initialized using the same technology file are synchronized. For example, the visibility property of a layer is identical in all layout windows if the palettes in these windows are synchronized.

For more information, see About Synchronized Palettes, Window Context (toolbar) and Synchronize/Desynchronize Window.

Note: You can save the current window context to a technology file or an LSW Info file and reload it later.

Layer Sets

A layer set contains a subset of layer-purpose pairs. You can use the Palette to set a layer set as active and to create new and modify existing layer sets. When you set a layer set as active, all layers, objects, and grids associated with that layer set are displayed in the Palette.

© 2020 All Rights Reserved.

Note: The *All Layers* layer set is a system-defined layer set. You can apply filters to this layer set, but you cannot add or remove layers from it.

For more information, see Layer Set Manager and Layer Set.

Note: The `.ini` configuration files from the old Palette are not compatible with the new Palette. To organize new configurations of Palette, save the settings again to a workspace.

About Synchronized Palettes

- Palettes that point to the same technology file are synchronized by default. If two palettes are synchronized, changes made in one Palette are reflected in the other Palette. For example, if you change the visibility of a layer-purpose pair in one Palette, the visibility of that layer-purpose pair are automatically updated in the other Palette.
- Palettes are desynchronized or independent of one another if they are linked to different technology files. For example, if the palettes in layout windows 1 and 2 point to different technology files, namely techfile 1 and techfile 2, the palettes are independent. The changes made in an independent Palette are not reflected in the other palettes.

If required, synchronized palettes can be desynchronized. For more information, see Window Context (toolbar) and Synchronize/Desynchronize Window.

Displaying and Hiding the Single Palette Assistant

To display the single Palette assistant, do one of the following:

- Choose *Window – Assistants – Palette*.
- Right-click anywhere in the layout window menu bar and choose *Assistants – Palette*.

To hide the single Palette assistant, do one of the following:

- Click the *Hide* button *(X)* in the Palette title bar.
- Right-click anywhere in the layout window menu bar and choose *Assistants – Palette*.
- Use the `hiUnMapWindow(dwindow('ptePaletteAssistantX'))` function, where X is the Palette instance number.

Palette Title Bar Buttons

The Palette title bar has the following buttons:

© 2020 All Rights Reserved.

- *Help*

  Displays information about how to use the Palette.

- *Float/Dock*

  Undocks the Palette assistant. Click the button again to dock the Palette assistant.

  Note: You can also drag the Palette to dock or undock it.

- *Hide*

  Hides the Palette assistant.

Palette Containers

Virtuoso Layout Suite supports the following types of Palette containers:

- Single Assistant Container (Default)
- Multiple Assistants Container
- Single Window Container

Single Assistant Container

Single assistant container encapsulates the *Layers*, *Objects*, and *Grids* panels into a single Palette assistant. Therefore, when you hide the Palette assistant, all three panels are hidden

© 2020 All Rights Reserved.

from view. If required, you can hide each panel individually. You can also dock or undock the entire Palette or each of the three panels individually.

Note: The single assistant container is linked to the layout window. Therefore, the assistant is automatically closed when the layout window is closed.

Multiple Assistants Container

Multiple assistants container displays the *Layers*, *Objects*, and *Grids* panels as three separate assistants. These are also listed as three separate assistants on the *Assistants* menu that is displayed when you right-click in the menu bar area of the layout window. You

© 2020 All Rights Reserved.

can hide or unhide each panel individually. Each panel can also be docked or undocked individually.

Note: The single assistant container is linked to the layout window. Therefore, the assistant is automatically closed when the layout window is closed.

Single Window Container

The single window container is similar to the Layer Selection Window (LSW). It encapsulates the *Layers*, *Objects*, and *Grids* panels in a single window that is independent of the layout

© 2020 All Rights Reserved.

window. In this mode, a single Palette window is used to manage all open designs. The three panels can be docked or undocked individually. You can also hide or unhide them individually.

Note: The Palette window is closed automatically when you exit the Virtuoso session.

To resize the window, use `pteResizeSingleWindowPalette`; to set the location of the window, use `pteMoveSingleWindowPalette`.

Setting the Palette Type

The default Palette container type is `SingleAssistant`. You can change the Palette container type by setting the `CDS_PALETTE_TYPE` Unix environment variable.

Type one of the following commands in the shell environment to set the Palette type:

```
setenv CDS_PALETTE_TYPE SingleAssistant
```


© 2020 All Rights Reserved.

```
setenv CDS_PALETTE_TYPE MultiAssistant

setenv CDS_PALETTE_TYPE SingleWindow
```

Important

The `CDS_PALETTE_TYPE` Unix environment variable value must be set before running Virtuoso. Its value cannot be changed during a Virtuoso session.

Layers Panel

By default, the *Layers* panel lists the layer-purpose pairs defined in the `leLswLayers` section of the technology file. Only valid layer-purpose pairs are displayed if the *Valid* check box is selected in the *Scope* toolbar. For more information, see Scope.

This section contains the following topics:

- Displaying and Hiding the Layers Panel
- User Interface
- Using Mouse Functions to Control Layer-Purpose Pairs
- Tasks in the Layers Panel

Displaying and Hiding the Layers Panel

Use one of the following methods to display the *Layers* panel:

- Right-click any panel title bar and choose *Layers*.
- Use the `pteMapWindow` function to specify the name of the Palette panel to be displayed.

Use one of the following methods to hide the *Layers* panel:

- Right-click any panel title bar and choose *Layers*.
- Use the `pteUnmapWindow` function to specify the name of the Palette panel to be hidden.

© 2020 All Rights Reserved.

User Interface

The default Palette user interface is shown below:

The *Layers* panel user interface comprises the following components:

- Title Bar Buttons
- Toolbars
- Palette Menu
- Column Header Menu
- Columns

© 2020
All Rights Reserved.

Title Bar Buttons

The *Layers* panel title bar has the following buttons:

- *Float/Dock*

  Undocks the *Layers* panel. Click the button again to dock the panel.

- *Hide*

  Hides the *Layers* panel.

Toolbars

The *Layers* panel contains the following toolbars by default:

- Layer Set
- Scope
- Search (Filter/Find)
- Active Layer
- Control Buttons

The following are additional toolbars available in the *Layers* panel.

- Edit
- Window Context (toolbar)
- Layer Set Manager

© 2020 All Rights Reserved.

To display these toolbars, right-click any toolbar that is displayed in the *Layers* panel, and then point to *Tools* and choose the required toolbar, as shown:

Note: The toolbars with the red check mark displayed next to them are available by default in the *Layers* panel.

Layer Set

The *Layer Set* toolbar lists all layer sets available in the current design.

To make a layer set active, select it from the *Layer Set* list. All layers contained in the selected layer set are displayed in the *Layers* panel.

Corresponding SKILL Function: `pteSetLSActive`

Use the *Layer Set* commands available on the Palette menu to import, export, reload, save, and delete layer sets. For more information, see Layer Set.

© 2020 All Rights Reserved.

Important

The *Layer Set* toolbar enables you to select only one layer set at a time. To select multiple layer sets at a time, use the Layer Set Manager.

Scope

The *Scope* toolbar provides options for filtering layers; namely, *Valid*, *Used*, and *Routing*.

- **Valid:** If selected, displays only valid layers.
- **Used:** If selected, displays only the layers that contain design objects of type shape. If you want to view layers that contain all types of design objects, such as blockages, boundaries, instances, mosaics, markers, and rows, set the pteShowUsedSystemLpps environment variable to `t`, and then select the *Used* check box. You need to reapply the filter to refresh the display in the *Layers* panel if you add a design object to a previously unused layer while the filter is still applied.
- **Routing:** If selected, displays only the routing layers.

  Routing layers are identified at the start of a Virtuoso session. By default, the following layers are identified as routing layers:

  - All layers with function *metal*, *cut*, and *poly* and purpose *drawing* through *drawing9*

    Note: You can exclude *cut* layers from routing layers by using the pteIncludeCutLayers environment variable. The value of this variable must be set before opening a design. Once set, it is applicable for the entire session.

  - All layers with purpose *drawing* through *drawing9* present in the *validRoutingLayers* constraint
  - All combinations of the layers and purposes present in the *validRoutingLPPs* constraint.

© 2020 All Rights Reserved.

For example, you can add layer purpose pairs, `diff/net` and `ndiff/pin`, to the *validRoutingLPPs* constraint. This means that following layer purpose pairs are treated as routing layer purpose pairs, `diff/net`, `diff/pin`, `ndiff/net`, and `ndiff/pin`.

Search (Filter/Find)

Use the *Search* toolbar to search for specific layers. The default search mode is *Filter*. You can usethe toolbar or the Options form to toggle between *Filter* and *Find* modes. For more information, see Search for a Layer.

Active Layer

The *Active* (*Layer*) toolbar displays the name of the layer that is currently active. The *Active* (*Layer*) list stores the history of up to 10 active layers for the current session.

Control Buttons

Use the *Control Buttons* toolbar to manage the visibility and selectability of layer-purpose pairs. The toolbar has the following control buttons:

- *AV* — All Visible

© 2020 All Rights Reserved.

Corresponding SKILL Function: `pteSetAllVisible`

- *NV* — None Visible

  Corresponding SKILL Function: `pteSetNoneVisible`

- *AS* — All Selectable

  Corresponding SKILL Function: `pteSetAllSelectable`

- *NS* — None Selectable

  Corresponding SKILL Function: `pteSetNoneSelectable`

You can use the following three methods to control the visibility and selectability of layers:

- Click a button on the *Control Buttons* toolbar or click the first option available on the drop-down menu associated with the button (for example, *AV* shown in the figure below).

  Controls the visibility and selectability of all layers that are currently displayed in the *Layers* panel. For example, to set the selectability of all *metal* layers in the current layer set to not selectable, type `metal` in the *Filter* field, and then click the *NS* button.

- Click the second option available on the drop-down menu associated with a control button (for example, *AV - Layer Set* shown in the figure below).

  Controls the visibility and selectability of all layers in the active layer set, irrespective of the layers displayed in the *Layers* panel.

- Click the third option available on the drop-down menu associated with a control button (for example, *AV - Technology File* shown in the figure below).

  Controls the visibility and selectability of all layers in the technology file.

© 2020
All Rights Reserved.

Note: The drop-down menus associated with control buttons are also available in the *Objects* and *Grids* panels.

Edit

The *Edit* toolbar provides the following options to add or remove items from layer sets:

- **Member:** Select the *Member* check box to display the Membership column (*m*) in the *Layers* panel. Select or deselect a check box in the *m* column to add or remove the corresponding layer from the current layer set.

  Corresponding SKILL Functions:

  - `pteSetAllLayerSetMember`
  - `pteSetLayerSetMember`
  - `pteSetNoneLayerSetMember`
  - `pteEditLayerSet`
  - `pteCloseLayerSetEdition`
  - `pteToggleAllLayerSetMember`

- **Validity:** Layers are specified as valid or invalid in the technology file. When you select the *Validity* check box, both valid and invalid layers are displayed in the *Layers* panel, and you also see both Validity (*v*) and Membership (*m*) columns displayed in the *Layers* panel.

© 2020 All Rights Reserved.

Note: Invalid layer-purpose pairs are displayed in red.

You cannot set as valid a layer-purpose pair by using the Palette if:

- The layer or the purpose is set as invalid in the technology file, that is, the `valid` attribute for it is set to `nil`.
- The layer, the purpose, or the layer-purpose pair is locked, that is, the `allowSetToValid` and `allowSetToValidInSession` attributes for it are set to `nil` in the technology file.

However, you can use the Palette to set a locked layer-purpose pair as invalid.

For information about validity and lock attributes for layers, purposes, and layer-purpose pairs, see LPP Valid Functionality in *Virtuoso® Technology Data ASCII Files Reference*.

For information about the `techIsLPValidBase` SKILL API, see *Virtuoso® Technology Data SKILL Reference*.

Corresponding SKILL Functions:

- `pteEditLayerSetValidity`
- `pteSetAllValidity`
- `pteSetNoneValidity`
- `pteSetValidity`

© 2020 All Rights Reserved.

- `pteShowValidLPP`
- `pteToggleAllValidity`
- `pteToggleLayerSetValidityEdition`
- `pteCloseLayerSetEdition`

Note: You cannot set an invalid layer as the active layer because such layers cannot be used effectively in a design; for example, you cannot create shapes on an invalid layer.

In addition, the *Edit* toolbar provides a *Discard Edits* button that you can use to discard all changes made to the membership and validity status of layers.

Corresponding SKILL Function: `pteDiscardLayerSetEdition`

Window Context (toolbar)

The (*Window*) *Context* toolbar displays the name of the current display context, which, by default, is the technology file name.

Note: Palettes with the same window context name are synchronized.

In addition, the (*Window*) *Context* toolbar contains a *Synchronize/Desynchronize* button. Use this button to synchronize or desynchronize the palette. For more information, see About Synchronized Palettes and Synchronize/Desynchronize Window.

© 2020
All Rights Reserved.

Layer Set Manager

The *Layer Set Manager* lists the layer sets that are enabled in the current design window.

Use the *Layer Set Manager* to perform the following tasks:

- Reorder layer sets by dragging them up and down the list.
- Select one or more layer sets by using the following methods:
  - To select a single layer set, click that layer set.
  - To select multiple adjacent layer sets, click the first layer set in the sequence, and then hold down the `Shift` key and click the last layer set in the sequence.
  - To select multiple non-adjacent layer sets, select a layer set, and then hold down the `Ctrl` key and click the other layer sets that you want to select.

  All layer sets selected in the *Layer Set Manager* are considered active and a layer-purpose pair is displayed in the *Layers* panel if it is a member of at least one of the selected layer sets.
- Turn on or off the visibility or selectability status of all the layers present in a layer set by clicking the corresponding + or – button, respectively. These buttons work with respect to the filter settings defined on the *Scope* and *Filter* toolbars if the pteLSManagerRespectFilters environment variable is set to `t`.
- Set a layer set to active and turn on the visibility and selectability of all its member layers by clicking the layer set with the middle mouse button. The visibility and selectability for all other layers is turned off.

© 2020 All Rights Reserved.

- Disable a layer set by deselecting the corresponding check box in the column labeled *E*. This column is displayed when you right-click a column header in the *Layer Set Manager* and choose *Columns – Enable Layer Set* from the shortcut menu.

Note: The layer set that is currently active cannot be disabled.

- Save the layer set order and status—whether enabled or disabled—in the `layerset.order` file. For more information about how the `layerset.order` file is saved and loaded, see Layer Set.

Note: Names of layer sets that are not synchronized with corresponding layer set files are italicized.

© 2020 All Rights Reserved.

Palette Menu

The Palette menu is a context-sensitive menu that is displayed when you right-click any toolbar in the *Layers* panel or anywhere in the *Objects* and *Grids* panels (except the title bar and the column headers).

The following options are available on the Palette menu:

- Synchronize/Desynchronize Window
- Context
- Layer Set
- Edit Valid Layers
- Edit Layer Set Members
- Discard Edit
- Edit Display Resources
- Load
- Save

© 2020 All Rights Reserved.

- Show Tools
- Options
- Bindkeys

Synchronize/Desynchronize Window

Use this command to synchronize or desynchronize a palette. For more information, see About Synchronized Palettes.

Corresponding SKILL Function: `pteSetWindowSynchro`

Note: This option works in the same manner as the *Synchronize/Desynchronize* button available on the Window Context (toolbar) toolbar.

Context

This is the Palette window context.

© 2020 All Rights Reserved.

Layer Set

The *Layer Set* submenu includes the following commands:

- *Save*

  Saves the changes made to a layer set file. The *Save* command is disabled when the *All Layers* layer set is selected. This is because *All Layers* is a system-generated layer set and cannot be modified.

  Corresponding SKILL Function: `pteSaveLayerSet`

- *Save As*

  Saves the layer set as a `<layersetname>.layerSet` file in the specified layer set repository, which is a directory that stores layer set files.

  The newly created layer set is automatically set as active and is displayed in the *Layer Set* list and the *Layer Set Manager*. For more information about how you can create a new layer set and synchronize it with another layer set, see Create a Layer Set.

  Corresponding SKILL Functions: `pteSaveAsLayerSet`, `pteSaveAsSynchronizedLayerSet`

- *Reload*

  Reloads the layer set that is currently in use. After making changes to the layer attributes, use the reload operation to re-synchronize the layer, object, and grid items.

© 2020 All Rights Reserved.

The visibility and selectability statuses of layer, object, and grid items are also re-synchronized if the *Layer Set Activation* option is set to *Filter and Apply V/S* in the Options form. For more information about the Options form, see Set Palette Options.

Corresponding SKILL Function: `pteReloadLayerSet`

- *Delete*

  Deletes a layer set from the *Layers* panel as well as from the layer set repository. For more information, see Delete a Layer Set.

  Corresponding SKILL Function: `pteDeleteLayerSet`

- *Import*

  Imports a layer set file from the file system to the layer set repository. The imported layer set is added to the *Layer Set* list in the *Layers* panel. For more information, see Import a Layer Set.

  Corresponding SKILL Function: `pteImportLayerSet`

- *Export*

  Helps save a layer set that exists in the layer set repository to the file system. For more information, see Export a Layer Set.

  Corresponding SKILL Function: `pteExportLayerSet`

- *Save Layer Set List*

  Saves the layer set order and status in the `layerset.order` file in the `.cadence` directory at the specified location, which you can select from the *Layer Set repository* list in the Save Layer Set List form. When you start a new Virtuoso session, the `layerset.order` file is loaded automatically based on the order of precedence defined in the `setup.loc` file.

  Corresponding SKILL Function: `pteSaveLayerSetListInRepository`

For more information about how you can manage layer sets, see Layer Set Manager.

Edit Valid Layers

Use this command to display the Membership (*m*) and Validity (*v*) columns in the *Layers* panel. For more information, see Edit.

Corresponding SKILL Functions: `pteEditLayerSetValidity`, `pteCloseLayerSetEdition`

© 2020 All Rights Reserved.

Edit Layer Set Members

Use this command to display the Membership column (*m*) in the *Layers* panel. For more information, see Edit.

Corresponding SKILL Functions: `pteEditLayerSet`, `pteCloseLayerSetEdition`

Discard Edit

Use this command to discard all changes made to the membership and validity status of layers. For more information, see Edit.

Corresponding SKILL Functions: `pteDiscardLayerSetEdition`

Edit Display Resources

Use this command to display the Display Resource Editor form. In this form, you can change the appearance of the icon that is displayed for a layer in the *Swatch* column. For more information, see *Virtuoso Technology Data User Guide*.

Load

Use this command to update the current display context with information from one of the following file types:

- LSW information file
- Technology file
- GDS Number file
- Defaults

Corresponding SKILL Functions: `pteLoadLSWInfo`, `pteLoadFromTechFile`, `pteLoadGDSNumber`, `pteLoadDefaults`

Save

Use this command to save the current display context into one of the following file types:

- LSW information file
- Technology file

© 2020 All Rights Reserved.

- GDS Number file

Corresponding SKILL Functions: `pteSaveLSWInfo`, `pteSaveToTechFile`, `pteSaveGDSNumber`

Show Tools

Use commands in the *Show Tools* submenu to selectively display or hide toolbars. You can display or hide all toolbars by using the *Show All* or *Hide All* commands, respectively.

Note: The options available on the *Show Tools* submenu vary depending on the panel in which you right-click.

© 2020 All Rights Reserved.

Options

Choose *Options* on the Palette menu to open the Options form. In this form, you can define local and global Palette options. For more information, see Set Palette Options.

Corresponding SKILL Function: `pteSetOptionString`

Bindkeys

Use this command to open the *Bindkey Editor*, which displays a list of the default bindkeys defined for Palette-related options.

Use the *Bindkey Editor* to modify existing bindkeys or to define new bindkeys. For more information about using *Bindkey Editor*, see Configuring Application Bindkeys.

Columns

The *Layers* panel, by default, displays the following columns:

- *Swatch*, which sorts layer-purpose pairs in the order in which they appear in the `leLSWLayers` section of the technology file.
- *Layer,* which displays the layer names.
- *Purpose,* which displays the purpose associated with the layers.

© 2020 All Rights Reserved.

(ICADVM20.1 Virtuoso RF Solution Only) For dynamic purposes, you can view void layers defined in the technology file.

- *V* (Visibility), which determines whether the instances and objects on a layer are visible.
- *S* (Selectability), which determines whether the instances and objects on a layer are selectable.

Tip

To turn on or off the visibility or selectability of all layer-purpose pairs, objects, or elements together, click the *V* or *S* column header, respectively. To sort the layer-purpose pairs, objects, or elements based on their visibility or selectability, hold down the `Ctrl` key and click the *V* or *S* column header. This default behavior can be modified by using the *Checkable Column Click* option in the Palette Options form. For more information, see Set Palette Options.

All *Layers* panel columns are not displayed by default. For more information about how to display these additional columns, see Columns.

Column Header Menu

When you right-click a column header in the *Layers* panel, a context-sensitive menu is displayed.

The following commands are available on the column header menu:

© 2020 All Rights Reserved.

- Columns
- View By
- Sort By
- Hide This Column
- Show

Columns

The *Layers* panel *Columns* menu lists the following column names:

Swatch, *Layer*, *Purpose*, *Function*, *Routing Dir*, *Mask*, *GDS Number*, *Priority*, *Min Voltage*, *Max Voltage*, *Signal Type*, *Parent*, *Description*, *Visibility*, *Selectability*, *Stipple*, and *LS Order*.

Use this menu to selectively display columns in the *Layers* panel. You can hide or display all columns, except *Swatch*, by selecting the *Hide All* or *Show All* commands, respectively.

Note: All columns except *V* (Visibility) and *S* (Selectability) can be resized by dragging the column margin separator.

© 2020 All Rights Reserved.

View By

Use the *View By* menu to display layer-purpose pairs in a tree structure in the *Layers* panel. You can view layer-purpose pairs on the following columns:

Layer, *Purpose*, *Function*, *Color*, *Routing Dir, Mask*, *GDS Number*, *Priority*, *Min Voltage*, *Max Voltage*, *Signal Type*, *Parent*, *Description*, *Layer Set List*, *and None*.

The column using which the layer-purpose pairs are arranged appears first.

Note: If *View By – None* is selected, the order in which layer-purpose pairs are displayed is determined by the column on which layer-purpose pairs are sorted. For example, if layer-purpose pairs are sorted on the *Layer* column and you select *View By – None,* the layer-purpose pairs are displayed in the alphabetical order by layer name.

If *View By – Layer* is selected, the layers with the same name, but different purposes, are grouped together. As a result, you can control the visibility and selectability of a layer and all the purposes associated with that layer by selecting or deselecting the *V* (Visibility) check box corresponding to the layer name. For example, if *Metal1* layer has different purposes such as *drawing*, *slot*, and *label*, you can control the visibility and selectability of all *Metal1* layer-purpose pairs together by selecting or deselecting the *V* (Visibility) and *S* (Selectability) check boxes corresponding to *Metal1* layer. You can also modify the visibility and selectability of individual layer-purpose pairs, if required.

© 2020 All Rights Reserved.

You can use the *LPP Group* option to group layers by specific purposes. To view the *LPP Group* option in the *View By* menu, you need to set the environment variable, pteLppGroup.

```
envSetVal("layout" "pteLppGroup" 'string "drawing label")
```

In the above example, when LPPs are groups by purposes drawing and label as specified in the environment variable, `pteLppGroup`.

© 2020
All Rights Reserved.

Selecting *View By – Advanced* displays the View By form. Use this form to specify display parameters at multiple levels. Layer-purpose pairs are displayed in a tree structure as per your specifications.

Sort By

Use the *Sort By* menu to sort layer-purpose pairs based on different columns. Each command on the *Sort By* menu corresponds to a column that can be displayed in the *Layers*

© 2020
All Rights Reserved.

panel. By default, the layer-purpose pairs in the *Layers* panel are sorted in the order in which they are defined in the `leLswLayers` section of the technology file.

The following sorting methods are available:

- *Swatch*

  Sorts layer-purpose pairs in the order in which they appear in the `leLSWLayers` section. If this section is not available, layers are sorted based on their priorities.

- *Layer*

  Sorts layer-purpose pairs in the alphabetical order by layer name. Layers with the same name but different purposes are sorted in the order of their priority. Layers for which a name is not specified are listed first, followed by layers with names.

- *Purpose*

  Sorts layer-purpose pairs by their purpose, in the alphabetical order. Layers with the same purpose but different names are sorted in the order of their priority. Layers for which a purpose is not specified are listed first, followed by the layers with purposes.

- *Function*

© 2020 All Rights Reserved.

Sorts layer-purpose pairs by their function, in the alphabetical order. Layer-purpose pairs for which the function is not specified are listed first, followed by the layer-purpose pairs for which the function is specified.

Note: If none of the layer-purpose pairs have functions specified, the layer-purpose pairs are sorted in the order of their priority.

- *Routing Dir*

  Sorts layer-purpose pairs by their routing direction. Layer-purpose pairs for which the routing direction is not defined are listed first, followed by the layer-purpose pairs with the routing direction defined. Layer-purpose pairs are sorted in the following order of routing direction:

  Horizontal, *Left Diagonal*, *None*, *Right Diagonal*, and *Vertical*

  You can change the routing direction by clicking in the *Routing Dir* column for a layer-purpose pair. A list is displayed as shown in the following figure:

  To change the routing direction for the selected layer-purpose pair, select a routing direction from the displayed list.

- *Mask*

  Sorts layer-purpose pairs in the ascending order of the mask number assigned to them in the `Layer Definition` section of the technology file. Layer-purpose pairs that do not have mask numbers assigned are listed first, followed by the layer-purpose pairs with mask numbers.

  Note: If none of the layer-purpose pairs have mask numbers assigned, the layer-purpose pairs are sorted in the order of their priority.

- *GDS Number*

  Sorts layer-purpose pairs in the increasing order of their GDS numbers.

  The *GDS Number* column displays both the Layer Stream Number and the Datatype Stream Number separated by a '*:*'. These numbers are present in the optional

© 2020 All Rights Reserved.

`<techLibName>.layermap` file that is stored in the technology library directory. If this file is not found, the Palette automatically searches for a `<techLibName>.gdsnumber` file in the `.cadence/dfII/gdsnumber` directory. If both these files are not available, nothing is displayed in the *GDS Number* column.

- *Priority*

  Sorts layer-purpose pairs in the descending order of the priorities assigned to them in the `Layer Definition` section of the technology file. Layer-purpose pairs with priority *0* are displayed first.

- *Min Voltage*

  Sorts layer-purpose pairs in the ascending order of the minimum voltage assigned to them in the `Layer Definition` section of the technology file. The value is either an integer or a float value. The layer-purpose pairs for which the minimum voltage is not specified are listed first, followed by the layer-purpose pairs for which the minimum voltage is specified.

- *Max Voltage*

  Sorts layer-purpose pairs in the ascending order of the maximum voltage that is assigned to the layer-purpose pair in the `Layer Definition` section of the technology file. The value is either an integer or a float value. Layer-purpose pairs for which the maximum voltage is not specified are listed first, followed by the layer-purpose pairs for which the maximum voltage value is specified.

- *Signal Type*

  Sorts layer-purpose pairs in the alphabetical order by signal type. The signal type indicates the signal that is carried by the nets routed with the layer-purpose pair. The signal type for a layer-purpose pair is specified in the `Layer Definition` section of the technology file. Layer-purpose pairs for which a signal type is not defined are listed first, followed by layer-purpose pairs with a signal type defined.

- *Parent*

  Sorts layer-purpose pairs by the parent purpose of the layer-purpose pair, in the alphabetical order. The parent purpose is the main purpose of the layer from which you can define other purposes. The parent purpose for the layer is defined in the `Layer Definition` section of the technology file. The layer-purpose pairs for which the parent purpose is not defined are listed first, followed by the layer-purpose pairs for which the parent purpose is defined.

- *Description*

  Sorts layer-purpose pairs by their descriptions, in the alphabetical order. The description field provides additional information about each layer-purpose pair, as specified in the

© 2020 All Rights Reserved.

`Layer Definition` section of the technology file. The layer-purpose pairs for which a description is not specified are listed first, followed by the layer-purpose pairs with a description.

- *V* (Visibility)

  Sorts layer-purpose pairs by their visibility statuses. Layer-purpose pairs with the visibility check box deselected are listed first, followed by layer-purpose pairs with the visibility check box selected.

- *S* (Selectability)

  Sorts layer-purpose pairs by their selectability statuses. Layer-purpose pairs with the selectability check box deselected are listed first, followed by layer-purpose pairs with the selectability check box grayed out. Layer-purpose pairs with the selectability check box selected are listed last.

- *Stipple*

  Sorts layer-purpose pairs by the their stipple patterns. Layer-purpose pairs with the stipple check box deselected are listed first, followed by layer-purpose pairs with the stipple check box selected.

- *LS Order*

  Sorts layer-purpose pairs in the increasing order of their layer set order numbers.

- *Reverse*

  Reverses the order in which the layer-purpose pairs are currently displayed.

Hide This Column

Use this command to hide a column in the *Layers* panel.

Show

Use the *Show* menu as an extended filter mechanism to filter layers based on the values of specific columns. Right-click a column header, except *V* (Visibility) and *S* (Selectability)

© 2020 All Rights Reserved.

columns, and choose *Show*. In the following example, the *Show* menu is displayed for the *Purpose* column.

The *Show* menu includes the following filter options:

- **All:** No filter is applied. All layers are displayed.
- **Custom:** Enables you to define criteria based on which layers are filtered. For example, selecting the *Custom* option for the *Purpose* column displays the Custom Show Purpose form.

© 2020
All Rights Reserved.

To define a criterion, select an operator, a purpose, and click either *OK* or *Apply*. You can add multiple criteria. Click *Add Criteria* to add a new criterion. Depending on your requirement, you can choose to match any or all criteria.

For more information, see Custom Show Purpose Form.

- **Selected values:** You can also select the required purposes from the displayed list. To do this, hold down the `Ctrl` key while selecting multiple purposes, and then click *OK* to apply the filter. Layers with the selected purposes are displayed in the *Layers* panel.

Using Mouse Functions to Control Layer-Purpose Pairs

You can perform the following actions to control layer-purpose pairs in the single-selection mode:

- To turn on or off the visibility of a layer-purpose pair, click the layer-purpose pair by using the middle mouse button.
- To turn on or off the selectability of a layer-purpose pair, right-click the layer-purpose pair.

  Note: The two functions listed above are available only if the pteEnableMouseBindings environment variable is set to `t`.

- To set a layer-purpose pair as current (active), click the layer-purpose pair. The visibility status of the layer-purpose pair is turned on.
- To change the appearance of a layer-purpose pair, hold down the `Shift` key and click the layer-purpose pair. The Display Resource Editor form is displayed. For more information, see Edit Display Resources.

  Note: This function is disabled if a user-defined trigger is registered. For more information, see `pteRegisterUserSelectionTrigger`.

© 2020 All Rights Reserved.

- To make all but one layer-purpose pairs invisible, hold down the `Shift` key and use the middle mouse button to click the layer-purpose pair that you want to make visible.
- To turn off the selectability of all layers but one, hold down the `Shift` key and right-click that layer-purpose pair.

You can perform the following actions to control layer-purpose pairs in the multi-selection mode:

- To select the first layer-purpose pair and to set it as current (active), click the layer-purpose pair. The visibility status of the layer-purpose pair is turned on.
- To extend the selection, hold down the `Ctrl` or `Shift` key and click the layers that need to be added to the selection.
- To add or remove a layer-purpose pair from the current selection, hold down the `Ctrl` key and click that layer-purpose pair.
- To view the multi-selection menu, right-click a layer-purpose pair.
- To reorder layer-purpose pairs, drag the required layer-purpose pair or a group of layer-purpose pairs up or down the list.

For more information, see Select Multiple Layers.

© 2020 All Rights Reserved.

Tasks in the Layers Panel

By using the *Layers* panel, you can perform the following tasks.

- Search for a Layer
- Change the Current Active Layer
- Change Layer Appearance
- Load LSW Information
- Save LSW Information
- Create a Layer Set
- Delete a Layer Set
- Import a Layer Set
- Export a Layer Set
- Set Palette Options
- Objects Panel
- Select Multiple Layers
- MPT Support

Search for a Layer

Use the *Search* toolbar to search for specific layer-purpose pairs. The default search mode is *Filter*. You can set the search mode to *Find* by using the toolbar or the Options form. For more information, see Options.

When the search mode is set to *Filter*, only the layer-purpose pairs that match the specified search criteria are displayed in the *Layers* panel and the active layer information does not change.

In contrast, in *Find* mode, all layers continue to be displayed in the *Layers* panel and the first layer that matches the search criteria becomes the active layer. If you now press `Enter`, the next layer that matches the search criteria becomes the active layer. You can continue to press `Enter` to move through the list of matching layer-purpose pairs in a cyclic manner.

Additionally, in *Find* mode, search is performed only on the layers that are currently displayed in the *Layers* panel. For example, consider a layer set named *ls1* that contains layers *via1*

© 2020 All Rights Reserved.

to *via10*. Of these, only *via1* to *via3* are displayed in the *Layers* panel if the *Used* check box is selected. At this point, if you search for *via5* by using *Find*, the search will be unsuccessful.

Corresponding SKILL Functions: `pteFindNext`, `pteFindPrev`

Search mode set to *Filter*

Search mode set to *Find*

Click the *Search History* arrow (the *Filter*/*Find* drop-down list arrow) to display the search history for the current session. Click the *Clear* button to clear the contents of the *Filter*/*Find* field.

Corresponding SKILL Function: `pteClearSearchHistory`

The *Filter* and *Find* toolbars are also available in the *Objects* and *Grids* panels.

Search String

Type a search string in the *Filter* or *Find* field in the *Layers* panel. As you type individual characters, you see the information in the panel getting updated. Finally, the search returns all layer-purpose pairs that contain the specified string in any of the displayed columns.

Note: The search operation includes only the columns that are displayed in the applicable Palette panel.

© 2020 All Rights Reserved.

The advanced search options determine the results that a search operation returns. By default, the search operation is case-insensitive and looks for values that begin with the specified string (prefix). For more information, see Advanced Search.

If a search string contains a space, it is interpreted as two strings. In such cases, the search operation searches one by one for each string in all the displayed columns and returns layer-purpose pairs that have a column value starting with one of the two strings AND another column value starting with the other string. This too is the default behavior.

Corresponding SKILL Function: `pteSetSearchText`

Advanced Search

You can use the advanced search options to further refine search results. To access the advanced search options, click the *Advanced* arrow next to the *Filter* field.

The following advanced search options are available for both *Filter* and *Find* toolbars:

- *Find With*

© 2020 All Rights Reserved.

The options listed in the *Find With* section are used to specify default operators for a search operation: AND (*All Of The Words*), OR (*Any Of The Words*), EXACTLY (*The Exact Phrase*), and NOT (*None Of The Words*).

Corresponding SKILL Function: `pteSetSearchOperator`

- *Match Word*

  The options listed in the *Match Word* section enable you to further refine the search by specifying whether a partial or an exact match is required. The options available are: *Substring*, *Prefix, Exactly*, and *Suffix*.

  Corresponding SKILL Function: `pteSetSearchMatchType`

- *Using Case*

  The options listed in the *Using Case section* let you specify whether the search operation is case-sensitive *(Sensitive Match)*, or whether the casing of the specified search string can be ignored (*Insensitive Match)*.

  For example, to display search results that include string `"M1"`, and not `"m1"`, ensure that the *Sensitive Match* option is selected.

  Corresponding SKILL Function: `pteSetSearchMatchCase`

© 2020 All Rights Reserved.

Change the Current Active Layer

The layer-purpose pair displayed in the *Active Layer* toolbar is the one that is active. This layer-purpose pair is displayed with a black border around it in the layer-purpose pair list. To set another layer-purpose pair as active, click a layer-purpose pair listed in the *Layers* panel.

This is the current active layer.

Click the layer that you want to set as active.

The layer you choose becomes the new active layer.

Note: The active layer can now be made invisible/invalid with a single click. When you make the active layer invisible/invalid, the next appropriate layer is automatically set as the active layer.

Change Layer Appearance

The appearance of layer-purpose pairs in the layout and in the panels is defined by the technology file and the `display.drf` file. The `display.drf` file defines the display resources, including colors, stipples, line styles, and packets. Each packet is a named combination of stipple, line style, fill color, and outline color. Packets are associated with one or more layer-purpose pairs in the technology file.

© 2020
All Rights Reserved.

To change a layer-purpose pair definition in the `display.drf` file, select the *Edit Display Resources* command from the Palette menu. The Display Resource Editor form is displayed.

You can change the color, stipple, and line style of the layer-purpose pair. For more information, see Display Resource Editor.

© 2020 All Rights Reserved.

Load LSW Information

To read the LSW information file, you can load it to the *Layers* panel.

1. From the Palette menu, choose *Load – LSW Info*. The Load LSW Info form is displayed.

2. Select the file that you want to load.

   Specify the path if you want to load a file stored outside your current directory.

3. Click *Load*.

   The visibility and selectability settings defined in the LSW information file for layer-purpose pairs and objects are loaded to the current display context.

Corresponding SKILL Function: `pteLoadLSWInfo`

© 2020 All Rights Reserved.

Save LSW Information

To save LSW information:

1. From the Palette menu, choose *Save – LSW Info*. The Save LSW Info form is displayed.

2. Select the file that you want to save.

   Specify a file name if you want to save a file with a different name.

3. Click *Save*.

   The visibility and selectability settings from the current display context are saved in the specified LSW information file.

Corresponding SKILL Function: `pteSaveLSWInfo`

© 2020 All Rights Reserved.

Create a Layer Set

To create a layer set:

1. From the Palette menu, choose *Layer Set – Save As*. The Save As Layer Set form appears, as shown in the following figure.

2. From the *Layer Set repository* list, select the repository (directory) in which you want to save the layer set file. For example, if you select `./.cadence`, the file is saved at the following location:

   `./.cadence/dfII/layerSets/<technology library name>`

   Note: The Palette includes layer sets defined in several `.cadence` directories, for example, the present working directory, the home directory, and the shared directory. The list of candidate directories and their order of precedence is defined in the `setup.loc` file. When displayed in the *Layer Set Manager* and the *Layer Set* list, a suffix comprising a number enclosed in parentheses is added to the layer set name to indicate where a layer set is defined. This helps avoid ambiguity if a layer set is saved with the same name in several directories. The order of precedence specified in `setup.loc` determines the value of the suffix. An exception to this rule are the layer sets stored in the `.cadence` directory listed first in the order of precedence. These layer sets are displayed without the suffix (1).

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

3. In the *Layer Set name* field, specify a layer set name.

   Note: If you specify the layer set name as `default`, this layer set is set as the active layer set, instead of the *All Layers* layer set, when a cellview is first opened. Palette searches for the *default* layer set in the layer set repositories in the order specified in the `setup.loc` file. If the `setup.loc` file is unavailable, Palette searches for the *default* layer set in the `.cadence` repository located in the present working directory. When the first layer set named *default* is found, the search stops.

© 2020 All Rights Reserved.

4. Select the *Synchronized* check box if you want to keep the layer set synchronized with the base layer set. As a result, any updates made to the base layer set are automatically reflected in your personalized layer set when you reload your layer set or start a new Virtuoso session.

   The following figure shows the `.layerSet` file for a synchronized layer set. Any layer-purpose pairs that you add or remove locally are listed in this file.

   Corresponding SKILL Function: `pteSaveAsSynchronizedLayerSet`

5. Click *OK*.

   A layer set file is created with the specified name (`<layersetname>.layerSet`) in the selected layer set repository. The newly created layer set is automatically set as active and is displayed in the *Layer Set* list and the *Layer Set Manager*.

Corresponding SKILL Function: `pteSaveAsLayerSet`

© 2020 All Rights Reserved.

A sample layer set file is shown below.

```
default.layerSet
# Layer Set File With Attributes Version 1.1
ref drawing t t
prBoundary boundary t t
DNW drawing t t
NW drawing t t
PSUB2 drawing t t
OD drawing t t
OD_25 drawing t t
NP drawing t t
PP drawing t t
PO drawing t t
CO drawing t t
M1 drawing t t
VIA1 drawing t t
M2 drawing t t
VIA2 drawing t t
M3 drawing t t
VIA3 drawing t t
M4 drawing t t
VIA4 drawing t t
M5 drawing t t
VIA5 drawing t t
M6 drawing t t
VIA6 drawing t t
M7 drawing t t
VIA7 drawing t t
M8 drawing t t
RV drawing t t
AP drawing t t
UBM drawing t t
```

Delete a Layer Set

To delete a layer set:

1. From the Palette menu, choose *Layer Set – Delete*. The *D*elete Layer Set form appears.

2. The *Layer Set repository* field displays the name of the repository from which the layer set will be deleted. This is a non-editable field.

© 2020
All Rights Reserved.

3. From the *Layer Set name* drop-down list, select the layer set that you want to delete.
4. Click *OK*.

   The layer set is deleted from the layer set repository and is no longer displayed in the *Layer Set* list and *Layer Set Manager*.

Corresponding SKILL Function: `pteDeleteLayerSet`

Import a Layer Set

To import a layer set:

1. From the Palette menu, choose *Layer Set – Import*. The Import Layer Set form appears.

2. Select the layer set file that you want to import.

   Specify the path if you want to import a layer set file that is stored outside your current directory.

   The selected file name is automatically displayed in the *File name* field in the *Origin* group box, as well as in the *Layer Set name* field in the *Destination* group box.

© 2020 All Rights Reserved.

3. From the *Layer Set repository* drop-down list, select the layer set repository in which you want to save the layer set file. The layer set file is saved in the selected repository with the name that appears in the *Layer Set name* field.
4. Click *Import*.

   The imported layer set is added to the *Layer Set* list in the *Layers* panel.

Corresponding SKILL Function: `pteImportLayerSet`

© 2020 All Rights Reserved.

Export a Layer Set

To export a layer set:

1. From the Palette menu, choose *Layer Set – Export*. The Export Layer Set form appears.

2. In the *Origin* group box, the *Layer Set repository* field displays the repository in which the layer set is stored. This is a non-editable field.
3. From the *Layer Set name* drop-down list, select the layer set that you want to export.
4. In the *File name* field of the *Destination* group box, specify the name with which you want to save the layer set.

Important

Specify `.layerSet` as the file extension along with the name that you specify for the layer set.

5. Click *Export*.

   A new layer set is created in your present working directory.

© 2020 All Rights Reserved.

Corresponding SKILL Function: `pteExportLayerSet`

Set Palette Options

To set Palette local and global options, choose *Options* from the Palette menu. The Options form is displayed.

The Options form includes the following fields:

- *Keep windows synchronized*: Initializes all new windows to synchronized state.
- *Search Mode*: Lets you switch between *Filter* and *Find* search modes. The toolbar displayed in the Palette is accordingly updated. The default *Search Mode* is *Filter*. For information about how to use the two modes, see Search for a Layer.

© 2020 All Rights Reserved.

- *Multi-select mode*: Lets you select multiple layers at one time. Only one layer—the one selected first—is set as active. For more information, see Select Multiple Layers.
- *Auto Redraw*: Sets the scope of the auto redraw feature. You can choose one of the following options:
  - *Off*: Turns off the auto redraw feature. As a result, manual redraw is needed to update all windows.
  - *Current Window*: Updates in real time the current window canvas whenever a change is made in a Palette panel.
  - *All Windows*: Updates in real time all synchronized window canvases whenever a change is made in a Palette panel.
- *Layer Set Activation*: Specifies how visibility and selectability status of layer-purpose pairs is affected when a layer set is made active.
  - *Filter only*: Displays all layer-purpose pairs present in the selected layer sets. The visibility and selectability status of layer-purpose pairs remains unchanged.
  - *Filter and apply visibility, selectability*: Displays all layer-purpose pairs present in the selected layer sets. The visibility and selectability status of layer-purpose pairs is reset to that defined in the layer set files.
  - *Force visibility and selectability only*: Displays all layer-purpose pairs present in the selected layer sets. The visibility and selectability of all layer-purpose pairs is turned on.
- *Scope of 'All Layers' Layer Set*: By default, only the layer-purpose pairs defined in the `leLswLayers` section of the technology file are displayed in the *Layers* panel.

  Choose one of the following options:
  - *LSW layers*: Displays layer-purpose pairs defined in the `leLswLayers` section of the technology file. If this section is not available, all layer-purpose pairs defined in the technology file are displayed.
  - *Techfile layers*: Displays all layer-purpose pairs defined in the technology file, even if the `leLswLayers` section is available.
- *Header Column Click*: Defines the action to be performed when you click the Visibility (*V*), Selectability (*S*), Membership (*m*), and Validity (*v*) column headers in the *Layers* panel.
  - *Toggle on/off*: Turns on or off the Visibility (*V*), Selectability (*S*), Membership (*m*), and Validity (*v*) status.

© 2020 All Rights Reserved.

- *Sort*: Sorts layer-purpose pairs based on the Visibility (*V*), Selectability (*S*), Membership (*m*), and Validity (*v*) status.

- *Scope of Used Layers in EIP*: Lets you specify the layers to display in the *Layers* panel when EIP mode is active. If different technology files are associated with the cellviews at different hierarchical levels in the design, only the layers defined in the technology file associated with the cellview being edited in place are displayed in the *Layers* panel.
  - *Current to bottom*: Displays layers used in the design at the current hierarchy level and below. This is the default option.
  - *Top to bottom*: Displays all layers used in the design.
- *Depth for Visibility*: Determines the number of layer-purpose pairs for which the visibility attribute is affected when you perform any of the following actions:
  - Hold down the `Ctrl` key and click a layer-purpose pair with the middle mouse button. This turns on or off the visibility of the layer-purpose pair that is clicked and the *n* layer-purpose pairs listed both above and below it, where *n* is equal to the visibility depth value, as shown below.
  - Hold down the `Ctrl` and `Shift` keys and click a layer-purpose pair with the middle mouse button. This sets as visible the layer-purpose pair that is clicked and the *n*

© 2020 All Rights Reserved.

layer-purpose pairs listed both above and below it, where *n* is equal to the visibility depth value. Additionally, this sets as active the layer-purpose pair that is clicked.

Hold down the `Ctrl` key and click *M3* with the middle mouse button.

The result shown is for visibility depth equal to 1 (default). The selectability settings remain unchanged.

The result shown is for visibility depth equal to 2. The selectability settings remain unchanged.

Note: This feature is available only when the *Selection Mode* is set to *Single Selection*.

- *Depth for Selectability*: Determines the number of layer-purpose pairs for which the selectability attribute is affected when you perform any of the following actions:
  - Hold down the `Ctrl` key and right-click a layer-purpose pair. This turns on or off the selectability of the layer-purpose pair that is clicked and the *n* layer-purpose pairs listed both above and below it, where *n* is equal to the selectability depth value, as shown below.

© 2020 All Rights Reserved.

- Hold down the `Ctrl` and `Shift` keys and right-click a layer-purpose pair. This turns on the selectability of the layer-purpose pair that is clicked and the *n* layer-purpose pairs listed both above and below it, where *n* is equal to the selectability depth value.

Hold down the `Ctrl` key and right-click *M3*.

The result shown is for selectability depth set to 1 (default).

The result shown is for selectability depth set to 2.

Note: This feature is available only when the *Selection Mode* is set to *Single Selection*.

- *Sync visibility/selectability by parent*: If selected, propagates changes in the visibility and selectability status of parent objects and grids to child objects and grids. For more information, see Determining Visibility and Selectability in Objects and Grids Panels.

- *Shape visibility by color visibility*: If selected, colored shapes become invisible if you deselect the Visibility (*V*) check box corresponding to the layer-purpose pairs on which they are drawn. If deselected, only the colored outline is hidden; the shapes remain visible. For more information, see MPT Support.

© 2020 All Rights Reserved.

- *Color mixed sort*: If selected, enables sorting on attributes other than color. For more information, see MPT Support.
- *Compact MPT mode*: If selected, hides the Color (*C*) column and displays on the *Active* (*Layer*) toolbar a set of color, lock state, and visibility controls. For more information, see MPT Support.
- *Inverse color scheme*: If selected, inverses the color scheme of the Palette panels.
- *Movable tools*: If selected, three vertical lines appear to the left of the toolbar name. When you point to these vertical lines, the pointer becomes a four-headed arrow. Drag the pointer to move a toolbar to a different location within the Palette panel to which it belongs.

For more information, see Palette Options Form.

Select Multiple Layers

To select multiple layers:

1. Choose *Options* from the Palette context menu. The Options form is displayed.

© 2020 All Rights Reserved.

2. Select the *Multi-select mode* check box and click *OK*.

© 2020 All Rights Reserved.

3. Hold down the `Ctrl` key and click the layer-purpose pairs that you want to select. To select a set of adjacent layer-purpose pairs, hold down the `Shift` key and click the first and last layer-purpose pairs in the set.

4. Right-click the selection and use the *Visibility*, *Selectability*, *Layer Set Member*, *Validity*, *Routing Direction, and Stipple* options to set the corresponding attributes for the selected layers. The *Layer Set Member* and *Validity* options are available only when the *Member* or *Validity* check boxes are selected on the *Edit* toolbar. For more information, see Edit. The *Routing Direction* option is available only for routing layers.

To remove a layer from a multiple-layer selection, hold down the `Ctrl` key and click that layer.

© 2020 All Rights Reserved.

MPT Support

The Palette MPT feature lets you draw shapes, wires, and pins with layer, color, and lock state information. Shapes that you can draw with color include rectangles, polygons, paths, circles, ellipses, and donuts.

Note: You cannot draw pins with color in *Auto* mode.

Enabling MPT Support

To enable the Palette MPT feature, choose *MPT Support* from the Palette context menu. This displays the *MPT* check box on the *Scope* toolbar and the Color (*C*) column in the *Layers* panel, indicating that MPT support is now enabled. As a result, all layer-purpose pairs with color are displayed in the *Layers* panel, grouped by their master layer-purpose pair, as shown below.

Note: At this stage, all layer-purpose pairs are displayed in the *Layers* panel, including those with no color (listed at the end). To display only the layer-purpose pairs with color, select the *MPT* check box.

© 2020
All Rights Reserved.

Each color and lock state combination for a layer-purpose pair is listed as a separate entry. When you set a layer-purpose pair as active, the color and lock state is automatically set for the shapes that you draw on that layer.

Note: For Move and Copy commands, the reference pathseg and via display the color of the WSP track during drag and command commit.

Corresponding SKILL Functions:

- `pteMPTSupportMode`
- `pteSetMPTSupportMode`
- `pteGetMPTSupportMode`
- `pteCloseMPTSupportMode`

When you open a new layout window, the pteMPTMode environment variable determines if *MPT Support* is enabled by default. The pteMPTScope environment variable determines if all layer-purpose pairs or only the layer-purpose pairs with color are displayed. The `pteMPTScope` environment variable is evaluated only if the `pteMPTMode` environment variable is set to `t`. The default value of both these environment variables is `nil`.

Note: For the `pteMPTMode` and `pteMPTScope` environment variables to take effect in a new layout window, they must be set before you open the window.

To hide the *MPT* check box displayed on the *Scope* toolbar, set the pteShowMPTScope environment variable to `nil`.

© 2020 All Rights Reserved.

Enabling Compact MPT Mode

You can use any of the following methods to enable compact mode:

- Set the pteCompactMPT environment variable to `t` at the start of a Virtuoso session to ensure that all layout windows have *MPT Support* set to compact mode by default.
- Use the `pteSetOptionString` SKILL function. This propagates a change to the `pteCompactMPT` environment variable in all open layout windows in real time. For example, to enable compact mode, enter:

  ```
  pteSetOptionString("pteCompactMPT" "t")
  ```

- Select the *Compact MPT mode* check box in the Palette Options form in an open layout window.

Enabling compact mode hides the Color (*C*) column in the *Layers* panel and displays on the *Active* (*Layer*) toolbar the *Color* and *Lock color* options that you can use to select the color and lock state combination for a layer-purpose pair with color, as shown below.

© 2020
All Rights Reserved.

When you enable compact mode, *Color* is set to `mask1Color` and *Lock state* is set to `unlocked` for an active layer-purpose pair with color, as shown above.

The *Color* and *Lock color* options are disabled for layer-purpose pairs without color. The *Color* for such layer-purpose pairs is set to `g` (`grayColor`), as shown below.

To display the two options that help control the visibility of colored shapes based on their lock status—*Show Unlocked Colors* and *Show Locked Colors*—set the pteShowColorControlsForMPT environment variable to `t`.

© 2020
All Rights Reserved.

Drawing Shapes with Color

1. Enable MPT support.
2. (Optional) Select the *MPT* check box.
3. Set as active a layer-purpose pair with the required color and lock status.

   Notice that the icons listed in the Color (*C*) column for each layer-purpose pair provide color as well as lock information. The swatch displayed in the first column indicates how a shape appears in the canvas.

4. Draw a shape. The color and lock state are automatically set for the shape.

5. Use the Visibility (*V*) and Selectability (*S*) check boxes corresponding to a layer-purpose pair to control the visibility and selectability of shapes drawn on that layer.

© 2020
All Rights Reserved.

Controlling the Display of Shapes

Shapes drawn on a layer become invisible if you deselect the Visibility (*V*) check box corresponding to that layer. However, if the shapes are drawn on a layer that supports color, their visibility is controlled by the *Shape visibility by color visibility* check box available in the Palette Options form:

- If this check box is selected, colored shapes become invisible if you deselect the Visibility (*V*) check box corresponding to the layers on which they are drawn.
- If this check box is deselected, only the colored outline is hidden; the shapes remain visible, as shown below.

© 2020
All Rights Reserved.

Controlling the Color Information Displayed in Palette

You can use the `explicitColoredPurposes` environment variable to specify a list of purposes for which you want color information to be displayed in the Palette when MPT support is turned on. When this environment variable is set to `""`, the default value, color information is displayed in the Palette for purposes `drawing` and `pin`, any purposes specified using the `coloredPurposeTypes` environment variable, and all their child purposes, as shown below.

© 2020
All Rights Reserved.

If you now close the session and reset this environment variable to, say, "`vdd`", and then reopen the session, color information is listed in the Palette only for layers with purpose "`vdd`", as shown below.

```
envSetVal("mpt" "explicitColoredPurposes" 'string "vdd")
```

Important

The Palette is not updated if the `explicitColoredPurposes` environment variable is reset during a session. Additionally, when you specify a purpose by using the `explicitColoredPurposes` environment variable, its child purposes are not included automatically. Therefore, you need to explicitly specify all purposes for which you want color information to be displayed in the Palette.

If shapes with color created in previous sessions are excluded in a new session by using this method, the shapes are no longer colorable. The visibility and selectability of such shapes can be controlled through the base layer-purpose pair (`grayColor`), which is listed for all excluded purposes, as shown in the example below.

© 2020 All Rights Reserved.

When color information for `Metal5` is excluded, the visibility and selectability of all `Metal5` shapes, regardless of color-lock status, is controlled through the base layer-purpose pair.

© 2020 All Rights Reserved.

When color information is available for `Metal5`, visibility and selectability can be controlled individually for each color-lock pair.

Performing a Search Based on Color and Lock Information

Important

The search functions listed below are available only when the Color (*C*) column is displayed in the *Layers* panel.

In the search field, with mode set to *Filter*, type:

- `mask`*N* to search for layer-purpose pairs with `mask`*N*`Color`, where *N* = 1, 2, …
- `_lock` to search for layer-purpose pairs with color set to `locked`
- `_unlock` to search for layer-purpose pairs with color set to `unlocked`
- `mask`*N* `_lock` or `_lock mask`*N* to search for layer-purpose pairs with `mask`*N*`Color` and color state `locked`
- `mask`*N* `_unlock` or `_unlock mask`*N* to search for layer-purpose pairs with `mask`*N*`Color` and color state `unlocked`

© 2020 All Rights Reserved.

All layer-purpose pairs that satisfy the search criteria are listed in the *Layers* panel. You can toggle the visibility and selectability of all filtered layers together by clicking the Visibility (*V*) and Selectability (*S*) column headers.

Saving Filtered Layer-Purpose Pairs to Layer Sets

After filtering layer-purpose pairs based on color and lock state, you can save them to a layer set file. This helps limit the number of layer-purpose pairs displayed in the *Layers* panel. The color and lock state is saved in the `.layerset` file along with the other attributes, as shown below. For more information, see Create a Layer Set.

```
mask1_lock.layerSet
# Layer Set File With Attributes Version 1.2
Metal1 drawing t t mask1Color locked
Metal2 drawing t t mask1Color locked
Metal2 localWSP t t mask1Color locked
Metal3 drawing t t mask1Color locked
Metal3 localWSP t t mask1Color locked
Metal4 drawing t t mask1Color locked
Metal4 localWSP t t mask1Color locked
Metal5 drawing t t mask1Color locked
Metal5 localWSP t t mask1Color locked
V0 drawing t t mask1Color locked
Via1 drawing t t mask1Color locked
Via2 drawing t t mask1Color locked
Via3 drawing t t mask1Color locked
Via4 drawing t t mask1Color locked
```


© 2020 All Rights Reserved.

The newly created layer set is automatically set as the current layer set and is displayed in the *Layer Set* list and the *Layer Set Manager*, as shown below:

You can use the buttons listed in the *Vis* and *Sel* columns to toggle with a single click the visibility and selectability of all layer-purpose pairs in the layer set. For more information, see Layer Set Manager.

© 2020 All Rights Reserved.

Using View By

Important

The *View By* option is not listed on the column header menu when MPT scope is enabled; layer-purpose pairs are listed in the *Layers* panel grouped by layer-purpose name.

When MPT scope is disabled, if you choose *View By – Color*, only the layer-purpose pairs without color are listed in the *Layers* panel, grouped by the color name (`grayColor`), as shown below:

If you choose an option other than *Color* on the *View By* menu, color and lock state information for layer-purpose pairs is not displayed. For information about other available options, see View By.

© 2020 All Rights Reserved.

Specifying the Sort Order

When you turn on MPT support, the layer-purpose pairs displayed in the Palette are by default sorted by color. As a result, layer-purpose pairs are organized into two distinct groups:

- Layer-purpose pairs with color
- Layer-purpose pairs without color

Layer-purpose pairs in each group are sorted within the group as follows:

- All layer-purpose pairs with color, grouped by their master layer-purpose pair, are sorted in alphabetical order.
- All layer-purpose pairs without color are sorted in the order specified by the selected attribute. For more information about the attributes on which you can sort layer-purpose pairs, see Sort By.

Sorting Together Colored and Uncolored Layer-Purpose Pairs

To sort all layer-purpose pairs, colored and uncolored, together as a single group, select the *Color mixed sort* check box in the Palette Options form or set the pteColorMixedSort environment variable to `t`. As a result, layer-purpose pairs when sorted are listed in the order defined by the selected attribute, such as swatch, layer name, purpose, and priority.

Sorted on swatch

Sorted on layer name

Sorted on purpose

© 2020 All Rights Reserved.

Layer-purpose pairs with color stay grouped by their master layer-purpose pair because each layer-purpose pair in a group has the same priority irrespective of color, as shown below:

Sorting within a Master Layer-Purpose Pair Group

Layer-purpose pairs within each master layer-purpose pair group are by default listed as follows: `grayColor`, `mask1Color locked`, `mask1Color unlocked`, `mask2Color locked`, `mask2Color unlocked`, and so on. To specify a different sort order, use the pteColoredLppSortOrder environment variable.

For example, if you set this environment variable as shown below, the layer-purpose pairs within each master layer-purpose pair group are listed as follows: `mask2Color locked`, `mask1Color locked`, `mask1Color unlocked`, `mask2Color unlocked`, `grayColor`.

```
envSetVal("layout" "pteColoredLppSortOrder" 'string "m2l m1l m1u m2u g")
```

© 2020 All Rights Reserved.

The layer-purpose pairs that are not included in the argument list follow the default sort order. For example, if you set the environment variable as shown below, the layer-purpose pairs within each master layer-purpose pair group are listed as follows: `mask1Color locked`, `mask2Color locked`, `grayColor`, `mask1Color unlocked`, `mask2Color unlocked`.

```
envSetVal("layout" "pteColoredLppSortOrder" 'string "m1l m2l")
```


© 2020 All Rights Reserved.

Objects Panel

The *Objects* panel displays objects such as instances, pins, and vias. It is a dockable panel that replaces the LSW *Object* pane.

The *Objects* panel is displayed below the *Layers* panel. By default, objects in the *Objects* panel are listed in the order shown in the following figure.

Note: The object *Row Region* shown in the figure above is an "Advanced Node Layout EAD Only" feature. For more information, see Layout Support for Row Regions in *Virtuoso® Placer User Guide*.

© 2020 All Rights Reserved.

Note: In Layout EXL, the *Objects* panel displays the *Virtual Hierarchy* object as well.

Displaying and Hiding Objects Panel

Use one of the following methods to display the *Objects* panel:

- Right-click any panel title bar and choose *Objects*.
- Use the `pteMapWindow` function to specify the name of the Palette panel to be displayed.

Use one of the following methods to hide a Palette panel:

- Right-click any panel title bar and choose *Objects*.

© 2020 All Rights Reserved.

- Use the `pteUnmapWindow` function to specify the name of the Palette panel to be hidden.

User Interface

The *Objects* panel user interface has the following components:

- Title Bar Buttons
- Toolbars
- Columns
- Column Header Context Menu

Title Bar Buttons

The *Objects* panel title bar has the following buttons:

- *Float/Dock*

  Undocks the *Objects* panel. Click the button again to dock the panel.

- *Hide*

  Hides the *Objects* panel.

Toolbars

By default, the *Objects* panel does not display toolbars. To display a toolbar, right-click an object and choose *Show Tools,* followed by a toolbar name. For more information about toolbars, see Layers Panel.

Columns

The following columns are displayed in the *Objects* panel:

- *Objects*

  Displays objects listed in a hierarchical tree structure.

- *V* (Visibility)

  Enables you to control the visibility of objects. To make an object visible, select the corresponding check box in the *V* (Visibility) column.

© 2020 All Rights Reserved.

Note: It is not possible to change the visibility of some objects such as fluid guard rings, fluid shapes, and soft blocks. Therefore, the *V* (Visibility) check box is unavailable for these objects.

- *S (Selectability)*

  Enables you to control the selection of objects. To select an object, select the corresponding check box in the *S* (Selectability) column. An invisible object cannot be selected.

  Note: In the *Objects* panel, the behavior of instance, via, and fig group objects is different from that of other objects because these are second-level objects. These objects can be selected even if they are invisible.

You cannot resize the columns in the *Objects* panel.

Instance selection and visibility is controlled through the *Objects* panel. Pin visibility and selectability is determined by the visibility and selectability status of the layer-purpose pair of the corresponding pin figure. You can further control the selection of pins by using the pinSelectionModeenvironment variable.

Blockage selection and visibility is controlled by the layer-purpose pair on which the blockage is created. You can further control the selection of blockages by using the blockageAttributeMode environment variable.

Column Header Context Menu

When you right-click a column header in the *Objects* panel, a context-sensitive menu is displayed, as shown in the following figure.

© 2020 All Rights Reserved.

Columns Menu

The *Columns* menu can be used to selectively hide or display columns. The *Hide All* and *Show All* commands hide or display the *Visibility* and *Selectability* columns. The *Objects* column is not hidden from view when you choose the *Hide All* command.

Sort By Menu

You can use the *Sort By* menu to sort objects by object name, visibility, or selectability.

- *Objects*

  Sorts objects in the alphabetical order by name.

- *Visibility*

  Sorts objects by visibility. The objects that cannot be made visible are listed first. These are followed by the objects for which the visibility check box is deselected. The objects for which the visibility check box is selected are listed last.

- *Selectability*

© 2020 All Rights Reserved.

Sorts objects by selectability. The objects for which the selectability check box is deselected are listed first, followed by the objects for which the selectability check box is selected.

- *Default*

  Sorts objects in the default order. The default order is as follows: *Instances*, *Pins*, *Vias*, *Label*, *Shapes*, *Mosaic*, *Rulers*, *Markers*, *Fluid Guardring*, *Fluid Shapes*, *Fig Groups*, *Boundaries*, *Soft Blocks*, *Blockages*, *Symmetric Axes*, and *Rows*.

- *Reverse*

  Reverses the order in which the objects are currently displayed.

Hide This Column

This command enables you to hide a column displayed in the *Objects* panel. To hide a column, right-click the column header and choose *Hide This Column*.

© 2020 All Rights Reserved.

Grids Panel

The *Grids* panel displays *Tracks* and *Grids*. It is a dockable panel that replaces the LSW *Grid* pane.

The *Grids* panel is displayed below the *Layers* panel. If both *Objects* and *Grids* panels are displayed, they appear on separate tabs and you need to click the *Grids* tab to view its contents.

Displaying and Hiding Grids Panel

Use one of the following methods to display the *Grids* panel:

- Right-click any panel title bar and choose *Grids*.
- Use the `pteMapWindow` function to specify the name of the Palette panel to be displayed.

Use one of the following methods to hide the *Grids* panel:

- Right-click any panel title bar and choose *Grids*.
- Use the `pteUnmapWindow` function to specify the name of the Palette panel to be hidden.

User Interface

The *Grids* panel user interface has the following components:

- Title Bar Buttons
- Toolbars
- Columns

© 2020 All Rights Reserved.

- Column Header Context Menu

Title Bar Buttons

The *Grids* panel title bar has the following buttons:

- *Float/Dock*

  Undocks the *Grids* panel. Click the button again to dock the panel.

- *Hide*

  Hides the *Grids* panel.

Toolbars

By default, the *Grids* panel does not display toolbars. To display a toolbar, right-click an element and choose *Show Tools,* followed by a toolbar name. For more information about toolbars, see Layers Panel.

Columns

The following columns are displayed in the *Grids* panel:

- *Grids*

  Displays elements in a hierarchical tree structure.

- *V* (Visibility)

  Enables you to control the visibility of elements. To make an element visible, select the corresponding check box in the *V* (Visibility) column.

© 2020 All Rights Reserved.

Column Header Context Menu

When you right-click a column header in the *Grids* panel, a context-sensitive menu is displayed, as shown in the following figure.

Columns Menu

The *Columns* menu can be used to selectively hide or display columns. The *Hide All* and *Show All* commands hide or display the *V* (Visibility) column. The *Grids* column is not hidden from view when you choose the *Hide All* command.

© 2020
All Rights Reserved.

Sort By Menu

You can use the *Sort By* menu to sort the elements displayed in the *Grids* panel by element name and visibility.

- *Grids*

  Sorts elements in the alphabetical order by name.

- *Visibility*

  Sorts elements by visibility. The elements for which the visibility check box is deselected are listed first, followed by the elements for which the visibility check box is selected.

- *Default*

  Sorts elements in the default order. The default order is as follows: *Tracks* and *Grids*.

- *Reverse*

  Reverses the order in which the elements are currently displayed.

Hide This Column

This command enables you to hide a column displayed in the *Grids* panel. To hide a column, right-click the column header and choose *Hide This Column*.

© 2020 All Rights Reserved.

Determining Visibility and Selectability in Objects and Grids Panels

You can set the selectability and visibility of parent and child items in the *Objects* and *Grids* panels in the following two modes:

- Synchronized mode

  This is the default mode. In this mode, the parent and child items are in a synchronized state. This means that when you reset the visibility or selectability of a parent by clicking the corresponding check box, the visibility and selectability of the child items is set to coincide with that of the parent. If you now reset the visibility or selectability of a child, the parent is set to a partial check mode to indicate that one or more child items override the visibility or selectability settings of the parent, as shown in the following figure.

- Desynchronized mode

  In this mode, the parent and child items are in a desynchronized state. This means that the visibility and selectability of a child is not reset when you reset the visibility or selectability of the parent.

© 2020 All Rights Reserved.

To understand this better, consider the *Shapes* object (parent) that has several child objects. Currently, only the *Circle/Ellipse* child object is set to visible. The other child objects are set to invisible.

If you now turn off the visibility of the parent by deselecting the corresponding Visibility (*V*) check box, the visibility and selectability states of the child objects are not affected. As a result, the Visibility (*V*) check box for the *Circle/Ellipse* child object stays selected, as shown in the following figure, but the child object would no longer be visible in the canvas.

If you next turn on the visibility of the parent, the visibility and selectability states of the child objects again remain unchanged, but the *Circle/Ellipse* child object becomes visible in the canvas.

Note: A change in the visibility and selectability state of a child affects the visibility and selectability state of the parent.

© 2020 All Rights Reserved.

Switching between Modes

You can use one of the following methods to switch between the two modes:

- Use the pteApplyParentState environment variable.

  The syntax for the environment variable is as follows:

  ```
  layout pteApplyParentState 'boolean t/nil
  ```

  By default, the environment variable is set to `t`, which is the synchronized mode. To switch to the desynchronized mode, set the environment variable to `nil`.

  You can set the `pteApplyParentState` environment variable in the `.cdsenv` file before you run Virtuoso or by using `envSetVal` in CIW.

- Use the `pteSetOptionString` SKILL function.

- Use the CIW to run the following commands:

  ```
  pteOpenForm("Options")
  pteOptionForm->objectGridSynchro->value = t
  pteOptionForm->OK->value = t
  ```

- Use the Options form.

  Right-click an item in the *Objects* or *Grids* panel and choose *Options*. The Options form is displayed. As shown in the following figure, the *Sync visibility/selectability by parent* check box is selected by default. This implies that the parent and child items are

© 2020 All Rights Reserved.

by default in the synchronized state. To switch to the desynchronized state, deselect this check box and click *Apply* or *OK*.

© 2020 All Rights Reserved.

4

Using Design Tools

This chapter contains these topics:

- Using Find/Replace
- Using Area and Density Calculator
- Using Rulers
- Re-mastering Instance Views
- Creating Pins from Labels
- Using the Tap Command
- Changing Objects on a Layer
- Generating Abstracts
- Working with Markers
- Working with Markers
- Pad Opening Info Command

© 2020 All Rights Reserved.

Using Find/Replace

The command lets you search for objects with specific attributes or property values. See the Objects Criteria Table for a list of conditions and logic to use when searching for objects.

Support for Objects	
ROD objects	You can search for ROD names on labels, paths, polygons, rectangles, any shape and text display objects.
Blockages	■ You can search for blockages by blockage type, layer name, and owner name. ■ You can use the Zoom to Figure and Previous and Next buttons to navigate all the blockages. ■ You cannot replace any of the values based on the search results.
Boundaries	■ You can search for PR and snap boundary by object type. ■ You can search for area boundary by boundary name and property. ■ You can search for cluster boundaries by boundary name, cluster name, and property. ■ You cannot replace any of the values based on the search results.

This topic covers the following:

- Specifying the Search Criteria
- Finding Objects
- Finding Objects in Instances
- Finding Instances
- Replacing Instances
- Finding and Replacing Text of Labels

Specifying the Search Criteria

In the text field where you specify the search text, you can make a text entry or choose from a cyclic field, depending on the type of value you want to search for.

© 2020 All Rights Reserved.

To search for multiple objects, you can use `*` and `.` wild card characters. The character `*` searches for the entry you made, with any number of unspecified characters before, after, and/or between any characters in your entry. The character `.` searches for the entry you made, with any single unspecified character at any position in your entry. For example, to search for all instances that have `PInst` at the beginning of their names, type "`PInst*`". However, if you want to search for all instances that have `Inst` in their names irrespective of the first character, type "`.Inst*`".

Note: The use of the wild card character for multiple instance search returns *all* results that meet the specified criterion, even if you are looking only for a few instances (two or three). So, if you are searching for only `PInst1` and `PInst2` out of all the instances that have `PInst` at the beginning of their names, specifying the search criterion as one of the following does not give the desired results:

- inst name == `PInst1 PInst2`
- inst name == `PInst1, PInst2`
- inst name == `PInst1`
  inst name == `PInst2`

Finding Objects

To find instances,

1. Choose *Tools – Find/Replace* [`Shift-s`].

   The Find/Replace Form appears.

2. In the *Search for* cyclic field, choose the type of object you want to find, and the location in which you want to search in the *Search Hierarchy Range* field.

© 2020 All Rights Reserved.

3. Click *Add Criteria* to add Boolean expressions, which control the search (search criteria).

.

© 2020 All Rights Reserved.

4. Set the search criteria by doing the following:

a. click the first field to choose the type of value to search for.

b. click the Boolean operator field to choose the operator you want: equal (==) or not equal (!=), and other operators depending upon the criteria you select.

c. Type or choose the value you want. (see Specifying the Search Criteria)

Choose the type of object to search for

Choose a boolean operator

Type the value to search for

- ❑ Click *Delete* at the end of the row to delete a row of search criteria.
- ❑ Click *Add Criteria* again to add more search criteria.

5. Click *Apply*.

The layout editor highlights all of the objects it finds matching the specified search criteria and puts them in a search group. The first or current object in the group is highlighted in a different color.

6. Click *Previous* or *Next* to search through the group.

© 2020 All Rights Reserved.

7. Click *Add Select* to select the current object in the search group.
8. Click *Select All* to select all of the highlighted objects.
9. When you are finished selecting objects, click *Close*.

Finding Objects in Instances

To search for objects or instances inside other instances,

1. Choose *Tools – Find/Replace* [`Shift-s`].

   The Find/Replace Form appears.
2. In the *Search for* cyclic field choose the type of object you want to find.
3. Click *Add Criteria* to further define the search.
4. Specify the *Search Hierarchy Range*.
5. Click *Apply*.

   During the search, instances are highlighted as they are found.

Note: If you use *Edit In Place* to edit a cell while the form is open, you must cancel and then restart it. Otherwise, command does not know that you have changed your current edit level.

Finding Instances

To search for instances in the current cellview,

1. Choose *Tools – Find/Replace* [`Shift-s`].

   The Find/Replace Form appears.
2. In the *Search for* cyclic field, choose *instance*.
3. Click *Add Criteria* to add search criteria.
4. Click the first field in the criterion line to search for the instance by one of the following:
   - *cell name*, to search for the name of the master cell
   - *instance name*, to search for the name assigned to the instance as you placed it
5. Type the cell or instance name. (see Specifying the Search Criteria)
6. Click *Add Criteria* to further define the search.

© 2020 All Rights Reserved.

7. Click *Apply*.

   The layout editor highlights all the instances in the cellview that match the criteria you entered.

Replacing Instances

To replace instances of one master cell with instances of a different master cell,

1. Find the required instances by following the steps described in Finding Instances.
2. In the *Replace* cyclic field, choose *cell name*.

© 2020 All Rights Reserved.

3. Type the name of the new master cell.

4. Click *Apply*.

© 2020 All Rights Reserved.

The layout editor highlights all the instances of the master cell.

5. To replace only one instance of the cell, click *Previous* or *Next* until the instance you want is highlighted.
6. To replace one instance of the cell, click the *Replace* button to replace the top cell in the search stack.
7. To replace all instances of the cell, click the *Replace All* button to replace all instances of the highlighted cell.

Note: Similar to the Find option, you can specify multilpe replace criteria using the *Add Criteria* option in the Replace section of the Find/Replace form. This will enable you to replace multiple attributes together.

Finding and Replacing Text of Labels

To search and replace labels in the current cellview,

1. Choose *Tools – Find/Replace* [`Shift-s`].

   The Find/Replace Form appears.
2. In the *Search for* cyclic field, choose *label*.
3. Click *Add Criteria* to add search criteria.
4. The first field in the criterion line shows *text* by default.
5. To search for all the labels in the cellview, type `*` in the text field. You can specify multiple criteria to search for the required labels.
6. Click *Apply*.

   The layout editor highlights all the labels in the cellview.
7. In the *Replace* cyclic field, choose *text.*
8. In the second cyclic field, choose one of the following:
   - *Change text*: Select this option and specify the text to replace the label text of all the labels matching the search criteria.
   - *Add Prefix*: Select this option and specify the text to be prefixed to all the labels matching the search criteria.

© 2020 All Rights Reserved.

- *Add Suffix*: Select this option and specify the text to be added at the end of all the labels matching the search criteria.

9. To replace only one label, click *Previous* or *Next* until the label you want is highlighted and then click *Replace*.

10. To replace all the highlighted labels, click *Replace All*.

    To perform this operation across different hierarchy levels, select *current to bottom* or *current to stop level* or *current to user level* instead of *current cellview* in the *Search Hierarchy Range* field.

© 2020 All Rights Reserved.

Video

You can view the Find/Replace enhancements in the video, *Find/Replace Enhancements*.

© 2020 All Rights Reserved.

Using Area and Density Calculator

The *Area and Density Calculator* command lets you calculate the of the shapes on specified layers within a region. The feature enables you to calculate the area of metal lines, resistor networks, MOS devices placed together. You can also calculate the layer density in a specified region.

Area is defined as the total area of the specified layers within the region that you select. Density is defined as the total area divided by the area of the region the you specify.

To calculate the area and density of a region,

1. Choose *Tools – Area and Density Calculator*.

   This opens the Area and Density Calculator Form.

   You can specify the object, layer, hierarchy range and region for which you want to calculate the in the Area and Density Calculator form.

2. In the *Object* field, select the object for which you want to calculate the area. You can select *All* (all objects) or *Selected* (only selected objects will be considered for area calculation).

3. In the *Layer* field, you select a single layer or multiple layers for area calculation.

© 2020 All Rights Reserved.

4. In the *Hierarchy Range* field, you specify the search depth. You can select the search depth *current cellView*, *current to bottom*, *current to stop level*, or *current to user level*.

5. In the *Region* section, select the region for which you want to calculate the area and density. You can select the *Entire CellView*, *Viewing Area*, *A Selected Boundary*, or *Draw a Rectangle Region* or *Draw a Polygon Region on the Canvas*.

   The area and density calculation report is diplayed.

Video

For an overview on how to use the Area and Density Calculator, see the video *Using the Area and Density Calculator*.

You can also use the SKILL function `leComputeAreaDensity` for this feature.

© 2020 All Rights Reserved.

Using Rulers

The *Create Measurement* command lets you create rulers to measure objects and distances between objects in a cellview.

This topic covers the following:

- Creating Rulers
- Saving Rulers
- Deleting Rulers

Creating Rulers

The *Create Measurement* command creates a ruler object, which lets you measure objects or the distance between objects. You can also use the command to measure distances between two edges or vertices.

You can create transient or persistent rulers in the layout editor. A transient ruler cannot be saved. However, a persistent ruler can be saved. By default, persistent rulers are created in the layout editor. If you want to create transient rulers, deselect *Create Measurement as a savable object* option on the *General* tab of the *Create Measurement* form. To open the Create Slot Form, choose *Tools – Create Measurement,* or press `k` and then press `F3`.

The *Create Measurement* command has the following three modes:

- Single-segment
- Edge
- Multiple

Single-segment

Single-segment is the default measurement mode for the *Create Measurement* command. In this mode, when you add two points on the canvas, the corresponding measurement is displayed on the canvas. If the *Repeat Commands* is on, the *Create Measurement* command continues, else the command completes.

By default, the placement of the measurement is calculated automatically. You can view the placement when you drag the pointer. You have the flexibility to specify the location of the measurement segment, either before adding the second point or after adding the second point.

© 2020 All Rights Reserved.

If you want to specify the location of the measurement segment before the second point is added on the canvas, move the pointer to the desired location, and then press `Ctrl` and click the middle mouse button. The location of the measurement segment is now fixed. You can then select the second point. If you have selected multi-segment measurement mode, the multi-segment measurement segment is drawn at the location where you press `Ctrl` and click the middle mouse button.

If you want to specify the location of the measurement segment after the second point is added on the canvas, press `Ctrl` and click. When you move the pointer, the measurement segment will also move. Click to fix the location of the measurement segment. The *Create Measurement* command now completes.

Note: You can move the measurement segment only if one of the points is snapped to a vertex or an edge.

Edge

The edge measurement mode allows you to click an edge of an object and a measurement is drawn with the two vertices as the two ends. The vertex of the edge closest to the point where you click is considered as the first vertex, and the other edge is considered as the second vertex. When you move the pointer on the canvas, the measurement also moves. You then click the canvas to place the measurement at the desired location. The first click lets you select the edge and the second click lets you place the measurement on the canvas. A double-click lets you create and place the edge measurement at the edge itself.

The edge measurement mode allows you to measure the width of an edge of an object. The first click lets you select the edge and the second click lets you place the measurement on the canvas. A double-click lets you create and place the edge measurement at the edge itself. When pointer snaps to a vertex, the edges connected to the vertex get measured.

Note: Edge measurement works only when smart snapping is on.

Multiple

There are two ways in which you can create multi-segment measurements:

- Click *Multi-segment Ruler* on the *General* tab of the *Create Measurement* form. In the multi-segment mode, a segment is added to the ruler, with every click the canvas. If you double-click or press `Enter`, the last point is added to the ruler and the *Create Measurement* command completes.
- You can also create multi-segment measurement while you are in the single-segment measurement mode. In the single-segment mode, after the first click, press `Shift` and then click to add a point to the measurement. Press `Shift` and click to add subsequent

© 2020 All Rights Reserved.

points to the measurement. If you click or press `Enter`, the last point is added to the ruler and the *Create Measurement* command completes.

Similar to single-segment mode, in multi-segment mode you can place the measurement segment by using the `Ctrl` and click combination.

The table below summarizes the mouse/key bindings that are in effect during the *Create Measurement* command for the three measurement modes, single-segment, edge, and multi-segment:

Mouse/Key	Single-segment	Multi-segment	Edge
`LMB`	Adds a point (second point completes the measurement) or positions the measurement segment	Adds a point	First click selects the edge or vertex, and the second click places the measurement segment
`MMB`	Completes the ruler without drawing	Completes the ruler without drawing	NA
`Ctrl+LMB`	Enters second point but waits to complete the measurement by allowing to place the segment by clicking `LMB`	NA	NA

© 2020 All Rights Reserved.

Mouse/Key	Single-segment	Multi-segment	Edge
`Shift+LMB`	Inserts a point so that the current measurement becomes multi-segment. Does not complete the measurement. You must enter one more point by using `LMB` or `Enter`. The `Shift+LMB` binding also freezes the dynamic measurement display of a ruler object.	NA	NA
`Ctrl+MMB`	Moves the measurement segment to the clicked location (after the first point)	Moves the measurement segment to the clicked location (after the first point)	NA
`Enter`	Completes the current measurement	Completes the current measurement	Completes the current measurement
`Esc`	Cancels the *Create Measurement* command	Cancels the *Create Measurement* command	Cancels the *Create Measurement* command

© 2020 All Rights Reserved.

(ICADVM20.1 Virtuoso RF Solution Only) For curved shapes. the measurement is drawn at a tangent to the point where you have snapped as shown in the figure below.

Note: Edge measurement is disabled for measurements on curved shapes.

You can create a ruler by using one of the following methods:

- Creating Rulers with Smart Snapping Off
- Creating Rulers with Snapping On

You can also create a ruler by using a combination of the above two methods – define the start or the end point with smart snapping on, and define the other point with smart snapping off.

Creating Rulers with Smart Snapping Off

In the *Smart snapping off* mode, the ruler object follows the Gravity Controls settings in the Layout Editor Options Form. If *Gravity On* is not selected, you can create the ruler by specifying the exact start and end points. If *Gravity On* is selected, ruler snapping to objects in the vicinity of the pointer is enabled. You can set the pointer to snap to the vertex, nexus, edge midpoint, edge, or centerline by selecting the appropriate *Types* options. In the *Smart snapping off* mode, with *Gravity On* and *Types* selected, the pointer snaps to the specified target type within the *Aperture* distance. The snap targets are not visually highlighted as in the smart snapping on mode. In the *Smart snapping off* mode, you cannot cycle the highlight over potential targets within the *Aperture* distance from the pointer.

To create a ruler when *Smart Snapping* is off:

© 2020 All Rights Reserved.

1. Choose *Tools – Create Measurement*.
2. Right-click in the design display area.

   The Ruler Context-Sensitive Menu appears.

3. Choose *Smart snapping off*.
4. Choose *Options – Editor*. This opens the Layout Editor Options Form.
   a. Select *Gravity On* or press `g`.

   This turns on ruler snapping to objects within *Aperture* distance from the pointer.

   b. Click *OK*.
5. Press `F3` or select *Options* from the Ruler Context-Sensitive Menu.

© 2020 All Rights Reserved.

This opens the Create Measurement Form.

a. Select the *Savable Measurement* checkbox if you want to create a persistent ruler, a ruler that is saved in the database when you save the cellview.

b. Click *Hide*.

6. Click in the design display area to specify the start point.

© 2020 All Rights Reserved.

After creating the start point of the ruler, you can move it around in the design display area depending on the specified *Snap Mode*.

7. Click again to specify the end point of the ruler.

If *Gravity On* is selected in the *Layout Editor Options* form, you can snap the ruler to the required target edge or point.

If you are creating a multi-segment ruler, click at the end of each segment and double-click to define the end point of the ruler.

A multi-segment ruler shows the total distance, in user units, between each point of the ruler.

8. Press `Esc` to finish creating rulers.

Creating Rulers with Snapping On

To quickly define the start and end points of a ruler or the various click points in the case of a multi-segment ruler without the need for zooming in to the exact click locations, select the snap target (edges, points, or both). This identifies objects in vicinity of the pointer and attaches to them.

When snapping is selected (edges, points, or both), the ruler object is created with the following features:

- Snap to vertex, nexus, edge midpoint, edge, or centerline.
- Cycle over potential targets within a specified distance from the pointer.
- Cycle over orthogonal end targets prior to parallel or non-orthogonal ones.
- Snap to targets within an hierarchy, in EIP mode, and inside a group.

  Rulers at level 0 remain visible when you EIP to any other hierarchical level.
- Snap to non-orthogonal edges as well as edges and vertices of instance boundary.

© 2020 All Rights Reserved.

By default, *Snap to Edges* is selected in the Ruler Context-Sensitive Menu. You can select any other smart snapping option from the Ruler Context-Sensitive Menu.

To create a ruler when snapping is on:

1. Choose *Tools – Create Measurement*.
2. Specify a value in the *Aperture* field on the *General* tab of the *Create Measurement* form.

   Aperture defines the distance between the pointer and the target object. The following figure shows the distance of the pointer from an edge at the distance at which the edge is highlighted.

Aperture = 1.0 *Aperture* = 5.0

3. Specify a value in the *Depth* field on the *General* tab of the *Create Measurement* form if your design is hierarchical to enable the pointer to snap to objects within the hierarchy.
4. In the Display Options Form, adjust the display level for displaying targets within the hierarchy.
5. Press `F3` or select *Options* on the *Ruler* context-sensitive menu.

© 2020 All Rights Reserved.

This opens the Create Measurement Form.

a. Select the *Savable Measurement* checkbox if you want to create a persistent ruler.

b. Select the *Multiple* from the *Segment Mode* drop-down to create a ruler with more than one line segment.

c. Choose a value from the *Snap Target* list.

© 2020 All Rights Reserved.

d. To specify the ruler and snap target visualization settings, use the options in the *Display Options* section of the form.

e. Select the snapping options in the *Snap Mode* and *Snap Target* field.

f. Click *Hide*.

6. Move the pointer in the design display area.

© 2020 All Rights Reserved.

A vertex, nexus, edge midpoint, edge, or centerline highlights dynamically based on the snapping option selected from the Ruler Context-Sensitive Menu or *Create Measurement* form.

7. Click near one of the highlighted targets.

The pointer snaps to the highlighted target.

8. Press the `Spacebar` key if there are multiple potential targets within the configured *Aperture* distance from the pointer.

The highlight cycles over these targets. With each `Spacebar` key press, the next possible target within the *Aperture* distance from the pointer is highlighted. You can highlight up to five targets in the proximity of the pointer by pressing the `Spacebar` key in succession.

9. Click to snap the pointer to the highlighted target.

- If the target is an edge or a center line, the pointer drops orthogonally from the point of click to the target edge or center line, respectively. In case of points, the pointer snaps to the target vertex, nexus, or edge midpoint. If the selected snapping option is *Mid Point of Distance*, click twice for the pointer to snap to the starting point of the smart ruler.

  Note: If you select *Mid Point of Distance* on the context-sensitive menu, by default, the *Snap Target* to *Edges* option appears selected on the context-sensitive menu.

  Click to snap the pointer to that edge. After you define the two targets, the pointer immediately snaps to the midpoint of the distance between the two targets on the selected edges.

10. Press `Esc` to finish creating rulers.

Note: If a ruler is started with the snap target as a point, the *Snap Mode* changes to *anyAngle*.

Note: When snapping labels in the `Ruler` command, snapping happens only to the origin of the labels.

Support for Objects	
Shapes	Snapping of ruler to all shapes is supported.
Blockages	Snapping of rulers to blockages is supported. All snapping modes, except snapping to *Nexus*, are supported for blockages.

© 2020 All Rights Reserved.

Rows	Snapping of ruler to rows is not supported.
Boundaries	Snapping of ruler to boundaries is not supported.

Note: In *Edit In Place* mode, the ruler can be snapped to top level shapes.

Note: For ruler and distance, the *Snap Mode* becomes *anyAngle* when the *Snap Target* is *Points* or *Edges and Points*.

Saving Rulers

By default, rulers are saved when you save the cellview. If you do not want to save the rulers, ensure that the *Savable Measurement* option on the *Create Measurement* form is not selected.

Deleting Rulers

You can clear all rulers from the design window by selecting *Tools – Clear All Measurements,* or from all levels of the hierarchy by selecting *Tools – Clear All Measurements In Hierarchy.* This will delete all persistent and transient rulers.

© 2020 All Rights Reserved.

Re-mastering Instance Views

The *Re-Master Instances* command is capable of hierarchically re-mastering cellviews based on the view names, cell names, and the library in which they reside. For example, re-mastering the abstract views with layout views directly before translating the final design to Stream format provides optimal performance. Replacing cellviews before editing the design can add unnecessary complexity to the design for routing and editing purposes.

You have a number options for the re-mastering cellviews. By leaving text fields blank in the form, entire libraries and cellviews can be searched for matching cellviews and view names. The * wild card can be used to replace any number of characters prior to or following the asterisk. See Re-Mastering Examples for more information.

Replacing Abstract Views with Layout Views

You can use the *File – Discard Edits* command to restore data to the last saved design in the event you make an error when replacing data. However, due to the extensive changes available when using this command, making a backup copy of your design to guard against errors when replacing data is recommended.

To search and replace instance abstract views with layout views,

1. Open the cellview in which you want to re-master instances.

   Instances are re-mastered only in the current open cellview.

2. Choose *Tools – Remaster Instances*.

   The Remaster Instances Form appears.

3. In the *Search for* group box, do one of the following:

   a. In the *Library* field, type the name of the library that contains the abstract cellviews you want to re-master.

   b. In the *CellName* field, type the name of the block or cellview that contains the cellviews you want to re-master.

      If the *CellName* field is left blank, the *Remaster Instances* command searches throughout the entire library in the *Update to* group box for matching cellview names.

   c. In the *ViewName* field, type the name of the view (abstract) to be re-mastered.

4. In the *Update to* group box, do the following:

© 2020 All Rights Reserved.

Note: You can use the *Browse* button to open the Library Browser and specify an Update to library name. The Library Browser lets you look though the libraries specified in your *cds.lib* file and set the library, cell, and view names for the form.

a. In the *Library* field, type the name of the library that contains the layout cellviews.

b. In the *CellName* field, type the name of the block or cellview that contains the layout cellviews.

If the *CellName* field is left blank, all cellviews that have matching cellview names specified by *ViewName* and *Library* name in the *Update to* group box are re-mastered.

c. In the *ViewName* field, type the name of the view (layout) that will re-master the current abstract cellviews specified by the *Search for* library.

5. (Optional) Turn on *Check Terminals* to check all corresponding cellviews for matching or a subset of matching instTerm information.

For more information, see Checking Terminals.

6. (Optional) Choose *File – Summary* to check the results of the *Remaster Instance* command.

In the *Instance Statistics* group box of the file, the master cellview name, view name, and library are listed for each cellview. Verify the results of the executed command by checking the master cellview names and views names for the specified cellview.

Note: Due to the complexity of the different types of errors, the CIW does not display a confirmation of the executed command or warning messages if the command is not successful.

© 2020 All Rights Reserved.

Re-Mastering Examples

Shown below are some of the more common types of searches and re-mastering of cellviews.

All instances in the current cellview that have masters from the library designLib and the view abstract are re-mastered to the library stdCellLib and the view Layout, and their master cell names are not changed.

All instances of designLib/fir_nib/abstract in the current cellview are re-mastered to Lefin_lib/fir_nib/Layout.

© 2020 All Rights Reserved.

All instances of designLib/fir_nib/abstract in the current cellview are re-mastered to stdCellLib/fir_nib/Layout.

All instances of designLib/MUX/abstract in the current cellview are re-mastered to Lefin_lib/MUX/Layout.

Checking Terminals

The *Check Terminals* option checks instTerms of cellviews for matching or a subset of matching instTerms. When *Check Terminals* is on, and the instTerms of the cellview in the *Search for* field are a subset of the instTerms of the cellview in the *Update to* field, or they are equivalent, the cellviews will be re-mastered.

Terminals expose a net up to the next level of the hierarchy. For example, a master might have 10 nets and 4 terminals (associated with 4 of the nets).

© 2020 All Rights Reserved.

When you create an instance of a master, it may have objects called instTerms associated with it. These instTerms map terminals in the instance's master to nets of the cellview in which the instance has been placed. The XL layout editor (or the Pick and Place command) creates and assigns these instTerms automatically. To expose these nets to the next level of the hierarchy, you create terminals for the nets, which are then mapped using instTerms at the next level.

For example, consider the instance in the following figure, which contains five pins: two on net A, two on net B, and one on net C. These nets need to be available at the next level of the hierarchy, so you create three terminals named A, B, and C.

Pins and Pin Numbers Terminal and Net Names instTerms and Nets

© 2020
All Rights Reserved.

Creating Pins from Labels

The *Tools – Create Pins From Labels* command lets you create pins from the labels and text displays in your layout cellview. You can specify the pin dimensions and the layer on which each pin is created. The generated pins are centered on the origin of the label or text display from which they are derived, and have terminal names matching those on the corresponding labels or text displays.

> **Note:** If the text label contains a colon (:), the colon and trailing text are dropped from the name of the generated pin. For example, given a text label VDD:P,28, the resulting pin name is VDD.

If a pin already exists for a selected label or text display, no new label is created for that object. The pin and object are not attached to each other. If you move the pin, the corresponding label or text display do not follow. To attach an object to the pin, see Attaching and Detaching Objects.

To create a pin from a label or text display:

1. In the layout canvas, select the labels or text displays from which you want to create pins.
2. Choose *Tools – Create Pins From Labels*.

   The Create Pins From Labels Form opens.

© 2020
All Rights Reserved.

3. Use *Objects Within* to specify whether to create pins at the top level of the *Cellview* or inside the *Selected Instances*.
4. Choose the *Object Type* to create pins from: *Label*, *Text Displays*, or *Both*.
5. Choose whether to create pins for *All* the *Objects* of the specified type or only those that are currently *Selected*.
6. In the *Pin Layer* cyclic field, choose the layers on which each pin is to be created.

   If you selected labels on more than one layer, choose the appropriate pin layer for each label layer from the list of available layers.
7. Type in the *Width* and *Length* of pins you require.
8. Click *OK* or *Apply*.

   The pins are created in the canvas and placed at the center of the labels.

© 2020 All Rights Reserved.

Using the Tap Command

There are two different *Tap* commands, the *Tap* command on the *Tools – Tap* menu and the *Tap* group box in the *Layout Editor Options* form.

Using the Tap Command on the Tools Menu

This *Tap* command accesses information from the following objects: paths, wires, rectangles, polygons, circles, ellipses, donuts, microwave objects, blockages, instances, vias, rows, labels and text displays. The Tap Form description explains how the Tap command affects the current layer, net, attributes, and command when each type of object is tapped.

To use the *Tools – Tap* Command

1. Choose *Tools – Tap*.
2. Press `F3` to open the form.

   The Tap Form appears.
3. Select the required *Tap Types*.
4. Click an object.

   The specified attributes will be picked from the selected object.
5. Choose a create command.

   The create form is populated with the information obtained from the original object you have tapped and seeds the form with the information you have selected in the *Tap* form.

Using the Tap Section in the Layout Editor Options Form

Wire Auto Tap, *Shape Auto Tap*, and *Tap Purpose List* are automatic during the creation of certain objects when the *Tap* fields are selected in the Layout Editor Options form. They are used during certain create commands when you want to automatically tap an existing object that is located at the first point of your new object.

Selecting *Shape* enables *Auto Tap* for rectangles, polygons, paths, circles, ellipses, and donuts. Selecting *Wire* enables *Auto Tap* for wires.

When tapping a wire or path, you can tap the LPP, extension type, and width of the wire. When using the XL layout editor, the net name and constraint group is also tapped.

© 2020 All Rights Reserved.

To tap specific layer purposes, create a prioritized list through the *Tap Purpose List* on the *Layout Editor Options* form or through the environment variables useTapPurposeList and tapPurposeList. The default purpose is `drawing`, which can be overridden through *Tap Purpose List*.

For example, if the *Tap Purpose List* is defined as:

```
drawing   net
```

- It sets the default purpose for `metal1` as `drawing`.
- Tapping an object on the LPP `metal1 net` changes the active layer to `metal1 net`.

If *Select from Overlaps* is on, the *Choose object to tap from* form appears if the clicked point has overlapping objects.

If *Select from Overlaps* is off, the object with an edge closest to the selected point is tapped. Also, the shape to be tapped is highlighted. You can use the space bar to toggle the highlight for shapes under the pointer. The layer information of the shape is displayed on the Selection toolbar.

© 2020
All Rights Reserved.

For detailed description of these options on the Layout Editor Options form, see the Tap section.

If you click an area in the design that has overlapping objects and start the `Tap` command, you can choose the object you want to tap by selecting it from the Choose Object to Tap form.

Note: If the *Auto Tap* option is selected in the Layout Editor Options form, a No Tap button is available on the Choose Object to Tap form to select objects where the `Tap` command should not be performed.

© 2020 All Rights Reserved.

Changing Objects on a Layer

The *Layer Generation* command lets you perform Boolean edits on layers, including resizing and merging objects on a layer. *Layer Generation* edits copies of the objects; the original objects are unchanged.

Making Changes to Objects on Layers

To enlarge copies of selected objects:

1. Select the objects you want to resize.
2. Choose *Tools – Layer Generation*.
3. Fill out the Layer Generation Form as shown below.

4. Click *OK*.

   Copies of the selected objects are enlarged.

Original objects on contact layer

After *GROW BY*, enlarged copies of originals appear on the *metal1* layer.

© 2020
All Rights Reserved.

How Sizing Converts Objects to Polygons

When you resize an object such as a path by using Size, the object is converted to a polygon. When you use Layer Generation Form, the original path remains and the new object on the new layer is converted to a rectangle or a polygon.

For example, if you enlarge a path with *Size*, the resulting object is a polygon.

A path before resizing

After resizing, the path has been converted to a polygon.

Making Boolean Edits by Layer

You can perform Boolean logical operations on objects on one or two layers. You edit copies of the objects, so the originals are unchanged.

1. Select the objects you want to edit.
2. Choose *Tools – Layer Generation*.

   The Layer Generation Form opens.
3. Choose the layers you want to edit and the Boolean operator you want to use.

4. Click *OK*.

© 2020
All Rights Reserved.

The following figure shows the results of different Boolean operations.

© 2020
All Rights Reserved.

Generating Abstracts

You can create an abstract of the open design by running the *Tools - Abstract Generation* command. This opens the Virtuoso Abstract Generator tool. For information about how to set different options in this tool to obtain the desired abstract, see the *Virtuoso Abstract Generator User Guide*. You can use abstracts for place and routing.

© 2020 All Rights Reserved.

Working with Markers

To get information about a marker, use these commands:

- *Verify – Markers – Explain* displays the reason for the error or warning marker in a text window
- *Verify – Markers – Find* searches for and highlights each error or warning marker

After you get the information you need, you can delete the marker using these commands

- *Verify – Markers – Delete* removes a specific marker
- *Verify – Markers– Delete All* removes all markers

Explaining Markers

You select markers either before or after you choose *Verify – Markers – Explain*.

To select markers,

1. Click the marker or draw a selection rectangle around multiple markers.
2. Choose *Verify – Markers – Explain*.

   The Marker Text Window appears and displays information about the error.
3. Continue selecting markers you want explanations for.

   The text scrolls in the marker text window so you can see it.
4. When you are finished viewing the errors, press `Escape` to end the command.
5. You can correct the errors and rerun the verification command.

Finding Markers

Verify – Markers – Find searches for and highlights each error or warning marker in a layout to help you correct errors. You can search for markers anywhere in the design hierarchy. You can also display the previous or next marker.

To find markers in a layout,

1. Choose *Verify – Markers – Find*.

   The Find Marker Form appears.

© 2020 All Rights Reserved.

2. Change the options on the form as desired.
3. Click *Apply*.

   The first error or warning marker is highlighted and the reason for the marker is displayed in the marker text window.

   Note: You can change the options on the form at any time and click *Apply*.
4. To find the next marker, click *Next* on the form, or to look at the previous marker, click *Previous* on the form. Then check the marker text window for the reason for the marker.
5. To delete the current marker, click *Delete*.

Note: There is a similar *Find Marker* command available in the schematic editor. The command options are a little different than the layout editor version and provide additional functionality. The only way to run the schematic editor *Find Marker* command in a layout window is by typing the Cadence® SKILL language function `geHiCommonFindMarker()` in the Command Interpreter Window (CIW). For more information on this command, see the *Virtuoso Schematic Editor User Guide*.

Deleting Individual Markers

To delete individual markers in the design window,

1. Indicate which marker you want to delete by doing one of the following:
   - Click the marker(s) you want to delete individually.
   - Draw a selection box around multiple markers.
   - Choose the message on the Find Marker form that corresponds to the marker you want to delete.
2. Choose *Verify – Markers – Delete*.

   If you press `F3`, the Delete Marker Form appears.

3. Select the *Cellview Access Mode*.

© 2020 All Rights Reserved.

If you set *Cellview Access Mode* to *edit*, select the *Confirm Access Mode Change* check box, and then point at a marker to delete it, the *Delete Marker* dialog box appears for you to confirm opening the selected cellviews in edit mode.

Select the ones that you want to open in edit mode and click *OK*. If you do not want to open the cellviews in the edit mode, select *None* and click *OK*.

If you set the *Cellview Access Mode* to *read* (the *Confirm Access Mode Change* check box is disabled) and then point at a marker to delete it, the marker is deleted and the cellview remains in read mode.

4. Click *Cancel* in the *Delete Marker* form to finish deleting the required markers.

Deleting All Markers

To delete all markers in the design window,

1. Choose *Verify – Markers – Delete All*.

© 2020 All Rights Reserved.

The Delete All Markers Form opens.

2. Select the *Severity* types, *Search Scope*, and the *Source*.
3. In the *Options* section, select the *Cellview Access Mode*.
4. Click *OK* or *Apply*.

   Clicking *OK* closes the form.

© 2020 All Rights Reserved.

If you set *Cellview Access Mode* to *edit*, select the *Confirm Access Mode Change* check box, and then click *OK* or *Apply*, the *Delete All Markers* dialog box appears for you to confirm opening the selected cellviews in edit mode.

Select the ones that you want to open in edit mode and click *OK*. If you do not want to open the cellviews in the edit mode, select *None* and click *OK*.

If you set the *Cellview Access Mode* to *read* (the *Confirm Access Mode Change* check box is disabled) and click *OK* or *Apply*, the markers are deleted and the cellview remains in read mode.

© 2020 All Rights Reserved.

Pad Opening Info Command

In hierarchical designs, a pad shape and its corresponding pad label might not be at the same hierarchical level. The *Tools – Pad Opening Info* command helps you find pad shapes at lower levels of your hierarchical cellview, generate reports and labels for the pad shapes that are found, and promote the generated labels to the top level of the cellview. This command is supported for all shapes. except lables and text displays.

For more information, see:

- Generating a Standard Report
- Generating a Customized Report
- Generating Labels for Pads
- Promoting Labels to the Top Level

Generating a Standard Report

To generate a standard report containing information on the pad shapes found in all or part of a hierarchical cellview:

1. Open a hierarchical cellview that contains bond pad shapes.
2. Choose *Tools – Pad Opening Info* from the Layout menu bar.

   The Pad Opening Info Form appears.
3. Click the *Generate Report* radio button at the top of the form.

The *Reporting* options are shown at the bottom of the form.

© 2020 All Rights Reserved.

4. Specify the *Search Criteria* for the pad shapes and labels you are interested in.

   a. Select the appropriate *Bond Pad Layer/Purpose* to search for pad shapes.

   b. Set the *Search Depth* to limit the number of hierarchy levels to be searched. For example, specifying 5 means that levels 0 through 5 are searched.

   c. Set the X and Y *Dimensions* for the pad shapes you are interested in, if required.

   d. Use *Create Rect* or *Create Polygon* to limit the region in the cellview that is searched if required. Either draw the region you require in the canvas or edit the coordinates manually in the field.

   e. Specify the *Label/Name Layer/Purpose* and *Search Depth* to search for existing labels. For example, the settings above search LPP `text drw` at level 2 only.

5. Specify the coordinates for the first pad to be generated by clicking the *1st PAD coord*s button and then clicking on the desired location in the canvas.

   The coordinates are entered in the field automatically. You can edit them manually if required.

© 2020 All Rights Reserved.

6. Choose whether subsequent pad shapes should be labeled in the *Clockwise* or *Counter Clockwise* direction.

7. Select the *Make Standard Report* check box and use the *Browse* button to specify the directory and file name for the report file.

8. Click *OK* or *Apply* at the bottom of the form.

© 2020
All Rights Reserved.

The report file is displayed.

If the design contains duplicate pads, the duplicate pads are displayed in the report only once, as shown below.

```
**********
Pad Opening Locator
**********
Created : May  1 11:28:17 2015
Libname : myLib
Cellname : p1

Pad Label Hierarchy : 0
8 bond pad elements found
0 labeled bond pad elements found
8 non-labeled bond pad elements found
**********

**********
Pad Opening Name      X              Y              Area           Width:Height
**********
NO_NAME               -0.500000      -4.250000      21.000000      6.000000 : 3.500000
NO_NAME               -7.250000      2.750000       22.750000      6.500000 : 3.500000
NO_NAME               8.250000       -1.500000      21.000000      3.500000 : 6.000000
NO_NAME               22.250000      2.250000       22.750000      6.500000 : 3.500000
NO_NAME               31.499500      -3.999500      5.004169       2.237000 : 2.237000
NO_NAME               27.499500      -5.999500      5.004169       2.237000 : 2.237000
NO_NAME               22.118500      -7.881500      5.004169       2.237000 : 2.237000
NO_NAME               18.118500      -9.881500      5.004169       2.237000 : 2.237000

# The design contains duplicate pads. The duplicate pads are listed only once in the above report.
```


© 2020 All Rights Reserved.

Note: If there are multiple labels, the pad opening name will be the label that has minimum distance between the pad origin and the label origin.

Generating a Customized Report

To create a customized report containing information on the pad shapes found in all or part of a hierarchical cellview,

1. Open a hierarchical cellview that contains bond pad shapes.
2. Choose *Tools – Pad Opening Info* from the Layout menu bar.

   The Pad Opening Info Form appears.
3. Make sure you are in *Generate Report* mode.

   The *Reporting* options are shown at the bottom of the form.
4. Set the *Search* and *Sort* criteria as required.

   See Generating a Standard Report steps **4** through **6** for more information.
5. Choose *Make Customized Report* in the *Reporting* group box at the bottom of the form.

   The *Add*, *Del*, and *Edit* buttons are enabled.

© 2020 All Rights Reserved.

6. Click the *Add* button to create a customized report. The Create Encore Report Form is displayed.

7. Specify the headers to be included in the report by selecting an option from the *Headers* list on the left side and clicking the >> button. The selected header name is moved to the right box.

   To remove a header from the report, select the header in the right hand list and click <<.

8. Specify the columns to be included in report by selecting an option from the *Columns* list on the left side and clicking the >> button. The selected header name is displayed in the right box.

   To remove a column from the report, select the column label in the right list and click <<.

© 2020 All Rights Reserved.

9. Specify where you want the report to be saved in the *File Path* field. You can use the *Browse* button to navigate to the desired location.

   For information on how to filter the data shown in your customized report, see Setting Filters for a Customized Report.

10. Click *OK*

    The custom report report is generated and saved to the specified location on disk.

Setting Filters for a Customized Report

You use the **Pad Opening Info: Filter** form to filter the data shown in a customized report. To do this:

1. Click *Set Filter in* the Create Encore Report Form.

© 2020 All Rights Reserved.

The Pad Opening Info: Filter form is displayed.

2. To enable a filtering selection, click the check box on the left and specify the mode by choosing either *Pad Opening Name*, *X*, *Y*, or *Area* from the first drop-down list.

3. Choose the appropriate operator from the second drop-down list.

   For example, to specify the criterion equal to the content in the left field, select the == operator from the drop-down list.

   The other valid operators in *Pad Opening Name* are *!=*, *contains*, and *doesn't contain*.

   For *X*, *Y*, and *Area*, the valid operators are: *==*, *!=*, *<*, *<=*, *>*, *>=*.

4. Specify the required values in the blank field.

5. Select the operator for logical operation to the fields below from the final drop-down list. The valid operators for logical operation are "and" and "or".

© 2020
All Rights Reserved.

6. Click *OK*.

   The filters are set for the customized report and displayed in the Filter field in the Create Encore Report Form.

Generating Labels for Pads

Use *Generate Labels* mode to specify what label names are generated and how they will be drawn in the canvas.

1. Open a hierarchical cellview that contains bond pads.
2. Choose *Tools – Pad Opening Info* from the Layout menu bar.

   The Pad Opening Info Form is displayed.
3. Click the *Generate Labels* radio button at the top of the form.

4. Set the *Search* and *Sort* criteria as required.

   See Generating a Standard Report steps **4** through **6** for more information.
5. In the *Label Control* group box at the bottom of the form.

   a. Set the pattern for the pad naming in the *Label (Pattern)* field.

   The label names will include the label prefix and the pad numbers (for example, PAD<1>, PAD<2>, and so on) or specify a file from which the names are to be read.

   b. Select the appropriate *Layer/Purpose*. This is the layer the label will be printed on.

   c. Set the *Font*, *Height*, and *Justification* for the labels as required.

   For more information on the available options, see Label Control.
6. If required, click *Delete Bond Pad Labels* to remove all of the labels from the top level.
7. Click *Apply*.

   You can now zoom in to any pad and see the label information placed on the pad opening.

© 2020 All Rights Reserved.

Promoting Labels to the Top Level

Use *Promote Label to Top* mode to promote generated labels to the top level of the cellview. You can also control how the labels are drawn on the canvas and whether the corresponding lower-level labels should be removed.

1. Open a hierarchical cellview that contains bond pads.
2. Choose the *Tools – Pad Opening Info* from the Layout menu bar.

   The Pad Opening Info Form is displayed.
3. Click the *Promote Label to Top* radio button at the top of the form.

4. Set the *Search* and *Sort* criteria as required.

   See Generating a Standard Report steps 4 through 6 for more information.
5. In the *Promote Label* group box at the bottom of the form:
   a. Select the appropriate *Layer/Purpose*. This is the layer the label will be printed on.
   b. Set the *Font*, *Height*, and *Justification* for the labels as required.
   c. Choose whether to *Remove Original Labels*.

   For more information on the available options, see Promote Label.
6. Click *Apply*.

   The required labels are promoted to the top level of the cellview.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

5

Working With Hierarchical Designs

This chapter contains these topics:

- Hierarchical Designs
- Hierarchical Editing
  - Descending into a Cellview
  - Returning to a Different Hierarchy Level
  - Editing a Cell in Place
  - Editing in Place into an Instance Array
  - Using the Tree Form
  - Using the Print Hierarchy Tree Command
  - Making Hierarchy Editable
  - Making Cells
  - Flattening Instances
  - Copying and Cutting Through the Hierarchy
- Embedded Module Hierarchy Designs
- Listing the Data In Cellviews
- Refreshing Memory after Editing

© 2020 All Rights Reserved.

Hierarchical Designs

Hierarchical designs are created by placing instances of other references to cellview inside other cellview.

Levels of hierarchy are numbered from top to bottom, with the top cellview having the lowest number. The top cell is the cellview you are currently viewing in a design window, unless you have descended into the hierarchy or are using *Edit In Place*.

In the above figure, when you view Top cell, it is level 0. When you view Cell C or Cell D, it is considered level 1, and instances Cell A and Cell B are considered level 2.

Master Cells and Cell Instances

When you place a cellview within another cellview, the placed reference of the cellview is called an instance. The placed instance now has a master cellview. In the above example, Cell A has been placed twice in the design. If you edit the master of Cell A, all instances of that cellview are changed.

Checking for Instance Recursion

You can prevent commands such as *Create Instance*, *Make Cell*, and *Search/Replace* from creating an instance of itself in a cellview by turning recursion checking on in the *Editor Options* form.

To automatically check for recursion when placing instances,

1. From the cellview window menu, choose *Options – Editor* [`Shift-e`].

   The Layout Editor Options Form opens.

© 2020 All Rights Reserved.

2. Click *Recursion Check*.
3. Click *OK*.

Changing the Origin of a Cellview

Caution

Moving the origin of a master cellview changes the location of all placed instances of that cellview.

Support for Objects	
Blockages	The *Move Origin* command fully supports blockage objects. ■ When you move the origin of blockage objects without owners, the coordinates are updated. ■ When you move the origin of owners of blockage objects, the blockage coordinates are also updated.
Boundaries	The *Move Origin* command fully supports boundary objects. ■ For a snap boundary, a form appears asking “Would you like to rearrange objects taking Snap Boundary as origin?”

To change the origin of the cellview,

1. From the cellview window menu, choose *Edit – Advanced – Move Origin*.

   Note: The *Move Origin* command is disabled if you are editing a group.

2. Move the pointer where you want the new origin.

© 2020 All Rights Reserved.

Two crosshairs showing the new X and Y axes follow the pointer.

Hierarchical Editing

Descending into a Cellview

The *Descend* command lets you descend into any level of an instance hierarchy, and the *Descend* form lets you choose the view type/name when descending into instances.

To descend into a cellview,

1. Choose *Edit – Hierarchy – Descend Edit* or *Descend Read.*

   You can choose whether you want to edit using *Descend Edit*, as long as the parent is opened in edit mode, or read only using *Descend Read* that allows you to descend into an instance and view the contents without the ability to edit the instance.

2. Optionally you can choose to be prompted for the view type by pressing F3 to bring up the Descend Options Form and checking the box.

   The default is set so such that you are not prompted for the view type. If the *Prompt For View Name* field is not set, and the master cell of the instance has more than one view type available, the command will determine which cellview to descend into. If the *Prompt For View Name* field is set, and the master cell of the instance has more than one view type available, a form is displayed containing a cyclic field with all the possible view names listed. You can then select the cellview to descend into. By selecting Prompt for View Name, each time you descend into a cellview, you will be prompted for the view type.

© 2020 All Rights Reserved.

3. Click the instance whose cell you want to edit. You can use the `Spacebar` key, to toggle between the overlapping objects under the pointer. (see Cycling Through Preselected Objects)

Tip

You can select the instance and right-click it to display the *Instance* context menu. Select *Descend Edit* or *Descend Read.*

4. Return up the hierarchy by choosing *Edit – Hierarchy – Return To Level* [b] or *Return* [Shift + b]. Alternatively, you can right-click and select *Return* on the *Layout* context menu. You can also choose *Edit – Hierarchy – Return To Top,* to return to the top level of the instance hierarchy.

Note: Once you have chosen whether to be prompted for the cellview type, you can or post-select instance when using the *Descend* command.

You can set in advance which mode you prefer using the `leHiDescend` SKILL function. It has an optional text string *`t_mode`* specifying the mode in which to open the selected instance. When "`nil`", the instance is opened in the same mode as its parent. When "`read`", the instance is opened in read mode. When "`edit`", the instance is opened in edit mode if its parent is open in edit mode; otherwise, a window pops up stating that the instance cannot be opened in edit mode and advising that you open it in read mode. When the *`t_mode`* argument is not specified, automatically selects the edit mode to use when descending into the hierarchy.

Note: The layout editor does not use the *Create New Window When Descending* option in the User Preferences form. Selecting or deselecting this option will not affect your work in the layout editor.

Note: If the top cellview is opened in Layout Viewer, the descend into a new tab or window always opens the master in the Layout Viewer, regardless of the default application settings.

Returning to a Different Hierarchy Level

If you have use *Edit in Place* or *Descend*, you can return to a higher level in the hierarchy.

To return to a specific level,

1. To return up the hierarchy by choosing *Edit – Hierarchy – Return To Level* [b] or *Return* [Shift + b].
2. When choosing *Return To Level*, click a cellview name in of the cellview.
3. Click *OK*.

© 2020 All Rights Reserved.

If you are using *Edit In Place*, the window border changes to show the cell you chose.

If you previously used *Descend,* the window contents change to show the cell you chose.

Editing a Cell in Place

You can edit the master cell of an instance while viewing the instance in the current cellview. This is called editing in place (EIP).

The behavior of the *Edit In Place* command differs based on the selection mode. Consider the following design that illustrates the difference between the two modes. Cellview `cv1` contains the instance `ILevel1`.

Using the `Spacebar` key, you can toggle between the overlapping objects under the pointer. You can also choose *Edit – Hierarchy – Return To Top,* to return to the top level of the instance hierarchy.

When you have created a group and you select the *Edit In Place* option, you can use the *Remove from Group* option in the context-sensitive menu to remove a shape from the group.

Editing in Place into an Instance Array

When editing in place into an instance array, the Select Array Member Form appears which allows you to enter the row and column information for the array member you want to edit.

© 2020 All Rights Reserved.

Cellview `cv1` Containing the Instance `ILevel1`

© 2020 All Rights Reserved.

The instance `ILevel1` contains a polygon, the Pcell instance `ILevel2pcell`, and the instance `Ilevel2`.

Instance `ILevel1`
Containing the Instance `ILevel2`,
the Pcell Instance `ILevel2`pcell,
and a polygon

Edit In Place in Pre-Selection Mode

In `cv1`, select `ILevel1` and then run the *Edit - Hierarchy - Edit In Place* command or right-click the selected `ILevel1` and select *Edit In Place* from the context menu. The master cellview of `ILevel1` displays for editing in place. Because you can only top-level instances, the ion mode lets you EIP only one level in the hierarchy. To EIP other levels of the hierarchy, select the instance again and run the *Edit In Place* command each time to descend the hierarchy. To ascend the hierarchy, select *Edit – Hierarchy – Return To Level* [b] or *Return* [Shift + b].

Edit In Place in Post-Selection Mode

The behavior of the post-selection mode *Edit In Place* command depends on the area clicked for selection. Invoke the *Edit - Hierarchy - Edit In Place* command and then click the bounding box of `ILevel1`. The following situations are possible:

- If the click point is on the bounding box of `ILevel2`, the master cellview of `ILevel2` becomes the cellview to be edited in place. The banner at the top of the window changes to show that the master cell for that instance is open for edit. If the *EIP Surround* is off in the *Display Options* form, only the cell to be edited in place appears.

© 2020 All Rights Reserved.

- If the click point is in the area of `ILevel2pcell`, the message “Cannot edit in place a parameterized cell” displays because the cellview at the lowest level in the hierarchy at the point of click is a Pcell. If the click point is at an empty area within the `Ilevel2pcell` bbox, there is a beep sound and the “Point at a shape in the cellview to be edited-in-place” prompt.
- If the click point is in the area of the polygon, the master cellview of `ILevel1` becomes the cellview to be edited in place.
- If the click point is in an empty area in `ILevel1`, there is a beep sound and the “Point at a shape in the cellview to be edited-in-place” prompt.

In the post-selection mode, when the pointer is over a shape of the cellview to be edited in place, the shape is highlighted. You can EIP ‘n’ levels; the destination cellview (master of the lowest instance in the hierarchy) should not be a Pcell. To return up the hierarchy, select *Edit – Hierarchy – Return To Level* [b] or *Return* [Shift + b].

Note: When you are done editing with *Edit In Place*, and as you return up the hierarchy, each level you pass is accessible in edit mode if you started in a cell that is in edit mode. If a lower-level cell was originally in read-only mode, it will return to read only once you return up the hierarchy. Additionally, if you have changed anything in a lower cell, you will be prompted to save that master cell.

Video

For an overview on how to toggle shapes in EIP and Descent Edit/Read mode, see the video *Toggling shapes in EIP and Descend Edit / Read mode*.

Note: *Edit In Place* and *Descend Edit* options are available for virtual hierarchies.

Freezing and Unfreezing Pcell Instances

For Pcell editing, all geometric updates are performed through the Pcell parameters. Also, stretch handles provide you with a graphical way to update Pcell geometries. However, these graphical updates are also internally managed through Pcell parameters. For user-defined Pcells, there may be a situation when the desired Pcell editing cannot be performed by the supported Pcell parameters. In such cases, you can use the freeze and unfreeze Pcells options. The freeze and unfreeze commands enable you to perform Pcell editing that is not supported by the Pcell parameters.

The freeze Pcells option enables you to convert a Pcell instance to a non-Pcell instance of a new master created on the disk. Similar to any other instance, once you freeze the Pcell instance, you can use the EIP command to perform geometric updates to the design master.

© 2020 All Rights Reserved.

The unfreeze Pcells option enables you to convert the frozen non-Pcell instance to its original Pcell instance.

Note: The freeze Pcells option creates a new design master on the disk. So, it should be used only when the Pcell parameters are not sufficient to create the required geometry.

You can use one of the following methods to use the freeze and unfreeze options:

- Use the below mentioned SKILL functions use the freeze and unfreeze options:
  - Use the `leFreezeInst` function to convert a Pcell instance to a non-Pcell instance of a new master created on the disk. This function preserves the geometry, connectivity, and cross-probing with schematic information of the instance.
  - Use the `leUnfreezeInst` function to convert the frozen non-Pcell instance to its original Pcell instance. All editing that has been performed on the non-Pcell instance is not preserved.
- Use the context-sensitive menu options, *Freeze Pcell* and *Unfreeze Pcell*.
  - The *Freeze Pcell* option is available in the context-sensitive menu of a Pcell instance. This option enables you to freeze a Pcell instance.

In the Freeze Pcell Instance Form, you specify the library, cell, and view of the non-Pcell instance of the new master created on the disk.

© 2020 All Rights Reserved.

Note: Alternatively, you can use the supressFreezePcellUI environment variable to freeze a Pcell instance without displaying the Freeze Pcell Instance Form.

- The *Unfreeze Pcell* option is available in the context-sensitive menu of a frozen non-Pcell instance. This option enables you to unfreeze a frozen non-Pcell instance.

Using the Tree Form

The *Tree* command displays a list showing the hierarchy of cellview instances in the current cellview.

To list the hierarchy of instances in the current cellview,

1. Choose *Edit – Hierarchy – Tree*.

   The Tree Form appears.

2. Set *Display* to one of the following:

   - To list the hierarchy from top level to stop level, select the *Top to stop level* option.
   - To list the hierarchy from the current level of hierarchy to the stop level, select the *Current to stop level* option.

© 2020 All Rights Reserved.

- To list the hierarchy from the top level to the current level of hierarchy, select the *Top to current* option.
- To list the hierarchy from the current level of hierarchy to a level of hierarchy that you specify, select the *Current to user level* option.

Note: The *Display Levels: Start and Stop* settings on the Display Options form limit the number of levels reported by this command.

3. Click *OK*.

   A text window, *Tree*, listing the hierarchy appears.

4. Choose *File – Close* to close the window.

Tree Window Information

The cellview instances information lists the hierarchy of instances in this cellview or the instances above or below the edit in place cell. The number in parentheses represents how many times that instance appears in the design.

Indents show that instances of the indented cells are inside the instance listed above.

Using the Print Hierarchy Tree Command

The *Print Hierarchy Tree* command displays a list showing the hierarchy of cellview instances in the selected cellview.

To list the hierarchy of instances in the selected cellview,

1. Choose *Tools – Print Hierarchy Tree* from the CIW.

   The Print Hierarchy Tree Form form appears.

2. Select the *Library*, *Cell*, and *View* for which you want print the hierarchy tree.

3. Click *OK/Apply*.

   The text window, *Hierarchy Tree*, listing the hierarchy appears.

The *Print Hierarchy Tree* command has better performance compared to the *Edit – Hierarchy – Tree* command. This command lets you print the hierarchy tree without the need to open the design, which could take long time to open for a large design.

© 2020 All Rights Reserved.

Hierarchy Tree Window Information

The cellview instances information lists the hierarchy of instances in this cellview or the instances above or below the edit in place cell. The number in parentheses represents how many times that instance appears in the design.

Indents show that instances of the indented cells are inside the instance listed above.

Note: There are two versions of hierarchy tree described in the above sections, *Edit – Hierarchy – Tree* and *CIW – Tools – Print Hierarchy Tree*. *Edit – Hierarchy – Tree* prints the hierarchy as displayed in the *Navigator* assistant. It considers edits in the virtual memory. However, *CIW – Tools – Print Hierarchy Tree* does not consider edits in the virtual memory which are not committed.

Making Hierarchy Editable

Use the *Make Editable* command to switch to the edit mode for updating the hierarchy.

To make a hierarchy editable,

1. Choose *Edit – Hierarchy – Make Editable*.

   The Make Readonly/Editable Form opens. This form lists master cellviews of all the instances existing in the current cellview.

2. To list only specific master cellviews, use the *Filter* options to specify a search criteria.

   a. You can display the master cellviews of all or only the instances selected in the cellview

   b. You can specify a *Name* and *Type* criteria for filtering the master cellviews to display in the left pane.

3. Select a *Readonly* master cellview in the left pane and click *Make Editable*. Alternatively, select an *Editable* master cellview in the left pane and click *Make ReadOnly*. You can select multiple master cellviews and change their mode.

Making Cells

You can use the *Make Cell* command to add levels of hierarchy.

Support for Objects

© 2020 All Rights Reserved.

Blockages	■ When you select blockages without selecting their owner, a new cell with the selected blockages is created without any owner information. When *Replace selected figures* option is selected, *Make Cell* replaces all the selected set including blockage objects with the instance of the new cell in the current cellview. Once it is replaced, the ownership information is not retained. ■ When you select both the owner(s) and the blockage(s), a new cell with the selected owner, all its children, and all associated blockages is created. When *Replace selected figures* is option is selected, all selected objects are replaced in the current cellview with the instance of the new cell and its owner.
Boundaries	The *Make Cell* command does not support snap and cluster boundary objects. ■ For area boundary objects, If the constraint group name is part of the same cellview, both the area boundary object and the constraint group are copied over. If the constraint group is not part of the same cellview, only the area boundary object is copied over. ■ For the PRBoundary object, if its associated blockages are selected as well, the relationship between the PRBoundary and blockage(s) objects must be maintained. Even when the *Replace selected figures* option is selected, the prBoundary remains in the current cellview.

You can copy selected objects in the current cellview into a new cell with the layout editor.

To create a cell from objects in your layout,

1. Select the objects you want to place into a new cell.
2. Choose *Edit – Hierarchy – Make Cell*.

   The Make Cell Form opens.
3. Type the library, cell, and view names for the new cell.
4. In the *Make Cell* form, do one of the following:
   - ❑ Select the *Replace selected figures* option to replace the selected objects with an instance of the new cell.
   - ❑ Deselect the *Replace selected figures* option to leave the objects unchanged.
5. Click *OK*.

© 2020 All Rights Reserved.

The lowerLeft point of the bounding box of the created cell will be placed at the origin (0,0).

Note: If you type the name of an existing cell, a dialog box lets you choose to overwrite the existing cell with a new one.

Flattening Instances

The *Flatten* command copies the contents of a cell or array up one or more levels in the hierarchy.

Support for Objects	
ROD Objects	The *Flatten* command fully supports ROD objects. To preserve the attributes of ROD objects (such as object name, alignments, multipart path subparts, multipart rectangle sub-rectangles, and master rectangles in multipart rectangles), select the *Preserve: ROD objects* option on th Flatten form. When this option is off, ROD objects become ordinary unnamed objects, the subparts of multipart paths become ordinary paths and rectangles, the master rectangles in multipart rectangles become ordinary, unnamed rectangles, and the sub-rectangles of multipart rectangles become ordinary rectangles. The system assigns the flattened object a name based on the hierarchical name of the ROD object by replacing slashes with dashes. For example, when you flatten the ROD object `I1/I4/rect3`, the resulting object is named `I1-I4-rect3`.

© 2020 All Rights Reserved.

Blockages	Only instances contained within the blockage are flattened. The blockage object itself is not flattened. ■ If you select an instance which is the owner of a blockage to flatten (based on the flattening level), only the instance is flattened, and the blockage remains without any ownership information. If the owner instance is not selected for flattening, then the ownership relation remains unchanged. ■ When you flatten an instance with a placement halo, the placement halo will convert to a placement blockage with the appropriate points. ■ If you flatten instances containing blockages and you want to delete the remaining blockages, use the *Delete Detached blockages* option on the *Flatten* form.
Boundaries	The *Flatten* command does not support P&R and snap boundary objects. ■ If you flatten an area boundary object which results in a non-unique name, an extension is added to the name, with a resulting informational message appearing in the CIW. If the associated constraint group no longer exists after the flatten, the object is stripped of its constraint group name, with a resulting informational message appearing in the CIW. All normal transformations associated with flattening of an instance will also apply for the area boundary object. ■ If you flatten a cluster, with a cluster boundary and the cluster boundary becomes irrelevant, the cluster object set is updated and the cluster boundary is removed.If you flatten a cluster boundary object which results in a non-unique name, an extension is added to the name, with a resulting informational message appearing in the CIW. It is possible to have a hierarchical set of clusters based on the depth of the flatten command. All transformations associated with the flattening of an instance will also apply for the cluster boundary object. ■ If the flatten command results in a cellview that has no instances, you must remove all clusters and cluster boundary objects from the cellview.

To display the detail in all instances you want to flatten,

© 2020 All Rights Reserved.

1. Choose *Options – Display* [e].

   The Display Options Form opens.
2. Set *Display Controls* to display the detail in all instances you want to flatten.
3. Click *OK*.
4. Choose *Edit – Hierarchy – Flatten*.

   The Flatten Form opens.
5. Set the *Hierarchy Level* to *one level*, *displayed level*, or *user level*.
6. Click *OK*.

The original instance (bold outline) contains four via instances.

One level flattened— geometries in the via instances are not flattened.

All levels flattened

Note: When flattening Pcell instances, you can delete fluid shapes by specifying the environment variable deleteFluidShapes.

Copying and Cutting Through the Hierarchy

Yank and *Paste* can cut and copy all or part of instances and shapes in a designated area.

To copy parts of cell instances,

1. Choose *Options – Display* [e].

   The Display Options Form appears.
2. Set *Display Controls* to display the detail in all instances you want to copy.
3. Click *OK*.
4. Choose *Edit – Basic – Yank* [y].

   The Yank Form opens.

© 2020 All Rights Reserved.

5. Set the *Yank Levels*.
6. Click and drag to create a box around the area you want to yank.
7. Choose *Edit – Basic – Paste* [`Shift-y`].
8. Click to place the copies.

The yank box (bold) cuts through the transistor instance on the left and encloses an instance on the right.

Paste places flattened copies of objects from the left instance plus the whole instance on the right.

Embedded Module Hierarchy Designs

Uniquifying the Cellview

Cellviews that contain EMH elements must be uniquified before making changes involving hierarchical objects. Examples would be creating, deleting, or editing objects with hierarchical names. An example of an edit operation that requires you to uniquify the cellview is the creation of an instance using a hierarchical path name.

Caution

You can open a non-uniquified design in Virtuoso Layout Suite. However, while performing an edit operation on the design, which tries to change the hierarchy of the EMH, you will receive warning messages regarding the non-uniquified instances. You will then need to uniquify the cellview.

The uniquification feature checks out the VLS XL or a higher license. The command works only on `oacMaskLayout` view types.

To uniquify the cellview:

1. Select *Tools — Uniquify* in the CIW.

   This opens the Uniquify Form.

© 2020 All Rights Reserved.

2. Specify the library, cell, and view names in the *Library*, *Cell*, and *View* fields. If uniquification happens successfully, a message displays after its completion.

   If the cellview is already uniquified, a message appears to inform you about it. If the specified cellview cannot be opened, the command exits displaying an appropriate message about the failure.

   The license is checked back in when the *Uniquify* form is closed.

Note: The *Uniquify* command does not disable the *Undo* command, and it can, therefore, be used when you uniquify the cellview.

Listing the Data In Cellviews

To display a summary of the contents of this cellview,

1. Choose *File – Summary*.

© 2020 All Rights Reserved.

A Summary Window containing the summary appears.

2. Choose *File – Close* to close the window.

Refreshing Memory after Editing

Another user might edit a master cellview that is instantiated in the cellview you currently have open. You can update the memory of your computer to reflect the results of such edits with the *Refresh* command.

To view other users' edits,

1. Choose *Edit – Hierarchy – Refresh*.

© 2020 All Rights Reserved.

- If no other users have made edits that affect this cellview, a dialog box tells you that no cellviews were refreshed.
- If the master cells for any instances in this cellview have changed, a message asks if you want to update these instances.

2. In the Refresh Cellviews Form, set any instances you want to update.

3. Click *OK*.

The cellview is redrawn to show changes made to the master cells.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

6

Creating Objects

This chapter contains these topics:

- Entering and Deleting Points
- How Creation Commands Work with ROD Objects
- Creating Rectangles
- Creating Polygons
- Creating an Arc
- Creating Paths
- Creating Circles
- Creating Ellipses
- Creating Donuts
- Creating Geometric Wires
- Creating Instances
- Creating Pins
- Creating Labels
- Creating Vias
- Creating Multipart Paths
- Creating Fluid Guard Rings
- Creating MPP Guard Rings
- Creating Slots
- Creating P&R Objects
  - Creating Blockages
  - Creating Rows

© 2020 All Rights Reserved.

- Creating Custom Placement Areas
- Creating Clusters
- Creating Track Patterns
- Creating P&R Boundary
- Creating Snap Boundary
- Creating Area Boundary
- Creating Cluster Boundary

■ Creating Groups

© 2020 All Rights Reserved.

Entering and Deleting Points

When you create objects, you enter and delete points to create the shape you want.

To enter points for a shape or wire:

➤ Click the left mouse button at the starting point for the object.

To complete a circle, rectangle, ellipse, or a donut,

➤ Click again to complete the last point of the shape.

To complete a path, polygon, or wire:

➤ Double-click to enter the last point or click and press the `Return` button.

To delete a point:

➤ Press `Backspace`.

Entering Points in the CIW

You can enter points by typing coordinates into the command interpreter window (CIW) as follows:

1. Choose *Create – Shape – Rectangle*.
   You are prompted to enter the first corner of the rectangle.
2. Type a coordinate pair in the CIW. Use a colon to separate the coordinates and press `Enter`. For example:

   ```
   -0.2:-9.0
   ```

3. When you are prompted for the opposite corner of the rectangle. Type another coordinate pair in the CIW and press `Enter`. For example:

   ```
   0.5:20.0
   ```

If you are creating a polygon, enter the same coordinates again to indicate the last point.

© 2020 All Rights Reserved.

Entering Points with the Keypad

You can use the keypad to move the pointer and to enter points. This can be helpful if, for example, you want to enter a point just a few units away from the last point.

Using the Enter Points Form

You can also use the *Enter Point*s form to specify the points to create an object. This is especially useful when you want to create an object or a list of objects at specific coordinates in the design window.

1. Choose *Create – Shape – Rectangle*.
2. Choose *Tools – Enter Points*.

© 2020 All Rights Reserved.

The Enter Points Form opens.

3. Enter points in the *XY* field. For example:

   `(0 10) (20 30)`

4. Click *OK*.

A rectangle is created with `(0 10)` as the first corner and `(20 30)` as the corner that is diagonally opposite to the first corner.

You can also use the *X*, *Y*, *dX*, and *dY* fields on the status toolbar to specify the points to create an object. For more information, see Status Toolbar.

Video

For details on using the Enter Points and Status toolbar to specify points for create and edit commands, see the video *Enter Points for Create and Edit Commands*.

Note: Access to this video will depend on the availability of a web browser and a Cadence Online Support account.

Note: The gravity settings are not applicable if Enter Points form is used to specify points.

How Creation Commands Work with ROD Objects

The following tables summarize the level of support for how creation commands work on relative object design (ROD) objects in the current release. A ROD object contains information about the object, including its name and database ID.

Creating a ROD object lets you:

- Specify connectivity to associate the object with other objects on the same net by typing the net name in the *Net Name* field

© 2020 All Rights Reserved.

- Give the object a ROD name by either using the default database name or typing another name in the *ROD Name* field

ROD names can have spaces in them. For example, `rectangle 1a` is a valid ROD name. Only one ROD object can be created and assigned to a net at a time. You cannot enter a series of names in the *Net Name* and *ROD Name* fields. Once the object is created, you can edit the object by typing ROD function commands in the CIW. These commands let you:

- Access ROD objects by name through all levels of hierarchy
- Access ROD objects' handle values through all levels of hierarchy
- Align ROD objects to each other or to specific coordinates
- Assign names to unnamed rectangles, polygons, lines, and paths
- Create complex parameterized cells

See the *Virtuoso Relative Object Design User Guide* for information about ROD objects.

Using commands that are not fully supported for ROD objects could cause the objects to lose the ROD information associated with them, changing the objects into ordinary shapes.

Note: ROD functionality is not implemented for mosaics.

Create Command	**Degree of ROD Support**
Rectangle... r	Create and name new rectangles as ROD objects using the *Create Rectangle* form.
Polygon... P	Create and name new polygons as ROD objects using the *Create Polygon* form.
Path... p	Create and name paths as ROD objects using the *Create Path* form.
Multipart Path...	Create and name new multipart paths as ROD objects using the *Create Multipart Path* form. This form lets you choose a template from your technology file, load templates from an ASCII file, and save form values as a template in your technology file (if you have write permission) or in an ASCII file.
Label... l	You cannot create a label as a ROD object. However, you can make an existing label a ROD object by assigning it a name with the `rodNameShape` function.

© 2020 All Rights Reserved.

Instance... i	An instance is automatically a ROD object because it has a unique name. The ROD object name is the same as the instance name.
Pin... ^p	Create and name new pins as ROD objects using the *Create Pin* form.
Pins From Labels...	You cannot create a pin from a label as a ROD object. However, you can make an existing pin a ROD object by assigning it a name with the `rodNameShape` function.
Circle	Circles are supported as ROD objects. However, there is no user interface for naming a circle; use the `rodNameShape` function.
Ellipse	Ellipses are supported as ROD objects. However, there is no user interface for naming an ellipse; use the `rodNameShape` function. An ellipse ROD object has only bounding box handles.
Donut	Donuts are supported as ROD objects. However, there is no user interface for naming a donut; use the `rodNameShape` function. A donut ROD object has only bounding box handles.
Layer Generation...	You cannot create a shape using a layer generation operation as a ROD object. However, you can make a generated shape a ROD object by assigning it a name with the `rodNameShape` function.

Using the rodAutoName Environment Variable

You can use the environment variable `rodAutoName` to set system default values in the Create Pin, Create Rectangle, and Create Polygon forms to:

- Turn on *Create as ROD object* so all shapes are created as ROD objects
- Make the *Name* field editable, allowing you to either use the default name or type another name

The following example sets the Create Rectangle, Create Pin, and Create Polygon forms to create shapes as ROD objects. In the Command Interpreter Window (CIW), type:

```
envSetVal("layout" "rodAutoName" 'string "rectangle pin polygon")
```

To turn off *Create as ROD object* in one or all of the forms listed in the `rodAutoName` variable, do one of the following:

- Turn off *Create as ROD object* in any of the forms
- Retype the `envSetVal` command, excluding the type of object you no longer want to create as ROD objects

© 2020 All Rights Reserved.

For example, if you turn off *Create as ROD object* in the Create Polygon form, the following is true for `rodAutoName`:

```
envGetVal("layout" "rodAutoName")
"rectangle pin"
```

You can also set the `rodAutoName` variable in your `.cdsenv` file, so the layout editor defaults to the options you want at startup. The syntax to use in your `.cdsenv` file is:

```
layout rodAutoName string "rectangle pin polygon"
```

Creating Rectangles

The *Create Rectangle* command enables you to create rectangular shapes. By using this command, you can also create rectangles with slots.

To create a rectangle:

1. Select the layer you want from the Layers Panel.
2. Choose *Create – Shape – Rectangle* [`r`].
3. Press `F3`.

© 2020 All Rights Reserved.

The Create Rectangle Form opens.

4. In the *Net Name* field, type the name of the net you want the rectangle to be associated with.

5. Expand *ROD* and select the *Create as ROD object* check box if you want to create a ROD object rectangle; otherwise, the rectangle is created as a polygon.

   In the *Name* field that now becomes editable, change the default rectangle name, if required.

6. Expand *Slotting* and select the *Enable* check box if you want create the rectangle with slots.

7. In the *Layer* list, select an LPP.

   This list is available only if the *Create as ROD object* check box is selected in the *ROD* section.

© 2020 All Rights Reserved.

8. Specify *Width/Length* or *Width/Area*.

   If you deselect *Specify width, area*, the text fields available are *Width* and *Length*. If you select *Specify width, area*, the *Width* and *Area* fields are available.

9. Specify the slot dimensions in the *Width* and *Length* or *Width* and *Area* fields.

10. Specify the spacing in the *Column*, *Row*, and *Edge* fields.

11. In the design display area, click to specify the first corner of the rectangle. Alternatively, you can specify the points of the rectangle using the *Enter Points* form. For more information, see Enter Points Form.

12. Click again to specify the corner that is diagonally opposite to the first corner that you specified.

    You cannot create a rectangle with zero area. If you click twice at the same point to specify the two corners of the rectangle that you are creating, a warning message is displayed in the CIW.

    If you are creating a rectangle with slots, ensure that you create a rectangle large enough to contain the slots with the specified dimensions; otherwise, slots are not created. Slots are added dynamically as you create the rectangle.

Dynamic Slot Generation While Creating the Rectangle

Resultant Rectangle with Slots

Creating Polygons

To create a polygon,

1. Select the layer you want.

© 2020 All Rights Reserved.

2. Choose *Create – Shape – Polygon* [`Shift-p`]. Press F3 to open the form.

   The Create Polygon Form appears.

3. Choose a value from the *Snap Mode* cyclic field.

   Snap Mode settings are illustrated here. If you chose an L90 mode, you can switch to the other L90 mode by clicking the middle mouse button. Keep clicking the middle mouse button to toggle between *L90XFirst* and *L90YFirst.*

4. In the *Net Name* field, type the net name you want the polygon to be associated with.

5. If you want to create a ROD object polygon, select *Create as ROD object*.

   The *Name* field becomes editable, allowing you to change the name from the default name that is displayed. If you do not change the name, the default name is assigned to the polygon.

6. Click to enter points. As you enter each point, a dotted line shows how the layout editor will close the polygon if you stop at that point.

   Alternatively, you can specify the points of the polygon using the *Enter Points* form. For more information, see Enter Points Form.

7. Double-click to finish the polygon or click and press `Return`.

Creating an Arc

To create an arc,

1. Select the layer you want.

2. Choose *Create – Shape – Polygon* [`Shift-p`]. Press F3 to open the form.

   The Create Polygon Form appears.

3. Click *Create Arc* in the form.

© 2020 All Rights Reserved.

4. Click to indicate the start point, the end point, and a radius point on the arc. Alternatively, you can specify the points of the arc using the *Enter Points* form. For more information, see Enter Points Form.

If you have already entered some points of a polygon, the last point you clicked is the start point of the arc.

Note: You can also use the Modify Corner command to curve the corner of a polygon.

Creating Paths

To create a path:

1. Select the layer on which you want to create the path from the Layers Panel.
2. Choose *Create – Shape – Path*.
3. Press `F3`.

© 2020 All Rights Reserved.

The Create Path Form opens.

4. On the Path Options tab, in the *Width* field, type the width for the path.

   The default path width is the `minWidth` for the entry layer defined in the technology file.

5. Select the *Fixed Width* check box if you want a fixed width for the path.

   If this check box is selected, the specified width is applied to all the segments of the path, even when you change layers.

6. In the *Justification* list, select the edge of the path you want to offset.

© 2020 All Rights Reserved.

The path justification settings control where you can enter the points—along the centerline or along the edge of a path.

7. Type a value in the *Offset* field if you want to offset the path.

For example, if you type a value of `0.5`, the path is created at a distance of `0.5` microns from the points you enter.

8. Select the required end type for the path.

The path *End Type* settings change the shape and spacing of path ends. If you set the end type as *variable*, specify the *Begin Extension* and *End Extension* values.

9. In the *Net Name* field, type the net name you want the path to be associated with.

10. Select the *Create as ROD Object* check box if you want to create a ROD object path; otherwise, the path is created as a polygon.

In the *Name* field that now becomes editable, change the default path name, if required.

11. Select the *Enable Metal Slotting* check box if you want to create a path with slots.

This makes available the Slot Options tab.

© 2020 All Rights Reserved.

12. Select a value from the *Snap Mode* list.

13. Set *Snap Mode* to *anyAngle* and select the *Acute Angle* check box if you want to create paths with acute angles.

14. Click the *Slot Options* tab.

15. In the *Slot Layer* list, select an LPP.

This list is available only if the *Create as ROD Object* check box is selected on the *Path Options* tab.

16. Select *Width/Length* or *Width/Area*.

If you select *Width/Length*, the text fields available are *Width* and *Length*. If you select *Width/Area*, the *Length* field is replaced with the *Area* field.

17. Specify the slot dimensions in the *Width* and *Length* or *Width* and *Area* fields.

© 2020 All Rights Reserved.

18. In the *Between Slots*, *Between Slot Rows*, *Edge to Slot*, and *Stagger* fields, specify values as required.

19. In the design display area, click to specify the path endpoints. Segments are created when you change the direction of the path.

Alternatively, you can specify the points of the path using the *Enter Points* form. For more information, see Enter Points Form.

20. Double-click in the design display area or press `Enter` to finish creating the path.

If the path you create is self-intersecting, every single-click after a self-intersection generates a message in the CIW, though you can continue to create the path. You can cancel the current and previous points by pressing the `Backspace` key or finish creating the path by double-clicking or pressing `Enter`. If you double-click or press `Enter`, the path is automatically truncated to a valid path. All the segments or points after the first intersection point are removed so that the remaining path shape is valid. A message appears in the CIW indicating the same.

© 2020 All Rights Reserved.

Even if there are multiple intersection points, the path is truncated to the first self-intersection point, as shown below.

Self-intersecting path

Path truncated at the first point of intersection to create a valid path

If you are creating a path with slots, ensure that you create a path wide enough to contain the slots with the specified dimensions; otherwise, slots are not created. Slots are added dynamically as you create the path.

Dynamic Slot Generation While Creating the Path

Resultant Path with Slots
(No Stagger)

© 2020
All Rights Reserved.

You can also create slots on *diagonal* paths, as shown in the figure below. Slots are not created on *anyAngle* paths.

Note: Slots are not supported in geometric wires created from paths. For creating a wire by using a path, see the Virtuoso Interactive and Assisted Routing User Guide.

Controlling the Display of Paths

You can display path borders and centerlines, borders only, or centerlines only.

1. Choose *Options – Display* [e].

   The Display Options Form opens.

2. Select one of the following from the *Path Display* list:

   - *Borders and Centerlines*: Displays both the path centerline and its outer edges.
   - *Borders only*: Displays only the path borders and eliminates the centerline display.
   - *Centerlines only*: Displays only the centerline.

© 2020 All Rights Reserved.

3. Click *OK*.

A path displayed with *Borders and Centerlines* set on.

The same path with *Borders only* set on.

The same path with *Centerlines only* set on.

Creating Circles

To create a circle,

1. Select the layer you want from the Layers Panel.
2. Choose *Create – Shape – Circle*.
3. Press `F3`.

   The Create Circle Form opens.

4. Select the shape type for the circle that you want to create.
5. You can specify the conic sides and radius for the circle.

© 2020
All Rights Reserved.

6. Click where you want the edge of the circle.

Click the center. | Move the mouse. | Click to finish the circle. | The finished circle.

Creating Ellipses

You create an ellipse by creating a rectangle that surrounds the ellipse.

To create an ellipse,

1. Select the entry layer you want.
2. Choose *Create – Shape – Ellipse*.
3. Click where you want the first corner of the ellipse. Alternatively, you can specify the points of the ellipse using the *Enter Points* form. For more information, see Enter Points Form.
4. Click where you want the opposite corner of the ellipse.

Click the first corner. | Move the mouse. | Click the opposite corner. | The finished ellipse.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Create Ellipse` command does not support curved shapes.

© 2020 All Rights Reserved.

Creating Donuts

To create a donut,

1. Select the entry layer you want.
2. Choose *Create – Shape – Donut*.
3. Click where you want the center of the donut. Alternatively, you can specify the points of the donut using the *Enter Points* form. For more information, see Enter Points Form.
4. Click where you want the inner edge of the donut.
5. Click where you want the outer edge of the donut.

Click the center. | Click the inner edge. | Click the outer edge. | The finished donut.

Creating Geometric Wires

To create a geometric wire,

1. Choose *Create – Shape – Geometric Wire*.

   The Create Geometric Wire Form opens.
2. Select *Paths* or *Segments* to create the wire.
3. Fine-tune the width, justification, and begin and end style settings. Alternatively, you can specify the points of the wire using the *Enter Points* form. For more information, see Enter Points Form.
4. Click or area select a pin, via, path, or wire to the layer and attributes.
5. Continue to click at subsequent points of the wire.
6. Press Return or double-click to end the wire.

For information about how to create wire interconnects, see Creating Wires in the *Virtuoso Interactive and Assisted Routing User Guide.*

© 2020 All Rights Reserved.

Creating Instances

An instance is a database object that represents a master cellview. You can have several instances of the same cellview in a design. The *Create Instance* command lets you place an instance of a cell into the current cellview.

To place a cell inside the current cellview,

1. Choose *Create – Instance* from the menu bar or click the *Create Instance* icon on the *Create* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Create – Instance.*

   The Create Instance Form opens.

2. Fill in the *Library*, *Cell*, and *View* fields.

3. To override the automatic naming of instances, type one or more instance names in the *Names* field, leaving a space between names. By default, the layout editor automatically assigns names to instances, starting with `I1` (Instance 1) and continuing with `I2`, `I3`, and so forth. If the instances are mosaics, the default names are `M1`, `M2`, and so forth. If you override the assigned name, the first name in the form is assigned to the first instance you place. The next name is assigned to the next instance you place, and so forth. You can use the *Display Options* form to display instance names in the cellview.

   In case the design has EMH, you can specify the complete hierarchical path name (starting from the top module to the new instance to be created using the hierarchy delimiter in CDB name space) for specifying the location in EMH where the instance is to be created. An example of such a name is `modA‘modB‘15`, where A and B are the names of the module instances and 15 is the name of the newly created instance.

   Note: Use of the forward slash character (/) is not allowed in instances names. If used, it gets replaced with its octal representation `#2f`. Therefore, do not use a name such as `I1/I2` for an instance.

4. Optionally, you can create a mosaic or array of instances by typing values in the *Rows* and *Columns* fields.

5. To change the spacing between the rows or columns, type values for *Delta Y,X*.

   By default, the fields are set to the width and length of the original cell, so that the edges of each instance touch but do not overlap.

6. To create the instance as a physical only instance, click the *Physical Only* check box.

© 2020 All Rights Reserved.

7. To rotate instances click *Rotate* or, as you place the instance, click the middle mouse button.
8. To mirror the instance click *Sideways*, *Upside Down*. Alternatively, as you place the instance, press `Ctrl` and click the middle mouse button to mirror the instance *Sideways* or press `Shift` and click the middle mouse button to mirror the instance *Upside Down.*
9. Click in the design window to place the instance. Alternatively, you can specify the points where you want to place the instance using the *Enter Points* form. For more information, see Enter Points Form.

Note: If you specify the *Description* field in the Edit CDF form for a cell with CDF parameters, the description is displayed as a tooltip on the mouse over of that field.

Creating Pins

The *Create Pin* command lets you create pins in a cellview to connect the cellview to another cell in your design hierarchy. The *Create Pin* command also lets you define the connectivity of the pin, strong or weak, even while creating the pin. For defining or editing the connectivity of existing pins in the design, you can use the *Connectivity – Pins* command. Using the *Connectivity – Pins* command, you can define connectivity as strong connect, weak connect, must-join connect, or pseudo parallel connect. For more information about these connectivity types, see Overview of Pin Connectivity.

You can apply net expression properties to pins.

To create pins, choose *Create – Pin* [`Control-p`] or click the *Create Pin* icon on the *Create* toolbar. Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Create – Pin.* The Create Pin Form opens.

This topic covers the following:

- Creating Strongly Connected Pins
- Creating Weakly Connected Pins
- Placing Pin Arrays

For creating pins from labels, see Creating Pins from Labels.

Creating Strongly Connected Pins

1. Click *Create – Pin*. This opens the Create Pin Form.
2. Set Mode as *Manual*.

© 2020 All Rights Reserved.

3. Set *Connectivity* to *Strong*.
4. Specify one or more terminal names in the *Terminal Names* field. This is the name of the net to which the pin(s) to be created will connect. Separate multiple names with spaces. The behavior of the *Terminal Names* field depends on whether or not the *Keep First Name* is on. The following scenarios may occur when *Keep First Name* is off:
   a. If you specify a single name in the *Terminal Names* field, a single new terminal and pin are created. If the specified terminal already exists, a new pin is added to the terminal.
   b. If you leave the *Terminal Names* field blank, you are prompted to either specify a terminal name or to click an existing pin. If you click an existing pin, the terminal name of this pin populates the *Terminal Names* field in the form and a pin figure is added to the existing pin through strong connectivity.
   c. If you specify multiple names in the *Terminal Names* field, new terminals with the specified names are created if they do not already exist. New pins are created on each of the new terminals.
5. Select *Physical Only* to create the terminal only in the block domain.
6. Select *Keep First Name* for adding multiple pin figures to the same pin shape on the same terminal. This option works in conjunction with the *Terminal Names* field. The following scenarios may occur when *Keep First Name* is on:
   a. If you specify a single name in the *Terminal Names* field, the behavior is the same when *Keep First Name* is off, that is, a new terminal and pin are created, if they do not already exist. Multiple pin figures can be added to this new pin shape.
   b. If you specify multiple names in the *Terminal Names* field, only a single terminal is created with the first specified name. Multiple pin figures are added to the same pin shape and they are strongly connected.
7. Edit the X and Y Pitch values if required.
8. Select *Create Label* check box to attach a label showing the terminal name to the pin. See Setting Pin Label Text Style Options for more information.
9. Select *Create as ROD Object* to create the pin figure as a ROD object; else, the pin figure is created as a polygon.

   The *Name* field becomes editable, allowing you to update the name from the default that is displayed. The default name is derived based on the *Mode* you select.
10. Set the *Pin Shape*.

© 2020 All Rights Reserved.

When creating a rectangle, dot, polygon, or circle, the system prompts you to enter a point for the shape. To create a zero-size dot pin, click twice on the same point.

11. Select an *I/O Type* to indicate the direction of the signal into or out of the pin.

12. If you are creating a polygon pin, set the *Snap Mode* cyclic field to limit how the pointer snaps to the pin.

13. Select the *Access Direction* to specify the direction in which the routing tools can connect routing to the pin.

14. In the layout window, click to create the pin.

15. If you have selected the *Create Label* check box, click again to place the label on the pin figure.

16. If you are creating multiple pins, continue to click to create and place pins and labels. If *Keep First Name* is not selected or you specified a single name in the *Terminal Names* field, specify a terminal name in the form or click an existing pin to pick the terminal name.

17. Click *Cancel* in the *Create Pin* form or press Esc when you are done.

If you are creating multiple pins, pin figures are added to the same first specified terminal and the pin figures share the pin name. If you are creating a single pin, the pin is added to the specified terminal name. Pin names are auto-generated.

Creating Weakly Connected Pins

1. Click *Create – Pin*. This opens the Create Pin Form.
2. Set *Mode* as *Manual*.
3. Set *Connectivity* to *Weak*.
4. Select *Must Connect All Pins* to apply the LEF58_MUSTJOINALLPORTS property to all newly created pin terminals in the layout, which implies that the router must connect all pins of the selected terminal.
5. Specify one or more terminal names in the *Terminal Names* field. This is the name of the net to which the pin(s) to be created will connect. Separate multiple names with spaces. The behavior of the *Terminal Names* field depends on whether or not the *Keep First Name* is on. The following scenarios may occur when *Keep First Name* is off:

   a. If you specify a single name in the *Terminal Names* field, a single new terminal and pin are created. If the specified terminal already exists, a new pin is added to the terminal.

© 2020 All Rights Reserved.

   b. You cannot leave the *Terminal Names* field blank. If you do, you are prompted to specify a terminal name.

   c. If you specify multiple names in the *Terminal Names* field, new terminals with the specified names are created if they do not already exist. New pins are created on each of the new terminals.

6. Select *Physical Only* to create the terminal only in the block domain.

7. Select *Keep First Name* to use the same specified terminal name for creating multiple pins. If multiple terminal names have been specified, the first specified terminal name is used for creating the terminal and all pins connect to the same net. If this option is not selected, you will be prompted to specify a terminal name for each of the multiple pins you create.

8. Select *Keep First Name* for creating multiple pins on the same terminal. This option works in conjunction with the *Terminal Names* field. The following scenarios may occur when *Keep First Name* is on:

   a. If you specify a single name in the *Terminal Names* field, the behavior is the same when *Keep First Name* is off, that is, a new terminal and pin are created, if they do not already exist.

   b. If you specify multiple names in the *Terminal Names* field, only a single terminal is created with the first specified name. Multiple pins are added to the same terminal and they are weakly connected.

9. Edit the X and Y Pitch values if required.

10. Select *Create Label* check box to attach a label showing the terminal name to the pin. See Setting Pin Label Text Style Options for more information.

11. Select *Create as ROD Object* to create the shape pin as a ROD object.

The *Name* field becomes editable, allowing you to update the name from the default that is displayed. The default name is derived based on the *Mode* you select.

12. Set the *Pin Shape*.

When creating a rectangle, dot, polygon, or circle, the system prompts you to enter a point for the shape. To create a zero-size dot pin, click twice on the same point.

13. Select an *I/O Type* to indicate the direction of the signal into or out of the pin.

14. If you are creating a polygon pin, set the *Snap Mode* cyclic field to limit how the pointer snaps to the pin.

15. Select the *Access Direction* to specify the direction in which the routing tools can connect routing to the pin.

© 2020 All Rights Reserved.

16. In the layout window, click to create or place the pin.
17. If you have selected the *Create Label* check box, click again to place the label on the pin.
18. If you are creating multiple pins, continue to click to create or place pins and labels. If *Keep First Name* is not selected or you specified a single name in the *Terminal Names* field, you will be prompted to specify a terminal name in the form for each pin you create.
19. Click *Cancel* in the *Create Pin* form or press Esc when you are done.

New pins are created on the same terminal, which are weakly connected. When *Keep First Name* is off, multiple terminals are created. The first name in the *Terminal Names* field is assigned to the first terminal, second name to the second terminal, and so on. Pin names are auto-generated.

Note: When the connectivity is set to strong, and a terminal name is entered in the create pin form, a new pin is created. This new pin is strong connected to itself but weakly connected to other pins on the same terminal. For creating a new pin strongly connected to another pin in the terminal, click the existing pin shape on the canvas.

Placing Pin Arrays

A pin array is one line of pins with a single origin. You might create such an array if you have many nets bundled together into a bus. The pins in the array are placed for every bit of the bus and are spaced according to the *XPitch* and *YPitch* coordinates.

To place a pin array,

1. Choose *Create – Pin* [`Control-p`].

   The Create Pin Form opens.

2. In the *Terminal Names* field, type the net bus name as shown.

   baseName`<`*x*`:`*x*`>` For example, `NetA<0:5>`

3. In the *X Pitch* and *Y Pitch* fields, type the horizontal and vertical pitch for the array.
4. Move the pointer into the cellview window and place the first pin where you want to start the array. The rest of the pins are automatically placed at the designated X and Y locations for every bit of the bus. They are measured from the center point of the previously placed pin.

© 2020 All Rights Reserved.

Examples of Pin Arrays with Different X and Y Pitch

Terminal Names = N<0:3>
XPitch = 2
YPitch = 2

Terminal Names = A<0:3>
XPitch = 4
YPitch = 0

Setting Pin Label Text Style Options

To specify pin name characteristics,

1. In the *Create Pin* form, select *Create Label* and click the *Options* button.

   TheSet Pin Label Text Style Form opens.

© 2020 All Rights Reserved.

2. Select the font height and style.

3. Select the text options.
   - Turning *Drafting* on prevents the label from being rotated more than 90 degrees.
   - The *Overbar* option determines how the underscore character is displayed.

4. Select a valid layer name.
5. Select a valid layer purpose.

Note: You must select a valid combination of layer name and purpose. In case, it is not a valid combination, an error message will be displayed in CIW while creating the text display.

6. Select the justification for the pin name origin.

   The origin appears as a small square on the name when placed or selected.

© 2020 All Rights Reserved.

7. Select the orientation of the pin name.

   Once the pin is placed, an outline of the pin name appears.

The dotted line shows that the name will be attached to the pin.

Support for Orthogonal WSP Grid

While creating a pin, orthogonal WSP grid constraint defined in the technology file automatically adjusts the pin with a variable extension value so that the begins and ends points of a pin are on the grid.

The constraint group is specified in the technology file as shown below.

```
constraintGroups(
;( group                [override]            [definition]          [operator] )
;( -----   ---------- ------------              ---------- )
( "orthogonalWSPGridCG"         nil    nil     'and
spacings(
     ( orthogonalWSPGrid          "Metal1"   "M2WSP" )
     ( orthogonalWSPGrid          "Metal2"   "M5WSP" )
     ( orthogonalWSPGrid          "Metal3"   "M4WSP" )
    ) ;spacings
  ) ;orthogonalWSPGridCG
) ;constraintGroups
```


© 2020 All Rights Reserved.

If the orthogonal WSP grid constraints are part of a specific constraint group, you can select the constraint group from the *Default Wire Constraint Group* field in the *Wire Editing* section of the Layout Editor Options form.

Creating Labels

You can identify objects or portions of your design by labeling the objects. This section covers the following topics:

- Configuring Label Creation Options
- Label Creation Methods
  - Single-Click Method
  - Scan Line Method
  - Auto Step Method
  - Selection Method
- Label Creation Modes
  - Manual Mode
  - Auto Mode
- Labelling Overlapping Shapes

Configuring Label Creation Options

1. Choose *Create – Label* [l].

© 2020 All Rights Reserved.

The Create Label Form opens.

Note: When you run the *Create – Label* command, the *Create Label* form opens if the last setting in the form was *Manual*. If the last selected mode was *Auto*, the *Create Label* form does not open, but you are prompted to click or select a shape to create labels. The label is not dropped at the pointer location if you start the *Create Label* command in *Auto* mode because this command is not affected by the Infix mode setting.

2. Choose the method of generating labels.

© 2020 All Rights Reserved.

If you select *Auto* and enable the *From Interconnect* tab, ensure that the selected objects have connectivity information. If you select Auto, configure the required settings on the From Interconnect and From Instance tabs.

3. In the *Label (Pattern)* field, which is available only in the *Manual* mode, type the text string or the pattern for the label names.

See the examples of the pattern strings that can be specified in the *Label (Pattern)* field in Appendix A.

4. In the *Label Layer/Purpose* section, choose a layer and purpose for placing the label.

5. In the *Label Options* section:

a. From the *Font* list, choose a font for the label.

The figure below shows the available fonts.

b. In the *Height* field, specify a height for the label text.

c. From the *Justification* list, select the position of the label origin with respect to the label position.

© 2020 All Rights Reserved.

The figure below shows the available options that you can select.

d. Select the *Drafting* check box if you do not want the label text to be rotated by more than 90 degrees.

e. Select the *Overbar* check box if you want underscore characters in the label text strings or patterns to be displayed as overbars.

If the *Overbar* check box is off (default), the software displays underscore characters (_) as part of the text string. If the *Overbar* check box is on, the software interprets underscore characters in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the following examples.

Property Name	Appears in Design Window As
_abcde	$\overline{abcde}$
ab_cde	$ab\overline{cde}$
_abc_de	$\overline{abc}de$
ab_cd_ef_gh_ij	$ab\overline{cd}ef\overline{gh}ij$

f. Select the *Keep Label* check box if you want the label or pattern specified in the *Label (Pattern)* field to be available after all the labels or the ones generated from a sequence have been used up.

© 2020 All Rights Reserved.

g. Select the *Attach* check box if you want to attach the label to an object.

h. Select the *Expand* check box if you want to generate separate label names from the pattern specified in the *Label (Pattern)* field.

6. In the *Scan Line* section:

a. Select the *Label Selected Objects* check box if you want only the selected objects to be labeled while using the Scan Line Method.

b. Ensure that the *Auto Zoom Out* check box is selected to enable automatic zooming and panning if the scan line extends beyond the current display area.

7. In the *Auto Step* section, select the *Step* check box if you want labels to be created automatically with the fixed, specified distance between consecutive labels.

8. Choose a value from the *Snap Mode* list to control the direction in which you can move the scan line while using the Scan Line Method and the Auto Step Method (*Custom*).

9. To rotate or mirror a label as you place it, set *Drafting* off, then click *Rotate*, *Sideways*, or *Upside Down* as required before placing the label.

Alternatively, you can click the middle mouse button to replicate *Rotate,* press `Ctrl` and click the middle mouse button to replicate *Sideways*, or press `Shift` and click the middle mouse button replicate *Upside Down.* Each time you click a button on the form or click the middle mouse button, the label is further rotated or mirrored. Click in the cellview to place the label.

After specifying various options in the *Create Label* form, as you move the pointer to the design display area, you see the *ghost image* of the label text in the *Manual* mode. This image represents how the label text will look when you place the label. The ghost image moves along with the pointer until you place the label. In the *Auto* mode, the ghost image of label text appears only when you hover the pointer over shapes that have connectivity information.

Note: You can add a label to members of all types of figure groups. For modgen figure group, you can add a label to modgen figure group if you are editing in group for that modgen.

Label Creation Methods

To place the labels, you can use one of the following methods:

- Single-Click Method (available in *Manual* and *Auto* modes)
- Scan Line Method (available in *Manual* and *Auto* modes)
- Auto Step Method (available in *Manual* mode)

© 2020 All Rights Reserved.

- Selection Method (available in *Auto* mode)

Single-Click Method

By using this method, you label the objects one by one.

➡ With the label attached to the pointer, click the object you want to label and the label will be placed at the point of click.

Auto Mode Manual Mode

Scan Line Method

By using this method, you label multiple objects in one go. This method is useful if the specified pattern in the *Label (Pattern)* field generates multiple labels in a sequence. For example, the label `GDD` will label only the first object even though the scan line may intersect more objects. The scan line assigns labels to only as many objects as the number of labels generated by the specified pattern. Consider the pattern `A<0:2>` with the *Expand* option selected. Three labels will be generated: `A<0>`, `A<1>`, and `A<2>`. If the scan line intersects five objects, only three of them will be labeled .

To generate labels by using the scan line:

1. With the middle mouse button, click and drag the pointer to draw a line scanning across the objects you want to label.
2. Release the mouse button at the end point of the scan line.

© 2020 All Rights Reserved.

Labels are placed at the intersection of the scan line and the objects. You can configure the direction in which you can move the scan line by setting the *Snap Mode* option, which is set to *orthogonal* by default.

Labeling Selected Objects

In the *Manual* mode, use the *Scan Line* method to label only the selected objects. You can select the objects before or after starting the *Create – Label* command.

1. Select the objects you want to label.

2. Choose *Create – Label*.

   The Create Label Form opens.

3. In the *Label (Pattern)* field, type the label pattern.

© 2020 All Rights Reserved.

4. In the *Scan Line* section, select the *Label Selected Objects* check box.
5. With the middle mouse button, click and drag a scan line that intersects the selected and unselected objects.

6. Release the mouse button.

   Only the selected objects are labeled.

7. Press `Esc` or click *Cancel* in the form to finish creating labels.

Only as many labels from the specified sequence in the *Label (Pattern)* field are used as there are selected objects.

In the *Auto* mode, you can similarly use the *Scan Line* method to label ed or post-selected objects with connectivity information and ed or post-selected instances.

Labeling Objects Out of the Design Display Area

In the scan line method, since you use the middle mouse button to draw the scan line, you cannot use the mouse scroll wheel to zoom in or zoom out of the design while creating the scan line. The `Tab` key is also unavailable to pan through the design. Therefore, in a scenario where there is a large number of objects to be labeled, some of which are out of the current display area, ensure that the *Auto Zoom Out* check box in the *Scan Line* section is selected. As you click and drag the scan line beyond one of the edges of the design display area (to allow the scan line to intersect objects currently not in view), the *Auto Zoom Out* option automatically zooms out the design to make the other objects to be labeled visible. As you continue to drag the scan line, the design keeps zooming out and panning automatically. When you release the middle mouse button to complete the scan line, the design returns to the original zoom level and labels are added to al objects that were intersected by the scan line. If you do not select this check box, no zoom out occurs, though the design pans as you drag the scan line beyond a design display area edge.

© 2020 All Rights Reserved.

Auto Step Method

This method can be used only in the *Manual* mode.

To create labels by using the step method:

1. Choose *Create – Label*.

   The Create Label Form opens.

2. In the *Label (Pattern)* field, type the label pattern.
3. In the *Auto Step* section, select the *Step* check box.
4. From the *Step* list, if you select *Input X&Y*:

   a. Specify values in the *X* and *Y* fields.

   All the generated labels attach to the pointer and are visible as per the specified spacing.

© 2020 All Rights Reserved.

b. Click to place the labels.

If the *Keep Label* check box in the *Label Options* section is selected, the label sequence remains attached to the pointer after you click to place the label.

5. From the *Step* list, if you select *Custom*:

The first label attaches to the pointer.

a. Set up the *Snap Mode* setting.

b. Click to place the first label in the design display area.

The remaining generated labels attach to the pointer, as shown below.

You can move the pointer around to determine the position of the second label.

c. Click to place the second label.

© 2020
All Rights Reserved.

The remaining labels are placed by using the same spacing as between the first and second placed labels and along the same direction as the first two labels.

6. Press `Esc` or click *Cancel* in the form to finish creating labels.

Selection Method

By using this method, you can fully or partially select the shape to be labeled , before or after starting the *Create – Label* command. This method works only in the *Auto* mode. Labels are generated for fully or partially selected shapes that have connectivity information or are instances.

Label Creation Modes

- Manual Mode
- Auto Mode

© 2020 All Rights Reserved.

Manual Mode

Each generated label is placed with a click, at the click location. For example, if you specify `A<0:3>` in the *Label (Pattern)* field, the labels are incrementally generated from `A<0>` through `A<3>` and are placed one-by-one at the click location.

Attach: With the *Attach* option on, label is attached to the object you click. If there is no object at the click point, you will be prompted to click an object to attach the label to the object.

© 2020 All Rights Reserved.

When attaching a label to an object, a dashed line extends from the label to the object and disappears when you click the object.

A dashed line shows the object to which you attach the label.

Label (Pattern): As you place a single label or multiple labels generated from a sequence, they get consumed from the *Label (Pattern)* field. For example, if `A1 B2 C<0:3>` is the specified label pattern, after placing `A1` and `B2`, the *Label (Pattern)* field displays only `C<0:3>`. The *Label (Pattern)* field shows `C<0:3>` until you place `C<0>` through `C<3>`. With the *Keep Label* option selected by default, the pattern specified is retained in the *Label (Pattern)* field and can be used to generate same set of more labels. If the *Keep Label* option is off, the *Label (Pattern)* field becomes empty after you place all the labels generated from the sequence, and you are prompted to specify a label text. You can specify the same or different label text.

© 2020 All Rights Reserved.

Auto Mode

From Interconnect

© 2020
All Rights Reserved.

From Instance

Label is generated based on the connectivity information of the object you want to label or from the instance name or instance master you want to label. Label is placed at the click location and is automatically attached to the object you are labelling.

Labeling Fully or Partially Selected Shapes: In the *Auto* Mode, you can or post-select the shapes to be labeled . In both the selection modes, you can select shapes either fully or partially. The full and partial scenarios described here are the same for both ion or post-selection modes.

© 2020 All Rights Reserved.

- Full Selection: The label (origin) is automatically placed at the center of the shape.

  Ensure that the *Full Select* icon is selected on the *Options* toolbar. If not, press `F4` to enable it.

  - In the case of shapes such as rectangles, circles, and polygons, the label is placed at the center of the shape. In the case of a fully selected polygon, if the center of the polygon is outside the polygon, the label is placed on the midpoint of the longest edge of the polygon.

  - For a donut, the label is placed at the midpoint of the longest edge of a donut-equivalent polygon.

© 2020 All Rights Reserved.

- For pathsegs, the label is placed at the midpoint of the centerline.

Label is placed at the midpoint of each pathseg.

- For paths, the label is placed on the midpoint of the middle segment's centerline.

Middle Segment of the Path

Middle Segment of the Path

- Partial Selection: Ensure that the *Partial Select* icon is selected on the *Options* toolbar. If not, press `F4` to enable it.

  Partial ion or post-selection of a shape in the *Auto* mode labels the shapes in the following way:

© 2020 All Rights Reserved.

- The label is placed at the midpoint of a selected edge.

- With multiple selected edges, the label is placed at the midpoint of the longest selected edge. If two or more selected edges are of the same length, label is placed on the midpoint of a randomly selected same-length edge.

- If multiple non-consecutive vertices are ed (no edge selected), the label is placed at the vertex closer to the midpoint of the shape.

ed Vertices

- If a combination of vertices and edges are selected, the label is placed at the midpoint of the longest selected edge.

© 2020 All Rights Reserved.

- Partial selection of a path or pathseg places the label origin at the end of the selection.
  - If the begin or end style is truncate, the label is placed on the centerline where it meets the edge.

Path with End Style Truncate

Pathseg with Begin and End Style Truncate

  - If the begin or end style is extend, the label is placed at the end of the centerline.

Path with End Style Extend

Pathseg with Begin and End Style Extend

Note: If an object is non-selectable, the label is not created on the object in both Manual and Auto label creation modes.

Labelling Overlapping Shapes

In the Manual Mode

- Label Layer/Purpose = *Use current entry layer* or *Select layer*: If the click point or intersection point of scan line lies on overlapping shapes, a single label is created at the click or intersection point. The label is created on the layer-purpose pair that is the

© 2020 All Rights Reserved.

current entry layer in the *Layers* assistant or on the one that you select from the *Select layer* list. Single label string is consumed from the *Label (Pattern)* field.

Move the shapes to confirm that only one label is created.

- Label Layer/Purpose = *Use same layer as shape, select purpose* or *Auto (Using stampLabelLayers section from technology file)*: If the click point or intersection point of scan line lies on overlapping shapes, multiple labels are created at the click or intersection point. One unique label is created on each layer-purpose pair. This is shown in the example below in which the *Label Layer/Purpose* is set to *Use same layer as*

© 2020 All Rights Reserved.

shape, select purpose. The two shapes in blue are on `metal1` layer and the red shape is on `metal2` layer.

If you move the shapes, you will notice that the labels are created on the same layers as shapes. However, since there are two shapes on `metal1` layer, only one label is generated for the two shapes.

In Auto Mode

- Label Layer/Purpose = *Use current entry layer* or *Select layer*: Separate labels are generated for overlapping shapes. The label name for each corresponds to the net name associated with the respective overlapping shape. Multiple labels with different label names overlap at the click location. The overlapping labels attach to the respective overlapping shapes by default. The labels are created on the layer-purpose pair that is

© 2020 All Rights Reserved.

the current entry layer in the *Layers* assistant or on the one that you select from the *Select layer* list.

Move the shapes to see that each shape has been labeled with its net name and all labels are created on the same layer-purpose pair.

If the shape you click or the shape that intersects the scan line does not have connectivity information, no label is assigned to the shape and a warning message is generated in the CIW.

- Label Layer/Purpose = *Use same layer as shape, select purpose* or *Auto (Using stampLabelLayers section from technology file)*: Separate labels are generated for overlapping shapes with the label names corresponding to the net name associated with

© 2020 All Rights Reserved.

the respective overlapping shapes. The labels are created on the same layer as the shapes if you select *Use same layer as shape, select purpose*, as shown below.

Move the shapes to see that each shape has been labeled with its net name and all labels are created on the same layer-purpose pair.

Note: Small labels, less than `4` pixels, are not displayed in the design. To view small labels, you can set the drawSmallLabelOrigin environment variable to `t`.

You can create and attach a label to an object inside a figgroup if you are in EIP mode or the transparent group option is ON. The scenarios to create and attach a label to an object inside a figgroup are listed below:

- If you create a label on an object inside the figgroup and attach it to the target object.
- If you click an empty area outside the figgroup and attach the label to a target object inside the figgroup.

© 2020 All Rights Reserved.

Creating Vias

The *Create Via* command lets you connect objects on two layers within a cellview. This topic covers the following sections:

- Types of Vias
- Parameter Caching
- Installing a cdsVia Device in the Technology File
- Setting Up Via Creation Options
- Creating Vias Manually
  - Creating a Single Via
  - Creating a Stack Via
- Creating a Via Automatically in Single and Stack Modes
  - Creating a Via Automatically on Intersecting Shapes
  - Automatically Placing a Via Based on Connectivity Information of the Intersecting Shapes
  - Auto and Fast Edit Mode Options
- Setting the Technology to Create Via
  - Specifying the Calculation Modes for Via Parameters
  - Supported Parameters for Custom Via
  - Specifying viaEngineContext User Type
- Parameter Caching
  - Supported Constraints for Local Interconnect Vias
  - Supported Constraints for Auto Mode and Fast Mode Vias
- Support for Voltage-Dependent Rules

Types of Vias

The various types of vias are OA standard, cds, syContact, Pcell, and fixed vias.

© 2020 All Rights Reserved.

Vias are defined in the `viaDefs` section of the technology file. The types of via definitions are:

- Standard via definition (`standardViaDefs`)
- Custom via definition (`customViaDefs`)
- Standard via variant (`standardViaVariants`)
- Custom via variant (`customViaVariants`)

Vias must exist in the technology file in order for them to appear in the *Via Definition* cyclic field in the *Create Via* form. In both the *Create Via* and the *Edit Via Properties* forms, viaDefs are listed in the order as they are defined in the `validVias` list in the technology file.

For information about adding customized vias to the technology file, see Technology File Via Definitions and Via Specifications in the *Virtuoso Technology Data ASCII Files Reference*. For information about accessing database information about via definitions, see the *Virtuoso Technology Data SKILL Reference*.

Important

Always use the *Create Via* form to place vias in a cellview or add them by using the *Select Via form* while creating wires. Do not use the *Create Instance* form to create vias because these will not be interpreted as vias by the OpenAccess.

Parameter Caching

When you first open the *Create Via* form, the parameter values for the respective viaDefs are seeded in the form from the technology database values, based on the values specified in the Create Via section in the *Layout Editor Options* form.

While creating a via or via stack, you can change the parameter values for a viaDef. If you successfully create a constraints-valid via or via stack using the updated parameters, the parameter set is cached for that viaDef. The next time you access the form in the same `virtuoso` session with the same viaDef and layer/width values, the cached parameter set is used. Each time you create a via with a different viaDef or different parameter values, the cache is created or updated, respectively.

The cached parameter values are validated against existing constraints before you can use the parameter set to create a via or via stack. If the parameters are not valid, they are recalculated based on enclosure calculation settings.

Similarly, if while creating a via, you manually update the enclosure values in the *Create Via* form, the values are cached if they comply with the minimum rules. The manually specified

© 2020 All Rights Reserved.

enclosure values are not re-calculated when you update other viaDef parameters, as long as the enclosure values do not violate the minimum rules.

Note: You cannot create a custom or standard via that will create an invalid geometry.

Installing a cdsVia Device in the Technology File

To install a `cdsVia` device in the technology file by using the *Install Device* form:

1. In the CIW, choose *Tools – Technology File Manager*.

   This opens the *Technology Tool Box*.

2. In the *Utilities* section, click *Install Device*.

© 2020 All Rights Reserved.

This opens the *Install Device* form.

3. From the *Technology Library* list, select the technology library in which you want to create the `cdsVia` device.
4. From the *Device Type* list, select *Via*.
5. In the third section of the form, select the *cdsVia* option.
6. In the first section, in the *Name* field, specify the name of a new via device or select an existing via device from the list.

   For the purpose of an example, specify the name as `cdsViaM1M2`.

© 2020 All Rights Reserved.

The form updates, as shown below.

Tip

To delete an existing via device, select the via name from the *Name* list and click the *Delete* button.

7. In the second section of the form, with the *Layers* check box selected, select the required layers in the *Layer 1*, *Contact*, and *Layer 2* lists. As an example, set the following:

   a. From the *Layer 1* list, select *Metal1 dg*.

   b. From the *Contact* list, select *Via1 dg*.

   c. From the *Layer 2* list, select *Metal2 dg*.

8. In the second section of the form, select the *Rule* check box.

   Ensure that all fields are populated with values.

   Notice that when you select *Rule* check box, the *Rules Browser* window opens. If you place the pointer in any field, an arrow in the Rules Browser window indicates the physical area where the rule applies, as shown below.

© 2020 All Rights Reserved.

9. In the second section of the form, select the *Parameters* check box.

   Ensure that all fields are populated with values. The *Abutment Class* field is optional for a `cdsVia` device.

© 2020
All Rights Reserved.

10. Click the *OK* or *Apply* button.

If you click *Apply*, a message appears in the CIW to let you know if there are any errors or whether the `cdsVia` device has been created. If the `cdsVia` device already exists in the technology file, you are prompted to confirm if you want to overwrite the existing `cdsVia` device.

If you click *OK*, in addition to the message that appears in the CIW, you are prompted to confirm if you want to save the technology library to the disk. If you click *Yes*, the *Install Device* form closes.

You can check the `DEVICES` section in the technology file to confirm that the `cdsVia` device has been defined as per the values specified in the *Install Device* form.

To make the installed cdsVia device available in the *Create Via* form, you need to add the cdsVia device to the `validVias` constraint of the required constraint group in the technology file. You can do this by manually editing the technology file and then reloading it or by using the *Process Rule Editor* to edit the technology file. For information about editing the technology file by using the *Process Rule Editor*, see Adding the cdsVia Device to the validVias Constraint.

Note: *Process Rule Editor* is available from the *Constraint Manager* in VLS XL or higher tiers.

© 2020 All Rights Reserved.

Adding the cdsVia Device to the validVias Constraint

You can add the cdsVia device to the constraint group that contains the purpose-aware rules. For example, the user-defined `LPPtestCG` constraint group contains the following purpose-aware `minWidth` and `minSpacing` rules.

```
 ;( group       [override] )
 ;( -----       ---------- )
  ( "LPPtestCG" nil
    interconnect(
     ( validLayers  ( ( Metal1  highVoltage) ( Metal2  highVoltage)
( Metal3  highVoltage) ( Metal1  lowVoltage) ( Metal2  lowVoltage)
( Metal3  lowVoltage) Via1  Via2  ) )
    ) ;interconnect
    spacings(
     ( minWidth                  "Metal1"      0.3 )
     ( minWidth                  ("Metal1" "lowVoltage")    1.0 )
     ( minWidth                  ("Metal1" "highVoltage")   2.0 )
     ( minWidth                  "Metal2"      0.4 )
     ( minWidth                  "Metal3"      0.5 )
     ( minWidth                  "Via1"        0.3 )
     ( minWidth                  ("Via1" "lowVoltage")      0.4 )
     ( minWidth                  ("Via1" "highVoltage")     0.6 )
     ( minWidth                  "Via2"        0.3 )
     ( minSpacing                "Metal1"      0.3 )
     ( minSpacing                "Metal2"      0.4 )
     ( minSpacing                "Metal3"      0.5 )
     ( minSpacing                "Via1"        0.2 )
     ( minSpacing                ("Via1" "lowVoltage")       0.4 )
     ( minSpacing                ("Via1" "highVoltage")      0.6 )
     ( minSpacing                "Via2"        0.3 )
     ( minSpacing                "Via3"        0.4 )
     ( minSpacing                "Via4"        0.5 )
     ( minSpacing                "Via5"        0.6 )
      .....
      .....
    ) ;spacings
  ) ;LPPtestCG
```

To edit the technology file by using the *Process Rule Editor*:

© 2020 All Rights Reserved.

1. In VLS XL or higher tiers, choose *Window – Assistants – Constraint Manager*.

   The *Constraint Manager* assistant opens.

2. Click the *Process Rules Editor* icon .

   The *Process Rules Editor* window opens.

3. In the *Process Rules on* section, select the *Tech* option.

4. In the *Browser* section, expand the constraint group to which you want to add the cdsVia.

5. In the bottom section of the *Process Rule Editor*, click the *Value* tab.

© 2020 All Rights Reserved.

6. Scroll to select the installed cdsVia device, as shown below.

7. Click *Update*.

8. Click *Close*.

In the above example, the `cdsViaM1M2`, `cdsViaM2M3`, and `cdsViaM3M4` cdsVias are added to the `validVias` constraint in the `LPPtestCG` constraint group, as shown below.

© 2020 All Rights Reserved.

```
;( group       [override] )
;( -----       ---------- )
 ( "LPPtestCG" nil
   interconnect(
    ( validLayers  ( ( Metal1  highVoltage) ( Metal2  highVoltage) ( Metal3
highVoltage) ( Metal1  lowVoltage) ( Metal2  lowVoltage) ( Metal3  lowVoltage) Via1
Via2  ) )
    ( validVias     (cdsViaM1M2  cdsViaM2M3 cdsViaM3M4) )
   ) ;interconnect
 ) ;LPPtestCG
```

To instantiate the installed `cdsVia` device in your design, see Creating a cdsVia.

Setting Up Via Creation Options

Before creating a single, stack, or auto via, you can set up the viaDefs to be available for via creation. To set up the via creation options:

1. Choose *Create – Via* or click the *Create Via* icon on the *Create* toolbar.

© 2020 All Rights Reserved.

This opens the Create Via Form.

2. Click the *Options* button. This button is available in all the three modes: *Single*, *Stack*, *Auto*.

This opens the Create Via Options Form.

© 2020
All Rights Reserved.

3. Select the *Override Variant Parameters* check box to allow the update of via variant parameter values.

4. Select the *Extend Enclosure Beyond Overlap* check box if you want the via enclosures to extend beyond the overlap region of the intersecting shapes.

   In the *Single* and *Stack* mode, use this check box along with selecting the *Compute From Shape(s)* check box on the *Create Via* form. For more information about the *Compute From Shape(s)* check box, see Creating a Via Automatically in Single and Stack Modes.

© 2020 All Rights Reserved.

5. Select the *Fill Overlap With Enclosures* if you want the via enclosure to extend and cover the complete overlap region between two intersecting shapes in the *Auto* mode.
6. From the *Cut Class* list, select the cut class you want to be available by default for the respective viaDef/via variant in the *Single* mode. The selected cut class is also used by the *Stack* and *Auto* automatic via creation engines.
7. From the *Via Definitions* list, select a viaDef/via variant for each layer pair. The selected viaDef/via variant is also used by the *Stack* and *Auto* automatic via creation engines.

Creating Vias Manually

You can create vias manually, using the single and stack via modes described in this section:

- Creating a Single Via
- Creating a Stack Via

Creating a Single Via

Use the *Single* mode option on the to place a via in a cellview. By default, in the *Single* mode, the via width, length, row and column spacing, and the enclosure values are derived based on the calculation mode set in the *Via Parameters Calculation Mode* list on the Layout Editor Options Form. For standard and custom vias, you can update values for origins, enclosures, and via arrays by using the *System* tab on the form. See Creating a Via with Arrayed Via Cuts.

Note: Via parameter values are calculated such that via metal shapes are minimized.

Creating a Via with Arrayed Via Cuts

An array of vias is a group of vias with a single origin. You can create such an array to connect two wide paths or to create part of a large transistor. To create a cdsVia, see Creating a cdsVia.

To place a standard or custom via with arrayed via cuts:

1. Choose *Create – Via* or click the *Create Via* icon on the *Create* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Create – Via.*

   The Create Via Form opens.

© 2020 All Rights Reserved.

2. Select *Single* as the *Mode* for creating vias.

3. Click the *Options* button to set up the default viaDefs to be available during the via creation and to enable overriding of the via variant parameters. For more information, see Setting Up Via Creation Options.
4. To associate the via to a particular net, type the name of the net in the *Net Name* field.
5. To create the via as a ROD object, select *Create as ROD Object*. Specify a name for the via ROD object in the *Name* field.
6. Next, select a viaDef or a via variant from either the *Via Definition* list or the Via Definition Selection widget (open by clicking *Filter* button).

© 2020
All Rights Reserved.

a. To use the selection widget, click the *Filter* button to open the Via Definition Selection widget.

b. Specify a search expression in the *Filter* field using wild card characters.

c. Select a *Sort By* mechanism.

d. Select the layers in *From* and *To* drop-down lists for specifying the layers between which the viaDef or via variant should be available.

e. Select one of the filtered viaDefs or via variants from the list box. The via variants are listed as child nodes of the parent standard or custom viaDefs. Press `Esc` if you do not want to select a viaDef from the list box.

The Via Definition Selection widget closes and the selected viaDef or via variant displays in the *Via Definition* cyclic field. The *Via Type / Technology Library* read-only field updates to show whether the selected viaDef is a standard or custom viaDef or is a standard or custom via variant and the name of the technology library in which the viaDef is defined.

7. To work with a via variant, see Creating or Updating a Via Variant.

8. On the *System* tab, click *Reset Parameters to* if you want to revert to the default values. The reset value you select depends on whether or not *Compute Enclosures* is on. For information about various reset values, see Appendix A.

© 2020 All Rights Reserved.

9. Choose a *Justification* setting to change the via origin.

   When set to `offset`, *X* and *Y* are editable and display the offset in the X and Y direction, respectively.

10. If the `'cutClass` parameter is specified in the `viaSpacing` rule, select a cut class or a cut size from the *Cut Class* cyclic field.

    If a cut class for the viaDef/via variant in the *Via Definitions* cyclic field is set up in the Create Via Options Form, the value is selected by default in the *Cut Class* cyclic field in the *Create Via* form.

11. Specify values for *Width* and *Length* for the via cuts. If you select a *Cut Class*, *Width* and *Length* become non-editable.

12. Update the number of *Rows* and *Columns*, if required.

13. Specify new values for *Row Spacing* and *Column Spacing* if you want to change the spacing between the center of each via cut.

14. In the design window, click where you want to place the via array.

Note: If the size of a via cut is not an even number of the manufacturing grid unit, the via cannot be centered around the origin. In such a case, the via origin is changed to ensure that via cut shapes are on grid. The via cuts are shifted (up for vertical adjustment and right for horizontal adjustment) by an offset of half of manufacturing grid.

© 2020 All Rights Reserved.

15. Place as many vias as required.

Note: If a via has more than 100 000 cuts, only a box is displayed on the canvas. Else, all the cuts of the via are displayed on the canvas.

16. Press *Esc* or click *Cancel* in the *Create Via* form to finish creating vias.

Creating a cdsVia

You can create a cdsVia in your design after installing the `cdsVia` device in the technology file. For more information, see Installing a cdsVia Device in the Technology File.

To create a standard or custom via, see Creating a Single Via and Creating a Via with Arrayed Via Cuts.

To add a cdsVia to a design:

1. Choose *Options – Editor*.

   This opens the Layout Editor Options Form.

2. In the Create Via section, ensure that the required constraint group is selected in the *Via* list.

3. Choose *Create – Via*.

   This opens the Create Via Form.

4. Select *Single* as the *Mode* for creating the cdsVia.

5. From the *Via Definition* list, select a cdsVia.

   Note: The *Purpose* section is added to the *Create Via* form if you select a cdsVia from

© 2020 All Rights Reserved.

the *Via Definition* list.

Notice the *Cut* section where *Width* and *Length* fields are set to `0.3`, and the *Row Spacing* and *Column Spacing* fields are set to `0.2`. In the *Purpose* section, the *Cut*

© 2020
All Rights Reserved.

purpose is set to *drawing*. Therefore, these values are getting populated from the following rule in the `LPPtestCG` constraint group:

```
( minWidth            "Via1"          0.3 )
( minSpacing          "Via1"          0.2 )
```

6. In the *Purpose* section, select *lowVoltage* from the *Cut* list.

The *Width*, *Length*, *Row Spacing*, and *Column Spacing* values are automatically re-calculated, as shown below.

These values are picked from the following entries in the `LPPtestCG` constraint group:

```
( minWidth            ("Via1" "lowVoltage")       0.4 )
( minSpacing          ("Via1" "lowVoltage")       0.4 )
```

7. Click the *Show Enclosures* button

Notice that the enclosure values update automatically depending on the purpose you select from the *Cut* list in the *Purpose* section and based on the following entries in the `LPPtestCG` constraint group:

```
( minWidth            "Metal1"                    0.3 )
( minWidth            ("Metal1" "lowVoltage")     1.0 )
( minWidth            ("Metal1" "highVoltage")    2.0 )
```

Note: Enclosure parameters are supported on Pcell vias.

8. In the design window, click where you want to place the cdsVia.

© 2020 All Rights Reserved.

9. Place as many vias as required.
10. Press *Esc* or click *Cancel* in the *Create Via* form to finish creating cdsVias.

The parameters supported by the via engine for `cdsVia` are:

```
rowParamName_          = "row";
columnParamName_      = "column";
widthParamName_        = "cutLayerW";
lengthParamName_       = "cutLayerL";
spacingXParamName_   = "xCutSpacing";
spacingYParamName_   = "yCutSpacing";
originParamName_        = "origin";
layer1XDefParamName_ = "layer1XDefOverride";
layer1YDefParamName_ = "layer1YDefOverride";
layer2XDefParamName_ = "layer2XDefOverride";
layer2YDefParamName_ = "layer2YDefOverride";
layer1XEncParamName_ = "layer1XEnclosure";
layer1YEncParamName_ = "layer1YEnclosure";
layer2XEncParamName_ = "layer2XEnclosure";
layer2YEncParamName_ = "layer2YEnclosure";
layer1DirParamName_    = "layer1Direction";
layer2DirParamName_    = "layer2Direction";
```

The parameters supported by the via engine for `customVia` are:

```
rowParamName_           = "row";
columnParamName_      = "column";
widthParamName_         = "w";
lengthParamName_        = "l";
spacingXParamName_    = "xCutSpacing";
spacingYParamName_    = "yCutSpacing";
pitchXParamName_        = "xPitch";
pitchYParamName_        = "yPitch";
originXParamName_       = "xBias";
originYParamName_       = "yBias";
layer1XDefParamName_  = "layer1XDefOverride";
layer1YDefParamName_  = "layer1YDefOverride";
layer2XDefParamName_  = "layer2XDefOverride";
layer2YDefParamName_  = "layer2YDefOverride";
layer1XEncParamName_  = "layer1XEnclosure";
layer1YEncParamName_  = "layer1YEnclosure";
layer2XEncParamName_  = "layer2XEnclosure";
```


© 2020 All Rights Reserved.

```
layer2YEncParamName_  = "layer2YEnclosure";
layer1DirParamName_     = "layer1Direction";
layer2DirParamName_     = "layer2Direction"
```

Creating a Stack Via

You can create a via stack by physically stacking vias belonging to adjacent layers on top of each other. The footprint of each via in the via stack matches the footprint of the top via layer. This ensures that the metal shapes across all vias in the via stack coincide. Vias in a via stack have the same origin. If a via stack has connectivity, all vias in that via stack have the same net name.

By default, in the *Stack* mode, the via width, length, row and column spacing, and the enclosure values are derived based on the calculation mode set in the *Via Parameters Calculation Mode* list in the Layout Editor Options Form.

You can use via variants for creating a via stack. Such via stacks can be edited by using both the *Edit Via Properties* form and the *Property Editor* assistant.

In the following example, the figure on the left-hand side is an M5M9 via stack created using the *Create Via* form's *Stack* option. The figure on the right-hand side shows the constituent vias `M5M6`, `M6M7`, `M7M8`, `M8M9` of the via stack.

To create a via stack:

1. Choose *Create – Via* or click the *Create Via* icon on the *Create* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Create – Via.*

   The Create Via Form opens.

© 2020 All Rights Reserved.

2. Select the *Stack* option.

   The lower portion of the form updates to display the *Start Layer* and *End Layer* drop-down lists.

3. Click the *Options* button to set up the default viaDefs to be available during the via creation and to enable overriding of the via variant parameters. For more information, see Setting Up Via Creation Options.

4. Specify the *Net Name* of the net you want to associate with the via stack. If the specified net does not exist, a new net is created. If you leave this field blank, no net is associated with the via stack.

5. To create the via stack as a ROD object, select *Create as ROD Object*. You can specify a name in the *Name* field.

6. To work with a via variant, see Creating or Updating a Via Variant.

© 2020 All Rights Reserved.

7. In the *Start Layer* and *End Layer* drop-down lists, select the layers between which you want to place the via stack. After you specify both *Start Layer* and *End Layer* values, the form updates to display all the vias available between the two selected layers.

8. Specify the number of *Top Rows* and *Top Columns* you want in the via stack.

9. Select a viaDef for each layer pair from the *Via Definitions* drop-down list.

   For each viaDef you select, the *Via Type / Source* column displays the type of viaDef it is - standard or custom - and the technology library where the selected viaDef is defined.

Caution

If you update the technology file while the Create Via form is still active with the Mode set to Stack, your changes to the technology file will not reflect in the form. To ensure the form gets updated based on your changes in the technology file, close and reopen the form.

10. Click in the design where you want to place the via stack.

11. To rotate or mirror the via stack as you place it, click *Rotate*, *Sideways*, or *Upside Down* and then place the via stack. After placing the via stack, each time you click any one of these buttons or click right, the via stack is further rotated or mirrored.

© 2020 All Rights Reserved.

Note: Using the Property Editor assistant, you can edit the layers of the via that is part of the via stack from the via stack. You cannot edit the layers from the via object.

Other Methods of Creating a Via Stack

You can create a via stack using any one of the following:

- *Create – Wire* command while switching from one routing layer to a non-adjacent routing layer
- *Auto* option on the *Create Via* form on two intersecting and non-adjacent routing layers
- *Single* option on the *Create Via* form and stacking the vias manually

Creating or Updating a Via Variant

To create a via variant:

1. In the Create Via Form, select the *Mode* as *Single*.
2. In the *Via Definition* cyclic field, select a standard or custom viaDef.
3. On the *System* tab, change one or more via parameters, such as number of rows and columns, cut spacing, offset, or justification, as required.
4. Click the *Save Via Variant* button.

   This opens the Save Via Variant Form.
5. Choose where to save the via variant, either in the *Technology Library* or in the *Design*. If you want to save the via variant in the technology library, select one from the drop-down list.
6. In the *Save as* field, specify a name for the via variant.
7. Review the updated parameters in the *Summary* read-only list box.
8. Click *OK*.

   If you specify an existing viaDef name, a message in the CIW informs you that a viaDef with the specified name already exists. To specify a different name for the via variant, follow steps 4 to 8 and specify a different name for the via variant.

   The created via variant appears in the *Via Definition* cyclic field as a child node of the selected parent viaDef.
9. Click the + sign to expand the node and select the saved via variant.

© 2020 All Rights Reserved.

To update an existing via variant:

1. In the Create Via Form, select the *Mode* as *Single*.
2. Click the *Options* button.

   This opens the Create Via Options Form.
3. Ensure that the *Override Variant Parameters* check box is selected.

   Note: You cannot update the via parameters of an existing via variant if the *Override Variant Parameters* check box is not selected. When the *Override Variant Parameters* check box is not selected, the fields for specifying the *Cut Class*, cut *Rows* and *Columns*, and enclosure values are grayed out.
4. Click *OK*.
5. In the *Via Definition* cyclic field in the *Create Via* form, select a standard or custom via variant.
6. On the *System* tab, change one or more via parameters, such as number of rows and columns, cut spacing, offset, or justification, as required.
7. Click the *Save Via Variant* button.

   This opens the Save Via Variant Form.
8. Choose where to save the via variant, either in the *Technology Library* or in the *Design*. If you want to save the via variant in the technology library, select one from the drop-down list.
9. In the *Save as* field, specify a name for the via variant or select from the cyclic field.
10. Review the updated parameters in the *Summary* read-only list box.
11. Click *OK*.

    - If you specify the same name as that of an existing via variant name, a message box appears to confirm whether you want to overwrite the existing via variant. You can click *Yes* or *No* on the message box to specify your choice. The via variant is not updated if you click *No*.
    - If you create a via variant that has the same viaDef and parameters as another existing via variant with a different name, a message box appears to confirm whether you want to rename the existing via variant. You can click *Yes* or *No* on the message box to specify your choice. The via variant is not renamed if you click *No*.

    The updated via variant appears in the *Via Definition* cyclic field as a child node of the selected parent viaDef.

© 2020 All Rights Reserved.

12. Click the + sign to expand the node and select the saved via variant.

Creating a Via Automatically in Single and Stack Modes

Use the *Compute From Shape(s)* check box to generate vias automatically in the *Single* or *Stack* modes when you click the overlap region of intersecting shapes.

1. Choose *Create – Via* or click the *Create Via* icon on the *Create* toolbar.

Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Create – Via.*

The Create Via Form opens.

2. Click the *Options* button to set up the via creation options. For more information, see Setting Up Via Creation Options.

3. Select the *Mode* for creating vias.

a. If you select *Single*, do the following:

i) Select the *Compute From Shape(s)* check box.

Notice that the fields for specifying the *Cut Class*, cut *Rows* and *Columns*, and enclosure values are grayed out.

ii) Select a viaDef from the *Via Definition* cyclic field.

Note: Ensure that the selected viaDef matches at least one of the two intersecting shapes for generating the automatic via.

iii) In the *Cut* section, specify the *Width*, *Length*, *Row Spacing*, and *Column Spacing* values.

b. If you select *Stack*, do the following:

i) Select the *Compute From Shape(s)* check box.

Notice that the fields for specifying *Top Via Rows* and *Top Via Columns* are grayed out.

ii) Select the start and end layers for creating the via stack.

iii) Select a viaDef for each layer pair from the *Via Definition* drop-down list.

4. Click in the overlapping area of the intersecting shapes to create a via/stacked via.

The figures below show an example of a *Single* via created at the intersection of two shapes when the *Compute From Shape(s)* check box is selected. The vias in the first

© 2020 All Rights Reserved.

figure are generated when the *Extend Enclosure Beyond Overlap* check box is not selected in the Create Via Options Form. The second figure illustrates the enclosure extension when the *Extend Enclosure Beyond Overlap* check box is selected.

Extend Enclosure Beyond Overlap is off.

© 2020
All Rights Reserved.

Extend Enclosure Beyond Overlap is on.

5. You can continue to place vias at the required locations.
6. Press `Esc` or click *Cancel* on the form to finish creating vias.

Note: The auto via mode supports shapes like circles, ellipses, and donuts.

© 2020 All Rights Reserved.

A statistics report is generated for auto via in the form of a table in the CIW. This report is best visualized when the default CIW font is mono. A sample report is displayed below.

```
(VIA-1010): Running the auto via command at point (1.5, 3.5).
*****************************************************************************
*                           Auto Via Statistics                             *
* Point : (1.5, 3.5)                                                        *
* Run Time = 0.01 sec                                                       *
* Total number of vias created: 6                                           *
*****************************************************************************
#
# Summary
==================================================================================================================
| Stack |  Metal Overlaps   |      Via      |    Initial Overlaps     |        Final Results        |  Percent   |
| Depth |  Bottom - Top     |               |   Total   | Unconnected | Newly Connected | Connected |            |
------------------------------------------------------------------------------------------------------------------
|   2   |  Metal1 - Metal3  |  Via1 - Via2  |         1 |           1 |               1 |         1 |   0 -> 100 |
------------------------------------------------------------------------------------------------------------------
|                Totals                     |         1 |           1 |               1 |         1 |   0 -> 100 |
==================================================================================================================
t
```

Note: Auto via considers stopped layers as defined by the connectivity extractor in the `virtuosoDefaultExtractorSetup` constraint group. For more information, see Specifying Layer-Purpose Pairs in the validLayers Constraint.

Creating a Via Automatically on Intersecting Shapes

You can automatically place a single via or a stacked via in the overlap area of intersecting polygons, rectangles, paths or path segments. The via is created if there are available valid vias in the current constraint group in the technology database that connect the two layers on which the shapes exist. If there is a taper type constraint in the constraint group in the technology database, the vias are added to the valid vias list. When you place vias with *Auto* option selected, the via origin snaps to the center of the overlap, irrespective of the *Snap Spacing* settings in the *Display Options* form.

- When a via is created in *Auto* mode or when the *Compute From Shape(s)* option is selected in the Create Via form, by default the *Minimum Rules* calculation mode is applied. However, you can change the calculation mode by setting the `createViaAlwaysUsesDefinedCalcMode` environment variable to `true`.
- The auto via mode supports polygonal overlaps. If a non-metal via is created in a polygonal overlap, such a via is non-editable. In case the overlap area is a path or path segment, the via is created in every path or path segment subpart.
- When shapes abut, the auto via mode creates a via only when the shapes are path or path segment and when they are connected by their begin or end center line point.

Note: In case there are multiple shapes on different layers overlapping at the same place, the auto via mode considers the adjacent metals. For example, if there is a `M1`, `M2`, and `M4` shape at the same location, the auto via mode considers two different overlaps, `M1`-`M2` and

© 2020 All Rights Reserved.

`M2-M4`. It does not consider `M1-M4` as an overlap, unless `M2` is not visible in the Palette assistant.

By default, in the *Auto* mode, the via width, length, row and column spacing, and the enclosure values are derived based on the calculation mode set in the *Via Parameters Calculation Mode* list in the Layout Editor Options Form.

By using the *Auto* option, you can do the following:

- Automatically Placing a Via with a Single Click
- Automatically Placing a Via Based on Connectivity Information of the Intersecting Shapes

Auto vias cannot be created at the click point within an intersection area if one of the following is true:

- No valid via or constraint group is defined in the technology file for connecting the intersecting layers.

  You can either edit the technology file to include the definition for vias connecting the intersecting layers or change the layers of the intersection objects to match a via defined in your technology file.

An appropriate warning message will be issued in the CIW in each of the above cases.

© 2020 All Rights Reserved.

Automatically Placing a Via with a Single Click

1. In the Create Via Form, select the *Auto* option.

2. Click the *Options* button to set up the default viaDefs to be available during the via creation and to enable overriding of the via variant parameters. For more information, see Setting Up Via Creation Options.

3. To associate the via to a particular net, type the name of the net in the *Net Name* field.

© 2020 All Rights Reserved.

4. Click in the overlapping area of the intersecting shapes.

Intersecting Paths

Via Created at the Path Intersection By Using the Auto Mode

© 2020 All Rights Reserved.

The figures below illustrate a few cases of auto via generation.

Non-orthogonal pathSegs; Non-rectangular metal intersection area

Non-rectangular intersection area; Overlapping shapes for one metal layer

Butting edges of pathSegs at metal intersection, rectangular metal intersection area

Non-rectangular metal intersection area; overlapping pathSegs for one metal layer

© 2020 All Rights Reserved.

Intersection of complex polygon shapes

Intersection of 45 degree pathSegs

Note: When the auto via mode creates polygonal vias, the metal shape is reduced to a single-cut-sized shape under the most lower left cut. The via cuts must be covered by the existing metal shapes in the design. For stacked vias, extra metal shapes for the intermediate layers are created, which are then associated with the vias as child shapes. Editing polygonal via parameters by using property editor or properly assistant or modifying the via geometries with Chop or Stretch commands is not supported.

© 2020 All Rights Reserved.

For polygons or pathsegs, the coverage and cuts between vias are always same, irrespective of where you click in the shape.

Tip

If there are multiple overlapping shapes, you can center-click and drag to select an area that encloses the required overlapping shapes. Vias will be generated at all the intersecting shapes.

Note: In the auto via creation mode, a via is created in the intersection between the overlapping shapes and the area selected by center-click and dragging the pointer.

5. Continue to place vias at the required locations, if needed.

© 2020 All Rights Reserved.

6. Press `Esc` or click *Cancel* in the form to complete creating vias.

Automatically Placing a Via Based on Connectivity Information of the Intersecting Shapes

1. In the Create Via Form, select the *Auto* option.
2. Click the *Options* button to set up the default viaDefs to be available during the via creation and to enable overriding of the via variant parameters. For more information, see Setting Up Via Creation Options.
3. Select the *Check Shape Connectivity* check box.

   The *Net Name* field is removed from the form.

4. (Optional) To place the vias on only those intersecting shapes that belong to a specific net, enter the particular net name in the *Only for Specified Net(s)* field.
5. Click in the overlapping area of the intersecting shapes. Alternatively, drag the middle mouse button to select the intersecting shapes.

   A via is placed in the intersecting area based on the connectivity information of the intersecting shapes.

5. You can continue to place vias at the required locations.

© 2020 All Rights Reserved.

6. Press `Esc` or click *Cancel* in the form to complete creating vias.

Auto and Fast Edit Mode Options

In the auto and fast edit via creation modes, you can use the *Prevent DRC Violations With Neighbors* option to prevent the creation of a via if it introduces DRC violations with the neighboring shapes. If this option is not selected, the via is created even if it introduces DRC violations with the neighboring shapes.

The example below depicts auto via creation using the *Prevent DRC Violations With Neighbors* option.

Auto via on Metal3- Metal5 overlap creates a minSpacing violation with neighboring shape in Metal4

© 2020 All Rights Reserved.

When the *Prevent DRC Violations With Neighbors* option is not selected, the auto via is created even though it introduces a DRC violation. This is depicted in the figure below.

Auto via is created

When the *Prevent DRC Violations With Neighbors* option is selected, the auto via is not created as it introduces a DRC violation. A shaded via and *(DRC ERROR)* label are displayed on the canvas.

© 2020 All Rights Reserved.

In the auto and fast edit via creation modes, you can also use the *Show Preview* option to preview of the via that will be created as shown in the figure below.

When you move the pointer over the overlap, the preview of the via is displayed

You can use the *Show Hints* option to display the Auto Via Preview Bindkeys hint box on the canvas.

© 2020 All Rights Reserved.

Locking the via captures the number of rows/columns along with the `cutClass`.

© 2020 All Rights Reserved.

The figure below shows the lock/unlock via option when the via preview is not displayed.

In the example below, create via command is invoked in auto mode. If you press the `Spacebar` key when the pointer is over an existing via, the via parameters of the existing via are locked, as shown in the figure below.

© 2020
All Rights Reserved.

You can then draw an area using the middle mouse button to drop the vias on the other overlaps, as shown in the figure below.

The vias that are dropped have the same parameters as the via that has been locked.

Setting the Technology to Create Via

This section describes the aspects related to the technology setup needed for creating via.

- Specifying the Calculation Modes for Via Parameters
- Supported Parameters for Custom Via
- Specifying viaEngineContext User Type
- Supported Constraints
  - Supported Constraints for Local Interconnect Vias
  - Supported Constraints for Auto Mode and Fast Mode Vias
- Installing a cdsVia Device in the Technology File

© 2020 All Rights Reserved.

Specifying the Calculation Modes for Via Parameters

You can set the calculation mode for via parameters to one of the following using the *Via Parameters Calculation Mode* option in the Layout Editor Options form:

- *Min Rules & ViaDef*: For all the via parameter fields, such as the number of cut rows and columns, and cut sizes, the minimum rule or the viaDef default values, whichever is greater, are used. If minimum rules are not defined, viaDef defaults are used. The spacing and enclosure values are calculated automatically from the minimum rules. The minimum rules are picked from the constraint group that you select from the *Via* list in the *Create Via* section. The calculated values for the spacing and enclosure fields are never less than the viaDef defaults.

  For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.1` is used as the enclosure value in the *Create Via* form. Similarly, if the `minSpacing` rule is `0.2` and the default cut spacing in viaDef is `0.18`, `0.2` is used as the cut spacing X and Y in the *Create Via* form.

- *Minimum Rules*: For all the via parameter fields, only the minimum rules, such as `minWidth` and `minSpacing`, are used.

  For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.05` is used as the enclosure value in the *Create Via* form. The viaDef default enclosure value is ignored.

- *ViaDef Defaults*: For all the via parameter fields, viaDef defaults are used.

Supported Parameters for Custom Via

The parameters supported by the via engine for customVia are:

`row`	`column`	`w`
`l`	`xCutSpacing`	`yCutSpacing`
`xPitch`	`yPitch`	`xBias`
`yBias`	`layer1XDefOverride`	`layer1YDefOverride`
`layer2XDefOverride`	`layer2YDefOverride`	`layer1XEnclosure`
`layer1YEnclosure`	`layer2XEnclosure`	`layer2YEnclosure`

© 2020 All Rights Reserved.

layer1Direction	layer2Direction

Note: The `cutPattern` parameter is supported for custom Pcell vias.

Specifying viaEngineContext User Type

When you call pre or post viaEngine callback function, the viaEngineContext userType is specified. The following example, depicts the usage of userType information.

```
procedure(myPreViaEngineCB(veCxt)
printf("call pre viaEngine procedure with context '%s' and viaDef '%s'.\n" veCxt->operationType veCxt->viaModel->viaDef->name)
printf("overlapInfo is: ")
println(veCxt->overlapInfo)
t
)
viaRegisterPreViaEngineCallback("myPreViaEngineCB")
```

When creating a via in single via mode, the output of the above-listed callback function is:

```
\o call pre viaEngine procedure with context 'check' and viaDef 'M3_M2'.
\o overlapInfo is: nil
```

When creating a via in auto via mode, the output of the above-listed callback function is:

```
\o call pre viaEngine procedure with context 'generate' and viaDef 'M1_PO'.
\o overlapInfo is: ((((0.25 -4.0) (0.25 1.0) (-0.25 1.0) (-0.25 -4.0)) "Metal1" "drawing" "bottom to top" 0.5) (((1.0 -1.0) (1.0 2.0) (-1.0 2.0) (-1.0 -1.0)) "Poly" "drawing" "vertical" 2.0) ((-0.25 -1.0) (0.25 1.0)))
\r t
```

In the above example, the value, `0.5`, means that the first involved shape on `"Metal1" "drawing"` has a width of `0.5`. The value, `2.0`, means that the second involved shape on `"Poly" "drawing"` has a width of `2.0`.

And the intersection of the two shapes gives the following overlap box:

```
((-0.25 -1.0) (0.25 1.0))
```

Supported Constraints

Create Via (*Single*), *Edit Via Properties* (single via), and *Property Editor* assistant for vias use technology library constraints and constraint groups to adjust via creation and editing to adhere to the following constraints:

© 2020 All Rights Reserved.

- minCutClassSpacing (One layer)

```
spacingTables(
     ( minCutClassSpacing "Via1"
        ( ( "cutClass" (Via1_Small (Via1_Bar 'shortEdge) (Via1_Bar 'longEdge)
Via1_Large) nil
        "cutClass" nil nil ) 'paraOverlap  -1  0.100 )
        (
            ((Via1_Small 'shortEdge) (Via1_Small 'shortEdge)) (0.134
'centerToCenter)
            ((Via1_Small 'shortEdge) (Via1_Small 'longEdge)) (0.134
'centerToCenter)
            ((Via1_Small 'shortEdge) (Via1_Bar 'shortEdge)) 0.100
            ((Via1_Small 'shortEdge) (Via1_Bar 'longEdge)) 0.100
            ((Via1_Small 'shortEdge) (Via1_Large 'shortEdge)) 0.080
            ((Via1_Small 'shortEdge) (Via1_Large 'longEdge)) 0.080
        )
     );minCutClassSpacing
);spacingTables
```

- viaSpacing (distance between the cuts of a via)

  The value of the distance parameter in the viaSpacing constraint is used as a less than (<) rule.

```
spacings(
         (viaSpacing   "via" 'distance  1.1  'numCuts  2  'centerToCenter
'exceptSamePGNet  'sameVia 'exactAligned  3 'cutClass  "viaCutClass.2.8"
'exactSpacing 'sideParaOverlap 2.2))
        )
```

- forbiddenCutClasspacingRange

```
spacings(
    ( forbiddenCutClassSpacingRange Via1 'cutClass (0.3 0.4)
    'shortEdge("[0.07 0.09]")
)
```

- minEndOfLineEdgeExtension

```
orderedSpacings(
     ( minEndOfLineEdgeExtension "V1" "M2"
     'cutClass  "viaCutClass.2.8"
     'jointWidth 0.14
     'spanLength 0.2 0.04
     )
); orderedSpacings
```

- minCenterlineExtension

```
orderedSpacings(
     ( minCenterlineExtension "VIA1" "METAL1" 'cutClass (0.3 0.4)
     'maxEffectiveWidth 0.2
     'endOfLineWidthRange (0.1 0.2)
     'distanceWithin 0.7
     'exactWidth 0.2 )
); orderedSpacing
```


© 2020 All Rights Reserved.

- minNumCut

```
spacingTables(
    (minNumCut "via2" (("width" nil nil)
    'cutClass  "viaCutClass.2.8" 'above)
    (
        1.0  2
        2.0  6
        4.0  8
            )
        )
    ...
    )
```

- minEdgeAdjacentLength, which applies when a via has an edge of less than a specified length. The edges adjacent to this edge must be at least the length specified by the constraint.

```
spacings(
     minEdgeAdjacentLength metal1 'maxLength 0.6   (1.0 1.2)
)
```

- minExtensionDistance

```
orderedSpacings(
(minExtensionDistance "metal1" "metal2" 1.0
 …
)
```

- minOppExtension (scalar and table-based)

Scalar:

```
orderedSpacings(
  ( minOppExtension  "Metal1"  "Via1"  'cutClass  Via1_Small  (0.0 0.03)  )
  ( minOppExtension  "Metal1"  "Via1"  'cutClass  Via1_Bar  'endSide  (0.04
    0.0)  )
  ( minOppExtension  "Metal1"  "Via1"  'cutClass  Via1_Large  'endSide  (0.05
    0.0)  )
  ( minOppExtension  "Metal2"  "Via1"  'cutClass  Via1_Small  (0.0 0.03)  )
  ( minOppExtension  "Metal2"  "Via1"  'cutClass  Via1_Bar  'endSide  (0.04
    0.0)  )
  ( minOppExtension  "Metal2"  "Via1"  'cutClass  Via1_Large  'endSide  (0.05
    0.0)  )
) ;orderedSpacings
```

Table-based:

```
spacingTables(
     ( minOppExtension "Metal1" "Via1" 'cutClass Via1_Small 'endSide
           (( "width" nil nil )
            (

              0.0  (0.01  0.015)
              0.20 (0.02  0.025)
              0.3  (0.03  0.035)
              0.4  (0.04  0.045)
              2.0  (0.05  0.065)
```


© 2020 All Rights Reserved.

```
            )
     );minOppExtension
);spacingTables
```

- minWidth

```
spacings(
    (minWidth "metal2" 0.1)
)
```

- minLargeViaArraySpacing

```
spacingTables(
    ( minLargeViaArraySpacing tx_cutLayer(( "numCuts" nil nil ) ['cutClass
      g_width|(g_width g_length)|t_name]
      ['intraArrayCutSpacing g_cutSpacing]
      ['paraOverlap] [g_defaultValue] )
      ( g_table )
    ) ;spacingTables
```

- minLargeViaArrayCutSpacing

```
spacings(
    ( minLargeViaArrayCutSpacing tx_layer [`minNumCuts numCuts] g_spacing)
      …
    );spacings
```

- minLargeViaArrayWidth

```
spacings(
    ( minLargeViaArrayWidth tx_layer g_width )
      …
    )
```

- largeRectViaArrayAllowed

```
spacings(
    ( largeRectViaArrayAllowed tx_cutLayer { t | nil } )
    …
    )
```

- allowedCutClass

```
spacingTables(
( allowedCutClass tx_cutLayer
    (( "width" nil nil "width" nil nil )
     'cutClass {f_width | (f_width f_length) | t_name}
     ['lowerLayer tx_lowerLayer]
     ['upperLayer tx_upperLayer]
     x_default
```


© 2020 All Rights Reserved.

```
        )
        (g_table)
    )
    ) ;spacingTables
```

- minSpacing (One layer)

```
    spacings(
    ( minSpacing tx_layer
        ['inLayerDir tx_layer2]
        [{'horizontal | 'vertical} ['eolWidth f_eolWidth]]
        ['sameNet ['PGNet] | 'sameMetal]
        ['sameMask]
        ['ignoreShapesInRanges ((("width"))(g_ignoreTable))]
        f_spacing ['manhattan]
    )
    ( minSpacing tx_layer
        ['area f_maxArea]
        f_spacing ['manhattan]
    )
    ) ;spacings
    spacingTables(
    ( minSpacing tx_layer
        (( "width" nil nil ["width" | "length" nil nil] )
         ['inLayerDir tx_layer2]
         [{'horizontal | 'vertical} ['eolWidth f_eolWidth]]
         ['sameNet ['PGNet] | 'sameMetal]
         ['sameMask]
         ['exceptEolWidth f_eolWidth]
         ['ignoreShapesInRanges ((("width"))(g_ignoreTable))]
         [f_default]
        )
        (g_table) ['manhattan]
    )
    ( minSpacing tx_layer
        (( "twoWidths" nil nil "length" nil nil )
         ['inLayerDir tx_layer2]
         [{'horizontal | 'vertical} ['eolWidth f_eolWidth]]
         ['sameNet ['PGNet] | 'sameMetal]
         ['sameMask]
         ['fillConcaveCorner f_fill]
         ['ignoreShapesInRanges ((("width"))(g_ignoreTable))]
```


© 2020 All Rights Reserved.

```
     [f_default]
    )
    (g_table) ['manhattan]
)
) ;spacingTables
```

- minNeighborViaSpacing

```
spacings(
( minNeighborViaSpacing tx_cutLayer
    'within f_within
    ['centerToCenter]
    f_spacing
)
) ;spacings
```

- minExtensionToCorner

```
orderedSpacings(
( minExtensionToCorner tx_layer1 tx_layer2
    {'toConcaveCorner
        {['parallelExtension f_parExt 'width f_width 'length f_length]
         | ['convexLength f_convexLength 'enclosure (f_encl1 f_encl2)]
        }
      | 'toConvexCorner ['eolWidth f_eolWidth] ['minLength f_minLength]
    }
    ['cutClass {f_width | (f_width f_length) | t_name}]
    ['viaEdgeType x_viaEdgeType]
    ['extendCornerBy f_extendCorner]
    ['exceptCornerTouch]
    f_spacing
)
) ;orderedSpacings
```

- allowedWidthRanges

```
spacings(
( allowedWidthRanges tx_layer
    ['length f_length]
    ['orthogonal]
    ['horizontal | 'vertical | 'measureHorizontal | 'measureVertical]
    ['stepSize f_stepSize ['stepRange g_range]]
    (g_ranges)
)
) ;spacings
```


© 2020 All Rights Reserved.

- minAreaEdgeLength

```
( minAreaEdgeLength tx_layer
    [
        ['exceptMinWidth f_width]
            | ['exceptEdgeLength f_edgeLength
                | (f_minEdgeLength f_maxEdgeLength)
              ]
          ['exceptMinSize (f_minWidth f_minLength)
            | ((f_minWidth f_minLength) …)
        ]
        ['exceptStep (f_length1 f_length2)]
    ]
    f_area
)
) ;spacings
```

- minEdgeAdjacentLength

```
spacings(
( minEdgeAdjacentLength tx_layer
    'maxLength f_maxLength
    ['edgeCount x_maxEdges]
    ['convexCorner ['exceptWithin f_within]
        | 'concaveCorner
        | 'threeConcaveCorners ['width f_width]
    ]
    f_length
)
( minEdgeAdjacentLength tx_layer
    'maxLength f_maxLength
    'edgeCount x_maxEdges
    (f_length1 f_length2)
)
) ;spacings
```

- maxWidth

```
spacings(
( maxWidth tx_layer f_width )
) ;spacings
```

- viaKeepoutZone

```
spacings(
( viaKeepoutZone tx_cutLayer
```


© 2020 All Rights Reserved.

```
    'cutClass {f_width | (f_width f_length) | t_name}
    ['otherCutClass {f_width1 | (f_width1 f_length1) | t_name1}]
    'dualExtension (f_ext1 f_ext2)
    ['dualSideExtension (f_sideExt1 f_sideExt2)['sideVertical]]
    ['sameMask]
    ['exactAligned f_spacing
        ['allEdge | 'shortEdge | 'longEdge]
    ]
    ['cornerOnly]
    ['exceptConnectivityType]
    f_extension
)
) ;spacings
```

- `rectShapeDir` (Area-based Constraint)

```
spacings(
( rectShapeDir tx_layer
    ['extendBy f_extendBy ['widthRanges (g_ranges)]]
    ['insideLayers (tx_layer1 tx_layer2 … tx_layerN)
     ['insidePurposes (t_purpose1 t_purpose2 … t_purposeN)]
        |
     'outsideLayers (tx_layer1 tx_layer2 … tx_layerN)
     ['outsidePurposes (t_purpose1 t_purpose2 … t_purposeN)]
    ]
    ['exceptViaLayer tx_viaLayer
        'exceptViaSize (f_viaSize1 f_viaSize2)
        'exceptExactSize (f_exactMetalSize1 f_exactMetalSize2)
    ]
    ['exceptEdgeLength f_maxLength 'width f_maxWidth ['twoSides]]
    ['exceptRanges (g_exceptRanges)]
    ['exceptNonCorePins]
    {"any" | "horizontal" | "vertical"}
)
) ;spacings
spacingTables(
( rectShapeDir tx_layer
    (( "width" nil nil )
     ['extendBy f_extendBy ['widthRanges (g_ranges)]]
     ['exceptViaLayer tx_viaLayer
        'exceptViaSize (f_viaSize1 f_viaSize2)
        'exceptExactSize (f_exactMetalSize1 f_exactMetalSize2)
```


© 2020 All Rights Reserved.

```
        ]
        ['exceptEdgeLength f_maxLength 'width f_maxWidth ['twoSides]]
        ['exceptRanges (g_exceptRanges)]
        ['exceptNonCorePins]
        [ "any" | "horizontal" | "vertical" ]
       )
       (g_table)
)
) ;spacingTables
```

The via editing and creation commands also comply with the following parameters:

- The 'coincidentAllowed parameter of the minExtensionDistance and minOppExtension constraints. If both minExtensionDistance and minOppExtension are used, minOppExtension takes precedence.

  Example:

  ```
  orderedSpacings(
      ( minExtensionDistance   "Metal1"   "Via1"   0.06   'coincidentAllowed)
      ( minExtensionDistance   "Metal2"   "Via1"   0.06   'coincidentAllowed)
  ) ;orderedSpacings
  ```

  With the 'coincidentAllowed parameter in the minExtensionDistance rule, an enclosure of 0.0 is allowed (the allowed enclosure can be either 0.0 or > = 0.06).

- The 'centerToCenter parameter of the viaSpacing and minViaSpacing (One layer) constraints and the constraint minCenterToCenterSpacing. If 'centerToCenter is specified, both distance and spacing are measured from cut center to cut center instead of from cut edge to cut edge.

  Example:

  ```
  Spacings(
    (viaSpacing  "Via1"  'centerToCenter  'numCuts 3  'distance  1.5 1.0)
  ); spacings
  ```

  Sets the minimum via cut spacing on Via1 to 1.0 when there are three or more cuts that are 1.5 or fewer user units apart.

- The 'cutClass parameter of the viaSpacing and minOppExtension constraints.

  Example:

  Cut class can be specified by cut sizes.

  ```
  ( viaSpacing  "Via2" 'cutClass 0.14 'numCuts 3 'distance 0.21 0.3  )
  ( viaSpacing  "Via2" 'cutClass (0.14 0.28) 'numCuts 3 'distance 0.21 0.4  )
  ( viaSpacing  "Via2" 'cutClass 0.28 'numCuts 3 'distance 0.21 0.5  )
  ```


© 2020 All Rights Reserved.

In the first rule, width is `0.14`. Since length is not been specified, it is the same as width. Similarly, in the third rule, both width and length are `0.28`. In the second rule, width is `0.14` and length is `0.28`.

Cut class can also be specified by cut class name.

```
( viaSpacing  "Via2" 'cutClass "Small" 'numCuts 3 'distance 0.21 0.3  )
( viaSpacing  "Via2" 'cutClass "Bar" 'numCuts 3 'distance 0.21 0.4  )
( viaSpacing  "Via2" 'cutClass "Large" 'numCuts 3 'distance 0.21 0.5  )
```

- The `'noPrl` parameter of the `viaSpacing` constraint.
- The `'extraCut`, `'area`, `'distanceWithin`, and `'nearbyLayer` parameters of the `minOppExtension` constraint.
- The `'distanceWithin` and `'nearbyLayer` parameters of the `minOppExtension` constraint.
- The `'overLayer` and `'exceptViaConnectivityType` parameters of the `minViaSpacing (One layer)` constraint.
- The `'maxNumCuts` and `'applyPerDirection` parameters of the `minLargeViaArrayCutSpacing` constraint.
- The `'above`, `'below`, and `'bothBelowAndAbove` parameters of the `minNumCut` constraint.

The *Create Via – Stack* and *Auto* commands adhere to the following additional constraints:

- `minArea` (for the intermediate layers in a via stack)
- `minOutsideCornerEdgeLength`
- `minQuadrupleExtension`
- `minSize`
- `minStepEdgeLength`
- `viaStackingLimits`
- `orthogonalWSPGrid (ICADV12.3 Only` (Auto via only)

If you change parameters on the *Create Via* or *Edit Via Properties* forms that cause a violation of these constraints, the values are adjusted to comply with the constraints defined in the technology library. The rule checking excludes Origin X and Y.

Before you create or edit a via, set the constraint group in the *Wire Editing* section of the Layout Editor Options Form. The selected constraint group determines the viaDefs that are available for creating and editing vias.

© 2020 All Rights Reserved.

The via engine uses both shapes and layer blockages to define regions for area-based constraints. For more information, see *Virtuoso Technology Data Constraints Reference*.

Supported Constraints for Local Interconnect Vias

Local interconnect vias are vias between local interconnect (LI) layers.

LI layers do not need a physical via or contact shape in order to be connected. However, the automatic router model requires a via for every transition from one layer to another in order for a route to be considered valid. This is why LI vias are required to establish connections between LI layers, even though there is no contact layer between these layers.

At advanced process nodes, many foundries add support for LI layers to the process stack. LI layers form connections through overlaps rather than using separate cut layers. Typically, only rectangles are allowed on LI layers and these shapes often have a fixed direction. LI vias adhere to the following constraints:

- `allowedLengthRange`
- `minExtensionOnLongSide`
- `minSideExtension`
- `minWireOverlap`
- `minDirectionalOverlap (Advanced Noded Only)`
- `rectShapeDir`

Supported Constraints for Auto Mode and Fast Mode Vias

The auto and fast edit modes of the Create Via command support area-based constraints. For more information, see *Virtuoso Technology Data Constraints Reference*.

Note: These constraints are not supported for single and stack vias.

In Virtuoso XL and higher tiers, the `'edgeExtension`, `'layer`, and `'extensionDirection` parameters are supported for the `viaSpacing` constraint when the *Prevent DRC Violations With Neighbors* option is selected.

Auto via also supports the `minVoltageExtension` constraint.

© 2020 All Rights Reserved.

Support for Voltage-Dependent Rules

Voltage-dependent rules are critical for design processes at 45 nm and less. The support for these rules is based on the ability to use user-defined layer purposes that represent voltage ranges and on the ability to define design rule constraint groups within these voltage ranges.

To support these rules in Virtuoso, you can define non-drawing purposes, purpose-aware constraints, such as those for width and spacing, in the technology file. Additionally, you can define `cdsVia`, which is a via device that can accept non-drawing purposes.

Note: Standard vias accept only the `drawing` purpose.

You can define a `cdsVia` either directly in the technology file or by using the *Install Device* form. For more information about the *Install Device* form, see the *Virtuoso Technology Data User Guide*.

To be able to create and edit cdsVias on non-drawing, voltage-aware purposes, the following tasks need to be performed in the order specified:

- Installing a cdsVia Device in the Technology File
- Adding the cdsVia Device to the validVias Constraint
- Creating a cdsVia

Creating Multipart Paths

See Chapter 8, “Creating and Editing Multipart Paths”.

Creating Fluid Guard Rings

See Chapter 9, “Creating and Editing Fluid Guard Rings”.

Creating MPP Guard Rings

See Creating an MPP Guard Ring in Creating and Editing Multipart Paths.

Creating Slots

Metal slots are the holes that you create in the metal layer to relieve the stress that results from heat generation, especially in large metal shapes. Creating slots in large shapes

© 2020 All Rights Reserved.

improves the structural integrity of shapes, making them less likely to fracture because of constant expansion (heating) and contraction (cooling).

The following figure shows a rectangle with slots and illustrates the various slot parameters:

You use the *Create Slot* command to create slots in exis>lp;/ting rectangle and path shapes. You can also create slots in rectangle and path shapes while creating these shapes. For more information, see Creating Rectangles and Creating Paths.

Note: The *Create Slot* command is available only for rectangles and paths.

To create slots in an existing rectangle or path that does not have slots:

1. Choose *Create – Slot*.
2. Press `F3`.

© 2020 All Rights Reserved.

The Create Slot Form appears.

3. In the *Layer* list, select an LPP.

   The LPP information is required if the resulting shape with slots is a ROD object.

4. In the *Shape Type* list, select *Preserve*, *Convert to Polygon*, or *Convet to ROD* based on the type of shape you want to create after slots are added to it.

5. Specify *Width/Length* or *Width/Area*.

   If you deselect the *Specify width, area* check box, the text fields available are *Width* and *Length*. If you select the *Specify width, area* check box, the *Length* and *Area* fields are available.

6. In the *Width* and *Length* or *Width* and *Area* fields, specify the slot dimensions.

7. In the *Spacing (Column), Spacing (Row), Spacing (Edge)*, and *Stagger* fields, specify values as required.

8. In the design display area, click an existing rectangle or path that does not have slots. You can also select multiple shapes to create slots in them. The selected shapes can be a mix of ROD objects and polygons.

   If the rectangle or path you select already contains slots, a message is displayed in the CIW, informing you to select only the rectangles and paths that do not have slots. If you want to edit existing slots in a rectangle or path, see Editing Slots.

   Slots are created in the selected shape(s). You can continue to select more rectangles or paths to create slots in them.

© 2020 All Rights Reserved.

9. Press *Esc* or click *Cancel* in the form to finish creating slots.

Note: You can use the `lxShapeSlotting` function to create slots on metal shapes.

The *Create Slot* command supports both ion and post-selection modes. For more information about these modes, see Pre-Selection and Post-Selection of Objects.

Slotting Flow for Wide Wires

You can use the density-aware `lxShapeSlotting` SKILL API to automatically create slots on metal shapes during wide wire routing. This saves you having to manually draw complex wires and other shapes that are time-consuming to create and which do not always take full account of required electrical and design rule constraints.

The recommended slotting flow is as follows:

1. Pre-route the design

   In this step, the wide wires match the expected resistance (R) value as well as the electromigration criteria. Some margin is considered in terms of wire width to anticipate further degradation that slotting might introduce. For more information, see *Virtuoso Interactive and Assisted Routing User Guide*.

2. Use the Virtuoso Electrically-Aware Design (EAD) flow as early as possible in order to verify whether the electrical constraints are satisfied. For more information, see *Virtuoso Electrically Aware Design Flow Guide*.

3. Run the density aware slotting SKILL command (`lxShapeSlotting`) so that slots are added to the shapes to fix `maxDensity` and `maxWidth` rules (wherever required).

4. Use EAD to verify that the electrical constraints are satisfied.

   If the electrical constraints are still not satisfied, you might need to increase the width of some of the shapes that are concerned by violations or remove some of the slot shapes to ease out the current flow. Then re-run the slotting SKILL command.

5. Run the DRD checker to verify the `maxDensity` and `maxWidth` rules. For more information, see *Virtuoso Design Rule Driven Editing User Guide*.

© 2020 All Rights Reserved.

Creating P&R Objects

Creating Blockages

A blockage object specifies an area in a design where certain objects cannot be placed or are to some degree restricted in their placement. The type of objects to exclude from the blockage area depends on the blockage type.

Blockages are defined by points. You can use the command to seed the *Create Blockage* form. The command will capture the layer information of the blockage, if applicable. It will also capture the blockage type and update the environment variable `blockageType`. In addition, if the blockage is of type `fill` or `screen`, the command will capture the value of maximum density per cent and update the environment variable `blockageMaxDensity`.

The different types of blockages include:

- A placement blockage prohibits the placement instances in the blockage area.
- A routing blockage prohibits creating or placing interconnect shapes (path segments, paths, multipart paths, or single-level shapes such as rectangles and polygons) or vias in the blockage area on the defined layer.
- A pin blockage prohibits creating or placing shape, instance, or via pins in the blockage area on the defined material and layer.
- A feedthru blockage prohibits creating feedthrus (routing through multiple gates through the terminal on the same layer) in the blockage area.
- A screen blockage allows you to create routing in the blockage area until the maximum density specified for the blockage area is reached on the defined material and layer.

Note: When Design Rule Driven Editing is set to on, blockages participate in DRD rule checking so that error markers/messages are created when you violate certain design rules.

Effective Width

The effective width for a blockage is an attribute that you can assign to any blockage types defined by a layer or layers such as routing, pin, feedthru.

- The effective width (if set), rather than the size of the block object is used for the entire blockage when a `minSpacing` value in the table spacing rules is defined in the technology file. These values determine which spacing rules apply. Design-Rule-Driven Editing (DRD) will flag the error in Notify mode or prevent placement in Enforce mode if the object is closer than the defined width for the layer.

© 2020 All Rights Reserved.

- If a single minSpacing value is defined in the technology file, It is applied and the effective width is not used.
- It is not necessary to have the effective width defined for all the layer blockages.
- When blockage type is placement the Effective Width field is grayed out, since it applies to instances and not layers.

Attaching a Blockage to an Owner

You can optionally use the *Attach to Owner* option to assign an object as the blockage's owner, depending upon the type of blockage. In the case of a placement halo blockage, the owner should be an instance or P&R boundary. An owner can have any number of blockages. Cyclical relationships are not allowed. For example, if object A is a parent to object B, then object B cannot be a parent for object A.

Blockage Summary

The *File - Summary* command will report information with regards to blockage objects. It will contain the total number of blockage objects, the blockage count of each type of blockage in the cellview, and is sorted by layer.

Creating a Blockage Using the Manual Create Method

To create a blockage,

1. Choose *Create – P&R Objects – Blockage...*.

   The Create Blockage Form opens.

2. Select the Blockage Type from the cyclic field.
3. If the blockage is a layer blockage, select the appropriate material type from the cyclic field.
4. If the blockage is a layer blockage, select the appropriate layer from the layers list in the Layer field.
5. Select Manual in the Create Method field.
6. Select the appropriate snap mode.

   You are prompted to "Point at the first point of the blockage:"

7. Click the first point of the blockage.

© 2020 All Rights Reserved.

You are prompted to “Point at the next point of the blockage:”

8. Click the next point of the blockage.

   Just as you would for a polygon, continue to enter points until the shape is the final shape you want the blockage to be.

9. Press the Return key or double click to complete the blockage object.

Creating a Blockage Using the Automatic Create Method

To create a blockage,

1. Choose *Create – P&R Objects – Blockage....*

   The Create Blockage Form opens.

2. Select the Blockage Type from the cyclic field.
3. If the blockage is a layer blockage, select the appropriate material type from the cyclic field.
4. Select the appropriate layer from the layers list in the Layer field.
5. Select Automatic in the Create Method field.
6. Select the appropriate Derive From method from the cyclic field.

   If you choose Selected Set bounding box you are prompted to “Select object(s) to derive blockage.

7. Click an instance or object which you want to include in the blockage area.

   You are prompted to “Use Apply button of the form to create blockage OR select more object(s)”

8. Continue to click the objects which you want to include in the blockage area.
9. When you have finished selecting the set of objects you want included in the blockage area, click the Apply button or press the Return key to complete the blockage object.

Creating Blockage Halos

A blockage halo defines an area around an instance that cannot overlap the placement blockage.

Placement Halo: A placement halo is a type of blockage associated with an instance or a master, representing an area around the master’s prBoundary. The shape of the halo is

© 2020 All Rights Reserved.

determined from the shape of the prBoundary of the master and the given offsets, and it is automatically updated to reflect changes in the master's prBoundary.

The placement halo is most commonly used to prevent standard cells from being placed within a certain range of an instance. This is usually done to make sure there is enough space to properly route to the (presumably) large number of pins on the side of a block.

Layer Halo: A layer halo is a type of blockage associated with an instance or a master, representing an area on a given layer around the master's prBoundary. The shape of the halo is determined from the shape of the prBoundary of the master and the given offsets, and it is automatically updated to reflect changes in the master's prBoundary.

The `oaLayerHalo` applies around the perimeter of an instance. It prohibits routing on the given layer parallel to the edge of the instance, but it allows routing perpendicular to the instance edge. This prevents long lines being routed parallel to the edge of the instance, which causes cross-coupling. However, it allows routing straight into the pins. Since power lines do not switch, they can be routed anywhere within the halo area.

Halo Editor Form is a sub-form invoked from the *Create – Instance* and *Create – P&R Boundary* commands. To bring up this form, click *Define Halo*.

Creating Rows

A row represents a location for the placement of standard cells, macros, or devices and is a repetition of the site element in a given direction (horizontal or vertical). A row is basically a one dimensional array of a given site. All sites in one row must be the same size. Rows are drawn using the display attributes for LPP *row boundary*, a Cadence reserved layer which should already be in the Palette Assistant. Rows are drawn as a small rectangle with a diagonal line in the corner of rows to show the row orientation. Sites are drawn as a large rectangle with a diagonal line in the corner of sites to show the site orientation.

Tip

The display of row name and row site can be turned on or off in the *Display Options* form accessed from the *Options – Display* menu. Use the *Row Name* and *Row Site* check boxes. You can control the snapping of instances to rows. Adding appropriate site pattern to the instance masters results in successful snapping of instances to rows.

© 2020 All Rights Reserved.

Site Definitions

Site definitions (`siteDefs`) are technology library objects which are the basic elements from which placement rows are constructed. Each row is associated with `siteDefs`. A site defines a type, width, height, and allowable symmetry of each placement location row. A new site is defined for each cell height. Cell widths must be a multiple of the site width. The height of a row equals the height of the site and the width of a row equals the multiple of a site width. Two types of site definitions (`siteDefs`) exist, scalar site definitions (`scalarSiteDefs`) and array site definitions (`arraySiteDefs`).

- Scalar site definitions define sites where you can place a standard row.
- Array site definitions define a pattern of s`calarSiteDefs`. `arraySiteDefs` define a pre-determined set of sites that you use to place a set of cells in part of a standard row. `arraySiteDefs` are primarily used in gate-array designs. Each scalar site definition in the array, along with its offset from the origin (the lower left corner) of the array and its orientation are defined. Each pattern is a named reference to a scalar site definition.

Site definitions must first be defined in the `siteDefs` section of the technology file as a `scalarSiteDef`. An example is shown below:

```
siteDefs(
scalarSiteDefs(
;( siteDefName          type width  height  symInX symInY symInR90)
;( -----------          ---- -----  ------  ------ ------ -------)
 ( gpdk090site          core 0.29    2.61     nil    t      nil)
) ;scalarSiteDefs
```

You can add multiple site names and definitions in this section and select the one you want from the *Site Definition Name* field in the Definition Section of the *Create Row* form.

Templates

After defining Site Definitions, you may define the rails and/or components as a SKILL program entered into the input line of the CIW. This information appears in the *Site Information* field in the Definition Section of the *Create Row* form. An example is shown below:

```
procedure( createDefs()
let(((cv geGetEditCellView()))
    railDef1 = dbCreateRailDef(cv "rail1" "vdd!"
            list("metal1" "drawing") "bottomAlignment" "bottomAlignment" 2.0
0.0 0.0 0.0)
    railDef2 = dbCreateRailDef(cv "rail2" "gnd!"
            list("metal1" "drawing") "topAlignment" "topAlignment" 2.0 0.0 0.0
0.0)
    compDef1 = dbCreateCompTypeSetDef(cv "compDef1" "bottomAlignment"
            "bottomAlignment" list("R0" "MX") list("MyComp1" "MyComp2") 0 0)
    compDef2 = dbCreateCompTypeSetDef(cv "compDef2" "topAlignment"
            "topAlignment" list("R0" "MX") list("MyComp1" "MyComp2") 0 0)
```


© 2020 All Rights Reserved.

```
        t
    )
    )
```

The rails and components are displayed in the *Pre-Defined* field in the ***Templates* Section** of the *Create Row* form. Use the arrow buttons to move the rail/component definitions from the *Pre-Defined* pane to the *Selected* pane or vice-versa. Only the *Selected* definitions will be used for creating the row or cluster boundary.

To create rows,

1. Choose *Create – P&R Objects – Row.*

   The Create Row Form opens.

   Opening the *Create Row* form will turn on Row visibility in the Objects Panel, even if you have previously turned it off. `siteDefs` must be present in the technology file for the *Objects* assistant to turn on visibility. If the Site Definitions are not present in the technology file, a warning message displays in the CIW. In this case, the *Create Row* form does not appear and the command ends.

   Definition Section

2. Type the row name in the *Base Name* field.

   When you enter the base name, an underscore is added followed by a number, which is greater than or equal to 1. As you enter new sites, each site is incremented by 1. For example, if your base name is ROW_1, the name assigned to the first site is ROW_1_1, the next is ROW_1_2, the next is ROW_1_3, and so on. If you do not enter a custom row name, the software assigns the name row_number, which increments by 1 each time you add a site and rows.

3. Select the orientation of the row, either *Horizontal* or *Vertical*.

4. Select the *Starting Site Orientation*.

   All subsequent rows will follow the orientation you set in this field. The cyclic field valid values are *R0*, *R90*, *R180*, *R270*, *MX*, *MY*, *MXR90*, *MYR90*.

5. You can configure various *Options* for rows, such as you may set every other row to flip or set only the first row to flip. You may specify the number of rows to abut or set to abut the first and second rows. You can also specify the space between the rows to be created.

6. Select the site definitions, core or pad or both, based on which you want to filter the row created. The cyclic field valid values are *ALL*, *CORE*, and *PAD*.

7. Select the site from the *Site Definition Name* list box whose elements are to be placed in the row. Selecting each site definition name one-by-one displays the dimension

© 2020 All Rights Reserved.

information about the site in the *Site Information* box. For more information about creating sitedefs, see Site Definitions.

Templates Section

8. Templates allow you to place rails and/or components automatically into the rows. Choose pre-existing rails or components or both to create the row. If pre-defined rails or components exist, they will display in the *Pre-Defined* pane. Select the ones you want to use for row creation one-by-one and move them to the *Selected* pane by using the left and right arrows. For more information about creating rails and components, see Templates.

   Note: Template definitions are associated only with the cellview in which they are defined. Therefore, the rail and component templates available to one cellview will not be available to another cellview.

Create Method Section

9. Next, specify the way in which the row is to be created. Select one of the following methods:

   a. In the *Assisted* method, specify the *Number of Rows* to be created and the *Number of Sites* to be created for each row. When you move the pointer to the design window, the wireframe displays the number of rows specified in the form. Click the cellview at the position where you want to place the row(s). Each row will have the number of sites you specified in the form. When you open the form again, the last values entered in the fields are retained.

   b. In the *Area Fill* method, select the type of area, *Rectangular*, *Rectilinear*, or *Partition*, to place the row(s). Drag and select the area in the cellview where you want to place the row(s). The software calculates and creates the maximum number of rows in the selected area and creates the maximum number of sites for each row.

   - *Rectangular*: Draw a rectangle. The software creates rows in the selected area until there is no more room in the rectangle for an additional row. Maximum number of sites are created for each row. If you set offsets, they are obeyed within the rectangular shape.

   - *Rectilinear*: It works in the same way as for a rectangle, but the rows are of different lengths and widths depending on the shape of the polygons. If you set offsets, they are obeyed within the rectilinear shape. The rectilinear polygon should not be self intersecting.

   - *Partition*: Create an area consisting of pre-defined areas by choosing either *prBoundary* or *snapBoundary*.

© 2020 All Rights Reserved.

In the *Area Fill* method, you can specify *Offset* values to place asymmetrical rows in the boundary areas. The offset must be a floating number — one or greater than one. Set the distance from *Top*, *Bottom*, *Left*, and *Right* of the boundary in which the rows should not enter.

In the following example, the left image is Row Orientation Horizontal rows, Starting Site Orientation is R0, Create Method is Assisted, Number of Rows is 4, Number of Sites is 5.

The right image is Row Orientation Horizontal rows, Starting Site Orientation is R0, Create Method is Assisted, Number of Rows is 4, Number of Sites is 5, with Flip Every Other Row, Flip The First Row, and Abut set at 4 rows. This is useful to make rows more compact and if

© 2020 All Rights Reserved.

you have rails in the rows, place source at the top of the rows and ground at the bottom. This will avoid shorts in the rows.

For more information on site definitions, see the *Virtuoso Technology Data ASCII Files Reference*.

© 2020 All Rights Reserved.

Creating Custom Placement Areas

To create a placement area,

1. Choose *Create – P&R Objects – Custom Placement Area.*

   The Create Custom Placement Area Form opens.

2. Select the method for creating a custom placement area - *Manual*, *Assisted*, or *Auto*. When creating the custom placement area manually, you need not specify other attributes such as height and width of the area. So if you have selected *Manual*, you can directly move to Step 8.

   Definition Section

3. Type the custom placement area name in the *Base Name* field.

4. In the *Assisted* or *Auto* mode, specify the *Height* and *Width* of the placement area.

5. If using the *Auto* mode, also specify the *Number of Placement Areas* you want to create.

6. Specify the width between the placement areas in the *Channel Width* field.

7. Select the orientation of the custom placement area, either *Horizontal* or *Vertical*.

8. Configure various settings for the placement area(s), such as set every other area to flip or set only the first area to flip, and specify the number of areas to abut or set to abut the first area.

   ***Templates* Section**

9. Templates allow you to place rails and/or components automatically into the custom placement areas. Choose pre-existing rails or components or both to create the custom placement area. If pre-defined rails or components exist, they will display in the *Pre-Defined* pane. Select the ones you want to use for custom placement area creation one-by-one and move them to the *Selected* pane by using the left and right arrows. For more information about creating rails and components, see Templates.

   Note: Template definitions are associated only with the cellview in which they are defined. Therefore, the rail and component templates available to one cellview will not be available to another cellview.

© 2020 All Rights Reserved.

Creating Clusters

Clusters provide a grouping mechanism for placement control interpreted and used by floor planning and placement tools. They are identified by a unique name. Clusters are not graphical objects

Clusters are used to associate instances and cluster boundaries. They can own area and layer blockages. Each instance can belong to only one cluster, but a cluster can have any number of cluster boundaries. Cluster boundaries are used to indicate the collective area into which you can place the contents of a specific cluster of instances.

Clusters can be hierarchical, so that you can associate clusters with other clusters. Clusters can also own blockages. You define each cluster as one of three types:

- **inclusive** - Instances other than instances in this cluster may be placed in its associated cluster boundaries.
- **exclusive** - Only instances in this cluster may be placed in its associated cluster boundaries.
- **suggested** - Instances in this cluster may be placed outside of its associated cluster boundaries.

To create a cluster,

1. Choose *Create – P&R Objects – Clusters*.
2. Select and filter the objects you want to include in the cluster. Once you select them, they are displayed in the Filter Instances and Clusters area, in the lower right quadrant of the form.

© 2020 All Rights Reserved.

3. Choose the displayed instances, clusters and sub clusters to add new or existing clusters in the Clusters Editing Form Area.

4. Use the Add to Cluster button to move instances or clusters you have selected to the Current Cluster Editing Area of the form.

   The selected instances and clusters are added to the trunk, or if you select the branch in the Current Cluster Editing Area the instances and clusters are added to the correct branch.

5. Use the scroll bar to display the view names.

© 2020 All Rights Reserved.

The Clusters Editing Form Area displays all of the clusters defined in the cellview in tree table format.

6. Create new and edit existing clusters and sub clusters using the upper right quadrant section of the form.

   Because you cannot select clusters, you also use this quadrant of the form to edit the properties of clusters, and to delete them.

   The newly created cluster is also added to the read-only cluster combo field, and the current cluster display area will display the newly created cluster.

Creating Track Patterns

Tracks patterns are uniformly-spaced orthogonal patterns extending across the base array. The tracks that make up the track pattern provide guidelines for laying interconnect routes. Gridded routers place the center-line of each route collinear with an unused segment of a routing track.

© 2020 All Rights Reserved.

This section covers the following topics:

- Technology File Requirements for Track Patterns
- Minimum Routing Pitch
- Via Stacking
- Preferred and Non-Preferred Routing Directions
- Diagonally Adjacent Vias
- Determining the Track Spacing
- Determining the Number of Tracks
- Track Patterns Assigned to Routing Layer “None”
- Displaying Track Patterns
- Creating Track Patterns

Technology File Requirements for Track Patterns

Following is the technology file requirements for creating track patterns.

```
layerDefinitions(
techPurposes(
(track 229 trk)
...
)
techDisplays(
(metal1 track packetName t t t t t)
...
)
techLayerPurposePriorites(
(metal1 track)
...
)
layerRules(
functions(
(metal1 "metal" 3)
...
)
```

To efficiently utilize the routing area, tracks should be created to consider the following design aspects:

- Minimum Routing Pitch
- Via Stacking
- Preferred and Non-Preferred Routing Directions

© 2020 All Rights Reserved.

- Diagonally Adjacent Vias

Minimum Routing Pitch

Track spacing is determined from the routing pitch. The most common routing pitch is line-to-via, because it allows the minimum pitch between routes in which a via can be placed next to a route. A larger pitch does not efficiently utilize the routing area, where as smaller pitch may result in low utilization of the routing area because the router will not be able to place vias next to a route. See Determining the Track Spacing for more information.

Via Stacking

© 2020
All Rights Reserved.

Routing layers using the same preferred direction can be placed on coincident tracks to allow for via stacking. If tracks can not be coincident, place the tracks at a ratio, for example, 1:2 or 2:3.

Preferred and Non-Preferred Routing Directions

Tracks are created in both the preferred and non-preferred routing direction to allow for X Y locations where routes can be jogged and vias and pins can be placed and connected. Points where a horizontal track on one layer crosses over a vertical track on another layer are via locations where a vias can be placed. Most routing is done on the tracks aligned in the preferred direction. Small-distance jogs use tracks in the non-preferred direction.

Diagonally Adjacent Vias

© 2020 All Rights Reserved.

Increasing the routing pitch may be necessary to allow for diagonally placed adjacent vias. This may increase the orthogonal spacing, but can reduce the overall die size by readily allowing connections to via rather than longer routes around congested areas.

Determining the Track Spacing

Determine the line-to-via track spacing (routing pitch) for each routing layer by adding:

half the via width + metal to via spacing + half the width of the metal

See, Minimum Routing Pitch for information about different types of routing pitches.

© 2020 All Rights Reserved.

Determining the Number of Tracks

You must determine the number of tracks needed for the width and the height of the cellview. Calculate the number of horizontal tracks by dividing the height of the design area by the routing pitch of the metal. Determined the number of tracks needed in vertical direction using the same method. Create horizontal and vertical track patterns for each routing layer. When *Snap to Track* is on in the *Layout Editor Options* form, you are only allowed to route in areas where there are intersecting horizontal and vertical tracks.

Track Patterns Assigned to Routing Layer “None”

Track patterns can be created without an assigned routing layer. This can be done to save a routing track for use with other applications. Some tools, such as a global router will section the chip by looking at the number of routing tracks. Creating additional tracks without a routing layer can trick the tool into seeing more sections.

This can also be used to defined points for placing pins without allowing routing in one direction. For example, on the vertical edge of a block, you may want to place pins for a horizontal route, but you may not want to route on the vertical edge, because the route would be half outside the block. To resolve this, you create a vertical track directly on the vertical edge. The intersection points of the vertical track and the horizontal tracks define valid points

© 2020 All Rights Reserved.

to place pins. The router can route to the pin inside the block using the horizontal tracks, but can not route to the pin (inside the block) vertically.

Displaying Track Patterns

To display all track patterns, do the following:

1. If necessary, update the technology file. See Technology File Requirements for Track Patterns.
2. Track layers (layers with the purpose *track)* must be valid and visible.
3. From the Grids Panel, turn on *Track – Vis*.
4. To display individual track patterns, turn visibility off and on for specific track layers through the Layers Panel.
5. Use *Redraw* to update the view each time display changes are made to track patterns.

© 2020 All Rights Reserved.

Note: The track pattern spacing and how closely the design is viewed using *Zoom*, will determine if track patterns can be seen.

Creating Track Patterns

Note: Create tracks in both preferred and non-preferred layer directions.

To create a track pattern, follow these steps:

1. From the layout window, choose *Create – P&R Objects – Track Patterns*.

   The Track Pattern Editor Form opens.

2. From the *Filter by Routing Layer* cyclic field, select the routing layers of the track patterns you want displayed in the form.

   This allows you to limit the form to only show the track patterns on the routing layer you want to edit. This does not effect the routing layers displayed in the design window.

3. From the *Layer* field, select the routing layer for the track pattern.

4. From the *Direction* field, select *Horizontal* or *Vertical* tracks.

   For more information, see Preferred and Non-Preferred Routing Directions.

5. In the *Spacing* field, type in the track spacing (routing pitch).

   For more information, see Determining the Track Spacing.

   In the *Origin Offset* field you can optionally type a value to offset the wire from the track.

6. In the *Number of Tracks* field, type in the number of required tracks.

   For more information, see Determining the Number of Tracks.

7. Click *Update*.

   To view the track pattern you created, you must redraw the design window (*View – Redraw*).

Creating P&R Boundary

A P&R Boundary is the boundary of a block for place-and-route applications. In a top-down design flow, it is commonly passed down to constrain the area in which the block's contents are placed. Any shape or instance created for the block should be enclosed within the P&R boundary. In a bottom-up design flow, a P&R boundary is used to indicate the primary extent of the block's contents. For cell placement that is directed by sites in a row object, the P&R

© 2020 All Rights Reserved.

boundary determines the aspect of the cell that must align with the site. Only one P&R boundary is allowed per cellview.

To create a P&R boundary

1. Choose *Create – P&R Objects – P&R Boundary*. Press F3 to open the form.

   If the visibility setting of *P&R Boundary* in the *Objects* panel is turned off, a dialog box appears with the option to set the P&R boundary to visible and proceed with the boundary creation. When you click *Yes*, *P&R Boundary* is set to visible in the *Objects* panel.

   The Create P&R Boundary Form opens.

2. From the *Create Method* choose,

   - *From Selected Set*

     Add or delete objects from the selected set. Once you have selected the objects, use the Apply button to create the P&R Boundary.

   - *Rectangle*

     Specify a rectangle area for the boundary. The *Apply* button and *Snap Mode* cyclic field are grayed out when using this method.

   - *Polygon*

     Specify a polygon area for the boundary and select a snap mode. The *Apply* button is grayed out when using this method. Valid snap modes are anyAngle, diagonal, orthogonal, L90XFirst, L90YFirst.

3. To create a halo blockage for the P&R boundary, click Define Halo. See Creating Blockage Halos.

Creating Snap Boundary

A Snap Boundary is used for designs that implement any rectilinear macros or standard cells. A snap boundary must be a rectangle. Placement tools use the snap boundary to place instances of the design in rows, thereby aligning the snap boundary origin with the site origin. If the P&R boundary is a rectangle, no snap boundary is needed. You may need a snap boundary for cells with rectilinear (but non-rectangular) P&R boundaries. Only one Snap boundary is allowed per cellview.

To create a snap boundary

1. Choose *Create – P&R Objects – Snap Boundary.* Press F3 to open the form.

© 2020 All Rights Reserved.

The Create Snap Boundary Form opens.

2. From the *Create Method* choose,

   - *From Selected Set*

     Add or delete objects from the selected set. Once you have selected the objects, use the *Apply* button to create the P&R Boundary.

   - *Rectangle*

     Specify a rectangle area for the boundary. The *Apply* button field is grayed out when using this method.

Creating Area Boundary

Area Boundary is a generic boundary in a physical block. It is used to indicate a region of a block for various purposes defined by a design methodology, thus each area boundary has a unique name. You can tie an area boundary to a constraint group that can direct advanced placement and routing solutions.

To create an area boundary,

1. Choose *Create – P&R Objects – Area Boundary.* Press F3 to open the form.

   The Create Area Boundary Form opens.

2. Assign a name to the area boundary in the *Name* field. The name must be unique to the cellview.

   The software automatically assigns names beginning with `AB` (for area boundary), followed by an underscore and a number, to make the name unique to the cellview.

3. From the *Create Method* choose,

   - *From Selected Set*

     Add or delete objects from the selected set. Once you have selected the objects, use the Apply button to create the P&R Boundary.

   - *Rectangle*

     Specify a rectangle area for the boundary. The *Apply* button and *Snap Mode* cyclic field are grayed out when using this method.

   - *Polygon*

© 2020 All Rights Reserved.

Specify a polygon area for the boundary and select a snap mode. The *Apply* button is grayed out when using this method. Valid snap modes are anyAngle, diagonal, orthogonal, L90XFirst, L90YFirst.

4. If you want to create the area boundary as a ROD object, select *Create as ROD Object*. The *Name* field becomes editable, allowing you to change the name from the default.
5. To create the soft fence, select the *Soft Fence* option.

Creating Cluster Boundary

Cluster Boundary defines an area in which the contents of a specific cluster are included or excluded, depending upon their definition. The cluster boundary is attached to a specific cluster object. Each cluster boundary has a unique name with respect to all other cluster boundaries in a given cellview.

To create a cluster boundary

1. Choose *Create – P&R Objects – Cluster Boundary.* Press F3 to open the form.

   The Create Cluster Boundary Form opens.

2. Select an existing cluster for the boundary from the *Cluster Name* field.

   Clusters can be created and named using the *Create – P&R Objects – Clusters* command.

3. Optionally you can overwrite the automatically assigned name by typing a name for the boundary in the *Cluster Boundary Name* field. The name must be unique to the cellview.

   The software automatically assigns names beginning with CB (for cluster boundary), followed by an underscore, then a number unique to the cellview.

4. Turn on *Cross-Highlight Boundaries* when generating new cluster boundaries, use this option to help you associate existing clusters with their boundaries.
5. From *Create Method* choose either, *Manual* or *Automatic*.

   - When *Manual* is selected choose either *Rectangle* or *Polygon*,

     - *Rectangle*

       Specify a rectangle area for the boundary. The Snap Mode cyclic field is grayed out when you use this method.

     - *Polygon*

© 2020 All Rights Reserved.

Specify a polygon area for the boundary and select a snap mode. Valid snap modes are *anyAngle*, *diagonal*, *orthogonal*, *L90XFirst*, *L90YFirst*.

- When *Automatic* is selected, the boundary is created using the aspect ratio, origin, and target area. Aspect Ratio W/H is the width-to-length ratio of the cluster boundary. For example, a value of 1 specifies a square boundary, 2 specifies a boundary twice as long as it is wide, and .5 specifies a boundary half as wide as it is long. The default is 1.

If the target area is greater than 0 and you create a rectangular cluster boundary whose area is less than the target area, a warning is issued in the CIW stating that the utilization value is greater than 100%.

Creating Groups

The group command allows you group shapes and/or instances together in order for you to manipulate the objects together, rather than individually. It will have as an owner an area boundary object, which will conform to the bounding box of the entire group (the composite of all of the bounding boxes of the members of the group). This area boundary is automatically created, edited and deleted by the Group command as objects are added to or deleted from the group.

A group:

- is a named database figure, which you can query in the database.
- can contain any number of figures or instances, including other groups.
- is selectable.
- has an origin and an orientation.
- does not have a layer/purpose, though its members may.
- does not have connectivity, though its members may.
- may have an applied constraint group.

To create a group:

1. Choose *Create – Group*.

   The Create Group Form opens.

2. (Optional) Name the group by typing the name in the *Name* field.

© 2020 All Rights Reserved.

The group is automatically named using the convention shown above; the first group is named `Group0` and as groups are added the number increments by one.

3. (Optional) Choose the orientation from the *Orientation* cyclic field.

   You are prompted to "`Select the figure(s) to add to group:`".

4. Click the objects you want to add to the group.

5. Click Apply.

   The group is created. When you place your pointer over a group a dotted line surrounds the objects in the group. To add an object to a group, remove an object from a group, or ungroup a group of objects, use the *Edit – Group* command. To edit the properties of objects in the group, use the *Edit – Properties* command.

For information about how to add or remove objects to or from a group and for ungrouping objects, see Editing Groups.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

7

Editing Objects

This chapter contains these topics about the Virtuos® Layout Suite L layout editor:

- Undoing and Redoing Commands
- Moving Objects
- Copying Objects
- Stretching Objects
- Deleting Objects
- Copying Objects in Repeat Mode
- Flipping Objects
- Rotating Objects
- Chopping Objects
- Merging Objects on a Layer
- Yanking and Pasting Objects
- Editing Instance Properties
- Reshaping Objects
- Splitting Objects
- Modifying Corners
- Sizing Objects
- Attaching and Detaching Objects
- Converting Instances to Mosaics
- Converting Circles, Donuts, and Ellipses to Polygons
- Converting Polygons to Paths

© 2020 All Rights Reserved.

- Changing Origin
- Aligning Objects
- Editing Slots
- Hierarchical Editing
- Editing Groups
- Selecting Objects
- Deselecting Objects

Undoing and Redoing Commands

For information about the *Undo* and *Redo* commands, see Undoing Commands and Redoing Commands.

Moving Objects

The *Move* command lets you move an object to another location in this or another cellview.

Support for Objects	
Multipart paths, Multipart rectangles	■ You can move multipart paths and multipart rectangles to a new cellview. All data remains intact. If the multipart path is chopped, that data also moves with it. ■ Multipart paths and multipart rectangles are renamed if there are objects with similar names in the new cellview. ■ To retain alignments while moving to another cellview, each of the aligned objects must be part of the selected set to be moved.

© 2020 All Rights Reserved.

ROD objects	■ You can move a ROD object within the same cellview or to another cellview. ■ Within the same cellview, moving a ROD object that has other objects aligned to it causes the aligned objects to move as well. ■ ROD objects are renamed if there are objects with similar names in the new cellview. ■ When you move a ROD object between cellviews, and the ROD object is aligned to another ROD object(s), the system preserves alignment only when the aligned ROD object(s) is also in the selected set; otherwise the alignment is broken. ■ Avoid rotating aligned ROD objects during a move because the aligned handle names are not updated after the move, so the results might not be what you want.
Ruler	You cannot move a ruler from one cellview to another.
Blockages	■ When blockages are in the selected set, the “Change To Layer” functionality remains enabled. The new layer information will apply to all shapes in the selected set. ■ If an owner is moved, the blockage will follow the owner.
Boundaries	■ All instances associated with the cluster, part of the cluster, and inside the bounding box are moved as a group. ■ You can use the Rotate, Sideways and Upside Down buttons on the move form for boundary objects.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Move` command is not supported for curved shapes.

Moving Objects by Direct Manipulation

You can move objects by selecting and dragging them.

To move an object,

1. Select an object. Alternatively, you can select an object by specifying the points at which the object is placed in the design area using the *Enter Points* form. For more information, see Enter Points Form.

© 2020 All Rights Reserved.

2. Move the pointer over the object to see the move pointer.

The move pointer appears over the object.

3. Press the left mouse button at the reference point for the move (the point from which the move starts) and drag the pointer to the destination point. Alternatively, you can specify the reference and destination points for the move using the *Enter Points* form. For more information, see Enter Points Form.

4. Release the left mouse button.

The object is moved.

Moving Objects By Using the Move Command

To move objects using the *Move* command,

1. Choose *Edit – Move* [m] or click the *Move* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Move.*

2. Press F3 to open the form.

   The Move Form opens.

3. Select one or more objects. If you create a selection box or press `Shift` while selecting all objects, the *Move* command prompts you for a reference point. If you click an object, the point (Pt, in the figure below) is taken as the reference point.

4. Click the reference point for the move (the point from which the move starts). Alternatively, you can specify the reference point for the move using the *Enter Points* form. For more information, see Enter Points Form.

© 2020 All Rights Reserved.

5. Click where you want to move the objects. Alternatively, you can specify the point where you want to move the objects using the *Enter Points* form.

Press left to create a selection box.

Click to enter a reference point.

Moving or Copying Objects to Another Layer

To move or copy objects to another layer,

1. Choose *Edit – Move* [m] or *Edit – Copy* [c]. Press F3 to open the form.

   The Move Form or Copy Form opens.

2. Set *Change Layer* on.
3. Click the layer field and choose the layer you want.
4. Click the object whose layer you want to change.
5. Click where you want to move or copy the object.

   The object is moved or copied and is now on the new layer.

Moving Objects to Another Cellview

To move objects to another open cellview,

1. Select the objects you want to move.
2. Choose *Edit – Move* [m].

   The Move Form opens.

© 2020 All Rights Reserved.

3. Place the pointer in another open cellview.
4. Click to place the objects.

Moving Multipart Rectangles

When you move a multipart rectangle within a cellview, the master rectangle and all of its associated arrays of subrectangles move together. You cannot move a master rectangle separately from its associated subrectangles, nor can you move subrectangles separately from their master rectangle.

When you move a multipart rectangle that has objects aligned to it, the objects move to stay in alignment with the multipart rectangle.

1. Choose *Edit – Move* [m].
2. In full selection mode, choose the multipart rectangle by clicking on the master rectangle or any subpart.

click any part of the multipart rectangle.

3. Click to place the multipart rectangle; the aligned objects follow.

The multipart rectangle and all objects aligned to it appear in the new location.

Copying Objects

The *Copy* command enables you to place a copy of an object in a cellview. You can copy an object to the same cellview or to another cellview.

Important

When you copy an object to another cellview, the constraints associated with an object are not copied to the resultant cellview.

© 2020 All Rights Reserved.

Support for Objects	
ROD objects	■ You can copy a ROD object within the same cellview or to another cellview. The system automatically assigns unique names to the copies. ■ Alignment between the ROD objects in the selected set determines the alignment between the corresponding copied objects. ■ Alignment with objects that are not in the selected set is ignored. ■ You can copy a whole multipart path (MPP) or a whole multipart rectangle (MPR), but not an individual subpart of an MPP.
Curved Shapes	**(ICADVM20.1 Virtuoso RF Solution Only)** You can copy curved shapes. **Note:** The design should be associated with the technology file that has non-IC fabric.
Ruler	You cannot copy a ruler from one cellview to another.
Blockages	■ If only the blockage object is selected, the resultant copied blockage does not have any owner attached to it. ■ If both the blockage object and its owner are selected, the resultant copied owner and blockage objects retain the ownership information. ■ If only the owner is selected, all blockages owned by that owner are copied to the resultant owner object and the blockage ownership information is retained. ■ The behavior described above also applies when you copy blockages from one cellview to another.
Boundaries	■ There can only be one P&R and one snap boundary in a cellview. Therefore, the *Copy* command works only when you copy a boundary from one cellview to another. If a P&R or snap boundary already exists in the destination cellview, a copy is not created. ■ If you copy an area boundary object with an associated constraint group between cellviews , the constraint group is copied with the area boundary.

To copy objects:

© 2020 All Rights Reserved.

1. Choose *Edit – Copy* [c] or click the *Copy* icon on the *Edit* toolbar.

   Alternatively, right-click in an empty space in the design display area and choose *Copy* from the *Layout* context-sensitive menu.

2. Press `F3` to open the form.

   The Copy Form opens.

3. Select the *Keep Copying* check box if you want to create multiple copies of an object.

4. In the *Snap Mode* list, select an option based on the direction in which you want to move the copied object.

© 2020 All Rights Reserved.

5. Select the *Change To Layer* check box if you want to copy an object to a different layer, and in the associated list, select the required LPP.

6. Specify the number of rows and columns if you want to create an array of copied objects. For more information, see Creating an Array of Copies.

7. In the *Chain Mode* list, select the appropriate option if you want to copy a chain of instances.

8. In the design display area, select one or more objects by using one of the following methods:

   - To select a single object, click it. Alternatively, you can select an object by specifying the points at which the object is placed in the design area using the *Enter Points* form. For more information, see Enter Points Form. To select multiple objects, hold down the `Shift` key and click the required objects, one at a time.

     A copy of the selected object appears in the design display area, attached to the pointer. This copy stays attached to the pointer and moves when you move the pointer. The *Copy* command prompts you to click at the location where you want to place the copy.

   - Drag the pointer over the objects you want to select. You can add more objects to the selected set by holding down the `Shift` key and then clicking the required objects or by dragging the pointer over the required objects.

     The *Copy* command prompts you to click to specify a reference point.

9. Click in the design display area to place the copy. Alternatively, you can specify the points at which you want to place the copy using the *Enter Points* form. For more information, see Enter Points Form.

   Note: If you copy a parent object, copies for all its child objects are created, whether or not you select the child objects individually. In the pre-selection mode, the copies of the parent and all the child objects remain selected after you place them in the design display area.

Important

The reference point is set to the lower left corner of the selection set on hitting the *Apply* button on the Copy form.

10. Press `Esc` or click *Cancel* in the form to finish copying.

The following example illustrates the creation of multiple copies of `pinA` on `Metal1`, where the copies are placed on different layers, `Metal3` and `Metal4`. In the *Copy* form, select the *Keep Copying* check box and set *Snap Mode* to *anyAngle*. Select the *Change To Layer*

© 2020 All Rights Reserved.

check box, and in the associated list, select `Metal3`. Select `pinA`. A copy of the pin attaches to the pointer. Notice that `pinA` remains selected.

Click in the design display area to place the first copy. Notice the change in the LPP of the copy you place. Copy remains attached to the pointer.

Click in the design display area to place the second copy, also on `Metal3`.

Next, press `F3` to display the *Copy* form and select `Metal4` from the *Change To Layer* list. As a result, the next copy will be placed on `Metal4`. Click in the design display area to place the copy. Press `Esc` to finish creating the copies.

© 2020 All Rights Reserved.

Video

To view how to use the Copy Connectivity option in the Copy command, see the video, *How to Use Copy Connectivity Option in Copy Command*.

If the `enableCopyAndPaste` environment variable is enabled (`t`), in the `Copy` command, the selected objects and the reference point are automatically stored in the buffer after the reference point for the copy command has been specified. You can use the `Paste` command to paste the objects stored in the buffer. The `Copy` and `Yank` commands use the same buffer to store objects.

Creating an Array of Copies

The array of copied objects is flat, that is, all objects in the array can be selected individually. The array can be placed only in the original cellview; it cannot be copied to another cellview. You can create the following types of arrays:

- Creating an array without specifying the distance between the copies
- Creating an array by specifying the corner-to-corner distance between the copies
- Creating an array by specifying the edge-to-edge distance between the copies
- Creating multiple, overlapping copies

  Note: In the pre-selection mode, all copies of the array are selected in the end. The original object is not selected. In the post-selection mode, no object remains selected after the copies are placed. The object you copy is part of the resultant array only when you specify the corner-to-corner (*Delta*) or edge-to-edge (*Spacing*) distance.

Creating an array without specifying the distance between the copies

1. Select the object to copy.
2. Choose *Edit – Copy* and press `F3` to open the Copy Form.
3. In the *Rows* and *Columns* fields, type the number of rows and columns you want in the array, say `3` rows and `3` columns.

© 2020 All Rights Reserved.

4. Click to place the first object of the array.

5. Click to place the second column of the array.

   The rest of the columns are automatically placed. The distance between the first and second copies controls the spacing and orientation of the remaining columns.

6. Similarly, click to place the rows of the array and complete the array.

   The distance between the first and second rows controls the spacing and orientation between the remaining rows.

Creating an array by specifying the corner-to-corner distance between the copies

1. Select the object to copy.

© 2020 All Rights Reserved.

2. Choose *Edit – Copy* and press `F3` to open the Copy Form.
3. In the *Rows* and *Columns* fields, type the number of rows and columns you want in the array.
4. Select the *Delta* check box.
5. Specify the required distance in the *X* and *Y* fields.

6. Click *Apply*.

© 2020
All Rights Reserved.

An array with the specified settings is created. The object you copied is part of the array.

To apply the corner-to-corner spacing while creating copies in a series of steps:

1. In the *Copy* form, ensure that the *Delta* check box is selected.
2. In the *Rows* and *Columns* fields, type `1`.
3. In the *Step* field, specify the number of copies you want to create.
4. In the *X* and *Y* fields, specify the required values.
5. Click *Apply*.

   Copies of the selected object are created in steps.

© 2020 All Rights Reserved.

Creating an array by specifying the edge-to-edge distance between the copies

1. Select the object to copy.
2. Choose *Edit – Copy* and press `F3` to open the Copy Form.
3. In the *Rows* and *Columns* fields, type the number of rows and columns you want in the array.
4. Select the *Spacing* check box.
5. Specify the required distance in the *X* and *Y* fields.

6. Click *Apply*.

© 2020
All Rights Reserved.

An array with the specified settings is created. The object you copied is part of the array.

To apply the edge-to-edge spacing while creating copies in a series of steps:

1. In the *Copy* form, ensure that the *Spacing* check box is selected.
2. In the *Rows* and *Columns* fields, type `1`.
3. In the *Step* field, specify the number of copies you want to create.
4. In the *X* and *Y* fields, specify the required values.
5. Click *Apply*.

© 2020 All Rights Reserved.

Copies of the selected object are created in steps.

Creating multiple, overlapping copies

1. Select the object to copy.
2. Choose *Edit – Copy* and press `F3` to open the Copy Form.
3. Select the *Exact Overlap* check box.

© 2020 All Rights Reserved.

4. Specify the number of copies in the *Step* field.

5. Click *Apply*.

Multiple, overlapping copies are created. Drag the copies to place them as required.

Copying Objects to Another Cellview

To copy objects to another open cellview,

1. Select the objects you want to copy.
2. Choose *Edit – Copy* [c].
3. Move the pointer into another open cellview.
4. Click to place the copies.

Note: You cannot use the *Array* fields in the Copy form to create and copy an array of copied objects from one cellview to another. Only the original objects you selected are copied to the next cellview even if you have typed numbers in the *Rows* and *Columns* fields.

© 2020 All Rights Reserved.

Copying Cellviews Across Libraries

You can use one of the following ways to copy a (layout) cellview from a source to a target library:

- Copy using the Library Manager.
- Copy using the SKILL API dbCopyCellView.
- Copy portions of a source layout cellview to a target layout cellview using the interactive Cellview-to-Cellview Copy command in Virtuoso.

In the IC6.1 environment, if the source and target libraries are attached to different technology libraries, then these two technology libraries are compared against each other before performing the copy operation. If inconsistencies are found between the two technology libraries, the copy operation is not performed.The comparison is done to prevent future problems with the copied cellview.

During the copy operation, the following components of the source and target technology libraries are compared:

- dbuPerUU and userUnits
- Layer Names and Layer Numbers
- Purpose Names and Numbers
- Name, types, layer names and numbers of all viaDefs
- Name, types, width, and height of all siteDefs
- Names of array siteDefs

Stretching Objects

The *Stretch* command lets you stretch an edge or corner of an object.

Support for Objects	
ROD objects	The *Stretch* command fully supports ROD objects. For more information about stretching Pcells, see stretchable parameterized cell.

© 2020 All Rights Reserved.

Blockages	The *Stretch* command stretches blockages (except for placement halos) in the same way it stretches a polygon, using points. ■ You can stretch both fully and partially selected blockages. When you stretch fully selected blockages, the *Stretch* command works in the same way as the move command. When you stretch a fully selected owner, its blockage(s) will also move.
Boundaries	The *Stretch* command stretches boundaries in the same way it stretches a polygon, using points. ■ You can stretch both fully and partially selected boundaries. ■ When you stretch fully selected boundaries, the *Stretch* command works in the same way as the move command. ■ The *Stretch* command supports IOBox and CoreBoxSpec during P&R boundary stretch. Stretching retains the relative distance of IOBox and CoreBoxSpec from the P&R boundary.
Mosaics	The *Stretch* command can be used to expand or shrink mosaics. The number of rows or columns increase or decrease according to the stretching direction and the target location. ■ Mosaics can be stretched in different directions depending on the edge selected and the current *Edit Snap* mode. ■ Multiple mosaics can be selected and stretched at the same time. ■ When the *Mosaic Partial Selection* option is selected on the Selection Options form, you can partially select and stretch mosaics. In this mode, you can select edges and corners. You can also select edges from multiple mosaics in the partial selection mode.
Vias	The *Stretch* command can be used to expand or shrink vias in X or Y direction when the *Via Partial Selection* option is selected on the Selection Options form. The number of rows or columns increase or decrease according to the stretching direction and the target location.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Stretch` command is not supported for curved shapes.

This topic covers the following:

- Stretching Objects by Direct Manipulation
- Stretching a Circle

© 2020 All Rights Reserved.

- Changing the Angle of Corners
- Stretching and Moving Objects
- Stretching Paths
- Stretching Multipart Rectangles

Stretching Objects by Direct Manipulation

You can stretch objects by selecting and dragging their edges or corners.

To stretch an object,

1. Choose *Edit – Stretch* or click the *Stretch* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Stretch.*

2. Press F3 to open the form.

   The Stretch Form opens.

3. Select one or more edges or corners as follows:

Press `F4` and click an edge or corner of a polygon.

Press `F4` and click an endpoint on a path centerline.

Press `F4` and click a point on a circle or donut.

   In partial selection mode, if you press `Shift` and click an edge or corner, that edge or corner is the reference point for the stretch.

   If you create a selection box around the edges, *Stretch* prompts you for a reference point. The reference point does not have to be on the selected shapes.

4. Move the pointer over the edge until you see the stretch pointer:

5. Click and drag the pointer to stretch the object.

© 2020 All Rights Reserved.

6. Release the mouse button.

Press on a reference point. | Drag the pointer. | Release the mouse button. The object is stretched.

Note: You cannot use *Stretch* to delete objects. If you try to stretch an object to a zero area, a warning appears in the Command Interpreter Window (CIW) and a dialog box opens stating the shape was not modified because it would be illegal.

Stretching a Circle

Circles are special cases of ellipses, so that if you stretch a circle it will not remain a circle, unless you use one of the correct methods. There are two ways to stretch a circle so that it will remain a circle.

1. Set the Display Options Edit Snap Mode to diagonal.
2. Set the Selection mode to Partial.
3. Select one corner (the vertex) of the circle, as shown below:

4. Move the mouse pointer to the vertex edge until you see the stretch pointer, as shown below:

© 2020 All Rights Reserved.

You can enlarge or shrink the circle by moving the stretch pointer, and the circle will remain a circle if you stretch at a 45-degree angle, however, the origin point of the circle will not remain the same.

Caution

You must move the pointer carefully to ensure it remains at a 45-degree angle. Even though the snap mode is set to diagonal, it is still possible to move the pointer orthogonally. If you move the pointer orthogonally the circle will become an ellipse.

Stretching a Via

Partially selected vias can be stretched in X or Y direction.

1. In the Selection Options form, select *Via Partial Selection*.
2. Choose *Edit – Stretch* or click the *Stretch* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Stretch.*
3. Press F3 to open the form.

   The Stretch Form opens.
4. Select the corner of the via that you want to stretch.
5. Move the pointer over the edge until you see the stretch pointer.
6. Click and drag the pointer to stretch the via. You can stretch the via in X or Y direction.
7. Release the mouse button.

Changing the Angle of Corners

To change the angle of a corner by stretching,

1. Choose *Edit – Stretch* [s] or click the *Stretch* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Stretch.*
2. Press F3 to open the form.

   The Stretch Form opens.

© 2020 All Rights Reserved.

3. Turn *Lock Angles* off.
4. Stretch the corner where you want and click.

The selected corner

With *Lock Angles* on, you cannot change the angle of the corner.

With *Lock Angles* off, you can stretch the corner in any direction.

Note: If you create a non-orthogonal segment while stretching the corner of a multipart path, subrectangles cannot regenerate in that particular segment. (Subpaths can and will regenerate.)

Stretching and Moving Objects

You can simultaneously stretch and move objects. This is often called a window stretch.

To stretch and move objects,

1. Choose *Edit – Stretch* [`s`] or click the *Stretch* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Stretch.*

2. Press F3 to open the form.

   The Stretch Form opens.

3. Press `Shift` and create a selection box around the objects you want to move and the edges you want to stretch.

4. Select a reference point for the stretch.

© 2020 All Rights Reserved.

5. Move the pointer to stretch the edges where you want.

6. Click.

The stretched and moved objects

Stretching Paths

You can stretch the ends, segments, and/or corners of a path. You stretch ends and corners in the same way you stretch any other object. To stretch segments, you need to select the path centerline in the segment(s) you want to stretch.

The resultant of stretching a path should not be malformed or self-intersecting (one with overlapping edges). A resultant path with coincident edges (abutted edges) can be manufactured and is allowed.

To stretch path segments,

1. Choose *Edit – Stretch* [s].

   Stretch automatically changes to partial selection mode and prompts you to select the object you want to stretch.

2. Select one or more segments by doing one of the following:

© 2020 All Rights Reserved.

- For a single segment, click the middle of the centerline of the segment or create a selection box around the centerline of the segment, making sure to include the ends of the segment centerline.

Highlighting shows the path segment is selected.

- For multiple segments, to add another segment, press `Shift` and click or create a selection box around the centerline of the segments, making sure to include the ends of the segment centerlines.

Highlighting shows the path segments are selected.

When you use a selection box, *Stretch* prompts you for a reference point. The reference point does not have to be on the selected shapes.

3. If you are prompted for a reference point, click where you want the stretch to begin.

© 2020 All Rights Reserved.

4. Point to the new location for the stretch and click.

5. To end the *Stretch* command, press `Escape`.

Stretching Multipart Rectangles

You can stretch the edges and/or corners of a master rectangle of a multipart rectangle in the same way you stretch regular rectangles, click an edge or corner, then click at a new location. The system regenerates the arrays of subrectangles associated with the stretched master rectangle. How the subrectangles regenerate depends on the definition of the multipart rectangle. You cannot stretch a master rectangle separately from its associated arrays of subrectangles, nor can you stretch an array of subrectangles separately from its master rectangle.

For information about defining multipart rectangles, see "Multipart Rectangles" and the `rodCreateRect` function in the *Virtuoso Relative Object Design SKILL Reference.*

To stretch an edge of multipart rectangle,

1. Choose *Edit – Stretch* [`s`].

   Stretch automatically changes to partial selection mode and prompts you to select the object you want to stretch.

2. Select the edge you want to stretch by clicking on it:

© 2020 All Rights Reserved.

The selected edge is highlighted.

Stretch prompts you to point to the new location. In this example, the master rectangle is stretched to the right.

3. Point to a new location for the stretch.

© 2020 All Rights Reserved.

Deleting Objects

Support for Objects	
ROD objects	The *Delete* command fully supports ROD objects. When you delete using the *Net Interconnect* option, all ROD objects on the selected net are deleted, except for pins. For multipart paths (MPP) and multipart rectangles (MPR), when the master path or master rectangle is on the selected net, the MPP or MPR and all subparts are deleted; otherwise, no part of the MPP or MPR is deleted. When you delete a segment of an MPP with the *Path Segment* option, the remaining segments of the MPP become two separate multipart paths, each with its own name.
Boundaries	■ You can delete pre- or post- selected boundaries. ■ You cannot delete partially selected edges of boundary objects.
Blockages	■ You can delete pre- or post- selected blockages. ■ You cannot delete partially selected edges of the blockage object. ■ When you delete the owner of a blockage, the blockage object is also deleted.

Deleting Objects, Edges and Corners

To delete an object,

1. Choose the objects, edges, or corners you want to delete.
2. Choose *Edit – Delete* or press `Delete` or click the *Delete* icon on the *Edit* toolbar.

   All selected objects are deleted.

   If you selected an edge or corner, the edge or corner is deleted and the object is redrawn.

Note: You cannot delete a point if it changes the shape so that the shape becomes invalid. For example, a path must have at least two points; if you try to delete one point of a two-point path, you see a warning dialog box. You must close the dialog box and choose a different point to delete.

© 2020 All Rights Reserved.

Using the Delete Form

To delete net interconnect, path segments, or device chains,

1. Choose *Edit – Delete* or click the *Delete* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Delete.*

2. Press F3 to open the form.

   The Delete Form opens.

3. In the form, select the type of objects to be deleted.

   If you select the *Net Interconnect* option, all ROD objects and shapes on the selected net, except pins, are deleted.

4. Select the objects to delete in the design canvas.
5. Click *Cancel* in the form.

Canceling Delete

If you discover you deleted the wrong object,

➤ Choose *Edit – Undo* [`u`] to restore the object.

Delete automatically repeats if you select the *Delete* command first and then choose objects.

➤ If delete automatically repeats, press `Escape` to cancel the command.

Note: You can cancel an ongoing *Delete* command by pressing `Ctrl` + `C`.

Copying Objects in Repeat Mode

The *Repeat Copy* command enables you to repeatedly place copies of the selected object. The command supports both pre-selection and post-selection modes. For a simple copy-paste operation, see Copying Objects.

To repeatedly place copies of an object:

1. Select *Edit – Repeat Copy* [`h`] or click the *Repeat Copy* icon on the *Edit* toolbar.

© 2020 All Rights Reserved.

2. Click the object to copy.

A copy of the object attaches to the pointer.

Note: You cannot change the selection set after you have placed the first copy.

3. Press `F3`.

The Repeat Copy Form appears.

4. From the *Snap Mode* list, select a direction in which to place the copy.

5. Click to place the first copy.

The distance (*d*) between the object and its first copy and the direction in which the copy is placed determine the placement distance and direction of the subsequent placed copies. After you place the first copy, the distance and direction are fixed.

© 2020 All Rights Reserved.

Alternatively, specify the distance at which to place the copy in the *Delta X* and *Y* fields in the *Repeat Copy* form and click the *Apply XY* button. The copy is placed at the specified distance from the object.

6. Press dot (`.`) to repeat placing the copy.

The previous placed object is taken as the reference. A copy is placed at the same *d* distance from the previous copy and in the same direction in which the previous copy was placed.

If you specified the *Delta* distance, it will be used to place the remaining copies.

You can continue to press `.` to place the copy as many times as required.

7. To start placing the copy from a different location, while retaining the distance:

a. Press `F3`.

b. In the *Repeat Copy* form, select the *Release Direction* check box.

The option is grayed out after you select the check box.

c. Click *Hide*.

You can move the pointer freely in any direction. You can place the copy in any direction and at any distance from the previous placed copy.

d. Click to place the copy.

© 2020 All Rights Reserved.

The copy is placed at the clicked location and a copy of the object attaches to the pointer.

e. Move the pointer and click at the next location to place the copy.

To place this copy, the pointer movement is limited to only either horizontal or vertical direction, depending on the earlier direction in which the copy was placed. For example, if the earlier direction of placing the copy was along the X axis, you will be able to move the pointer only to the left or right of the placed copy. If the earlier direction of placing the copy was along the Y axis, you will be able to move the pointer only to the top or bottom of the placed copy.

The same distance *d* will be used to place the copy. *Delta* will be used if you specified it in the *Repeat Copy* form.

f. Press . to place the copy in the fixed direction at the fixed distance.

Final Result

© 2020 All Rights Reserved.

8. Press `Esc` or click *Cancel* in the form to finish placing the copies.

Explanation of the Steps to Reach the Resultant

The original copied object remains selected throughout.

You can use the `leRepeatCopyMoveStretch` SKILL function to copy, move, or stretch the selected objects and apply the same transform as done in the last copy, move, or stretch command and select the newly created object.

Video

For details on how to use the `leRepeatCopyMoveStretch` SKILL function, see the video *How to Use leRepeatCopyMoveStretch() API*.

Flipping Objects

You can flip an object horizontally or vertically. To flip multiple selected objects around the center of the boundary box of the selected set, use the *Flip Horizontal* and *Flip Vertical* commands on the *Edit – Flip* menu. To flip an object by specifying a reference point, use the Flip Form. The *Flip* command works in both pre-selection and post-selection modes.

While editing an instance or mosaic, you can use the *Instance Editing Mode* option on the Layout Editor Options Form to specify whether the *Flip* command uses the instance bounding box, the PR boundary, or the snap boundary (if defined) as the instance border for the operation. For example, if you have a standard cell with a large label that extends beyond the

© 2020 All Rights Reserved.

PR boundary, the bounding box of that cell will be much larger than its place and route boundary; you can set *Instance Editing Mode* to *P&R Boundary* to have the *Flip* command ignore the bounding box and consider only the PR boundary when operating on the cell.

The *Edit – Flip – Flip Horizontal* and *Edit – Flip – Flip Vertical* commands are available only in pre-selection mode. If you flip multiple selected objects, the *Flip Horizontal* and *Flip Vertical* commands, by default, flip the objects around the center of the boundary box of the selected objects (*Flip Around Combined Center* check box in the Layout Editor Options Form is on by default). If the *Flip Around Combined Center* check box is off, the selected objects are flipped around their own centers.

Note: You must set the stop display level to greater than zero before flipping geometries in the via instances.

Note: For `Copy` and `Move` commands, the `Flip` option on the toolbar is disabled.

For information about the *Rotate* command, see Rotating Objects.

To flipping the selected objects by specifying a reference point:

1. Select the objects to be flipped.
2. Choose *Edit – Flip*.
3. Press `F3`.

   This opens the Flip Form.

4. You can do one of the following:
   - ❑ Do not select the *Specify reference point* check box in the *Flip* form.

     a. In the design display area, click at the reference point for flipping the objects.

     The coordinates of the clicked reference point populate in the reference point field.

© 2020 All Rights Reserved.

The ghost image of the resultant displays. The location of the ghost image moves as you move the pointer.

The figure below shows the selected objects and the reference point.

Assuming that the selected objects are in quadrant 2:

- If the pointer is in quadrant 1 relative to the selected objects, the ghost image is also displayed in quadrant 1.

© 2020 All Rights Reserved.

- If the pointer is in quadrant 3 relative to the selected objects, the ghost image is also displayed in quadrant 3.

- If the pointer is in quadrant 2, no ghost image is displayed.
- If the pointer is in quadrant 4, nothing changes. If a ghost image is displayed in quadrant 1 or 4, it continues to be displayed.

b. Click to select the required ghost image.

The selected objects are flipped and the command ends.

In the this mode, the *Flip Horizontal* and *Flip Horizontal* buttons on the form are disabled.

© 2020 All Rights Reserved.

- Specify a specific point in the Flip form.

a. Select the *Specify reference point* check box.

a. Specify the coordinates of the point at which you want to flip the object in the reference point field.

b. Click *Flip Horizontal* or *Flip Horizontal*, as required.

The objects are flipped around the specified reference point, as shown below.

Rotating Objects

The *Rotate* command enables you to change the orientation of most geometric objects.

Support for Objects	
Donuts and dots	You can rotate any object except donuts and dots with the *Rotate* command. To rotate donuts and dots, use the *Move*, *Copy*, or *Paste* command.
Instances and labels	Instances and labels cannot be rotated to other than 90-degree increments.

© 2020 All Rights Reserved.

ROD objects	With the exception of multipart rectangles, the *Rotate* command supports the rotation of ROD objects, unless there are alignments. Avoid rotating aligned ROD objects because aligned handle names are not updated after the rotation, so the results might not be what you want. For multipart rectangles with subrectangles offset from the master rectangle corners, avoid rotation unless the offsets are all equal. The Rotate command does not rotate these offsets. Do not rotate chopped multipart paths as the chop holes might not rotate correctly. If you want to chop and rotate an MPP, rotate the MPP first, then chop it.
Blockages	The *Rotate* command fully supports blockage objects. The *Rotate* command permits any angle rotations when blockage objects are part of the selected set ■ When you rotate blockage objects without owners, the coordinates are updated. ■ When you rotate owners of blockage objects, the blockage coordinates are also updated.
Boundaries	The *Rotate* command fully supports boundary objects. The *Rotate* command permits any angle rotations when boundary objects are part of the selected set. The *Rotate* command supports the rotation of IOBox and CoreBoxSpec when P&R boundary is rotated.

Objects that are on-grid remain on-grid after rotation.

You can rotate an object around its center by rotating it left or right. The commands to perform these actions are available on the *Edit – Rotate* menu—*Rotate Left* and *Rotate Right*. These commands are available only in pre-selection mode. If you rotate multiple selected objects, the *Rotate Left* and *Rotate Right* commands, by default, rotate the objects around the center of the boundary box of the selected objects (*Rotate Around Combined Center* check box in the Layout Editor Options Form is on by default). If the *Rotate Around Combined Center* check box is off, the selected objects are rotated around their own centers.

While editing an instance or mosaic, you can use the *Instance Editing Mode* option on the Layout Editor Options Form to specify whether the *Rotate* command uses the bounding box, the PR boundary, or the snap boundary (if defined) as the instance border for the operation. For example, if you have a standard cell with a large label that extends well beyond the PR

© 2020 All Rights Reserved.

boundary, the bounding box of that cell will be much larger than its place and route boundary; you can set *Instance Editing Mode* to *P&R Boundary* to have the *Rotate* command ignore the bounding box and consider only the PR boundary when operating on the cell.

You can also rotate an object by specifying a reference point for rotation and an angle by which to rotate the object. The options for specifying these settings are available in the Rotate Form (*Edit – Rotate – Rotate*). The settings in the *Rotate* form rotate multiple selected objects as a group with respect to the specified reference point (even if *Rotate Around Combined Center* check box in the Layout Editor Options Form is off). For information about how to flip an object by specifying a reference point, see Flipping Objects.

Note: For `Copy` and `Move` commads, the `Rotate` option on the toolbar is disabled.

You can perform the following rotate operations:

- Rotating a Geometric Object by Any Angle
- Rotating Multiple Selected Objects
- Rotating or Mirroring an Object
  - Using the Mouse to Rotate or Mirror Objects
- Rotating Instances and Labels

To flip objects, see Flipping Objects.

Rotating a Geometric Object by Any Angle

To rotate an object by using the *Rotate* form:

1. Choose *Edit – Rotate – Rotate* or click the *Rotate* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context-sensitive menu and select *Rotate – Rotate.*

2. Select the object you want to rotate.

© 2020 All Rights Reserved.

3. Click at a reference point.

4. Press `F3`.

The Rotate Form opens.

5. In the *Angle* field, type a degree value or click any one of the rotate buttons.

6. Click *Apply*.

© 2020
All Rights Reserved.

The figures below illustrate the use of *Rotate 90*, *Rotate 180*, and *Rotate 270* buttons. If you click any of these buttons, a ghost image appears to give you a preview of the resultant of the rotation, as shown below. You get the rotated object after you click *Apply*.

Rotate 90

Rotate 180

Rotate 270

To rotate an object by using the pointer:

1. Choose *Edit – Rotate – Rotate* or click the *Rotate* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design display area to display the *Layout* context-sensitive menu and select *Rotate – Rotate.*

2. Select the object you want to rotate.

3. Click a reference point and move the pointer.

© 2020 All Rights Reserved.

The object rotates and the current rotation angle appears.

4. Click to stop the rotation.

   The object is rotated.

Rotating Multiple Selected Objects

To rotate multiple selected objects around their respective center points:

1. Select the objects you want to rotate.
2. In the Layout Editor Options Form, switch off the *Rotate Around Combined Center* check box.
3. Choose *Edit – Rotate* or right-click and select *Rotate* from the context-sensitive menu.

   If the selected objects are all shapes, the *Shape* context-sensitive menu displays. If the selected objects are of different types, the *Multiple* context-sensitive menu displays.

4. From the *Rotate* menu, select *Rotate Left* or *Rotate Right*, as required.

Multiple Selected Objects

© 2020 All Rights Reserved.

The figures below show the resultants when the *Rotate Left* or *Rotate Right* command is run on the selected set of objects shown in the previous figure.

Rotate Left

Rotate Right

To rotate multiple selected objects around the center of the selection set:

1. In the Layout Editor Options Form, ensure that the *Rotate Around Combined Center* check box is selected.
2. Select the objects you want to rotate.

Multiple Selected Objects

3. Select *Edit – Rotate* or right-click and select *Rotate* from the context-sensitive menu.
4. From the *Rotate* menu, select *Rotate Left* or *Rotate Right*, as required.

© 2020 All Rights Reserved.

The figures below show the resultants when the *Rotate Left* and *Rotate Right* commands are run on the selected set of objects.

Rotate Left

Rotate Right

Rotate Around Combined Center is selected.

To rotate multiple selected objects as a group and by specifying a rotation reference point:

1. Choose *Edit – Rotate – Rotate* and press `F3` to open the Rotate Form.
2. Select the objects you want to rotate.
3. Click at a reference point.

4. In the *Rotate* form, specify a value in the *Angle* field.
5. Click *Apply*.

© 2020 All Rights Reserved.

The figure below illustrates the rotation when you specify an angle of 20 degrees.

Rotating or Mirroring an Object

To rotate any object in 90-degree increments or to mirror an object, use the *Move*, *Copy*, or *Paste* command. To rotate most geometric objects to any angle, use the *Rotate* command.

To rotate an object using the *Move* Command

1. Choose *Edit – Move* [m].

   The Move Form opens.

2. Select the objects you want to rotate.

3. Click the *Rotate 90*, *Rotate 180*, or *Rotate 270*button.

   Continue clicking the buttons to further rotate or mirror the object.

© 2020 All Rights Reserved.

Each time you click *Rotate 90,* the object rotates 90 degrees counterclockwise.

Rotate 180 mirrors the object along the X axis.

Rotate 270 mirrors the object along the Y axis.

4. Click where you want to place the rotated object.

Using the Mouse to Rotate or Mirror Objects

You can use the middle mouse button to rotate or mirror objects.

While using *Move*, *Copy*, or *Paste*, do one of the following:

© 2020 All Rights Reserved.

- To rotate the object 90 degrees, click the middle mouse button.
- To mirror the object, press `Shift` and click the middle mouse button.

The object mirrors first along the X axis, then along the Y axis.

Rotating Instances and Labels

To rotate an instance or label by using the *Rotate* form:

1. Choose *Edit – Rotate – Rotate*.
2. Select the instance or label you want to rotate.
3. Press `F3` to open the Rotate Form.

   Alternatively, you can select an instance and right-click it to display the *Instance* context-sensitive menu. Choose *Rotate – Rotate* and press `F3`.
4. In the *Rotate* form, the *Angle* and *Angle Snap To* fields are grayed out because instances and labels can be rotated only at 90-degree increments.
5. Click the reference point.
6. To rotate the object,
   - 90 degrees, click *Rotate 90*.
   - 180 degrees, click *Rotate 180*.
   - 270 degrees, click *Rotate 270*.

To rotate multiple selected instances or labels around their respective center points:

1. Select the objects you want to rotate.
2. In the Layout Editor Options Form, switch off the *Rotate Around Combined Center* check box.
3. Choose *Edit – Rotate.*
4. Alternatively, right-click the selected instance and click *Rotate* on the *Instance* context-sensitive menu.
5. Choose *Rotate Left* or *Rotate Right*.

© 2020 All Rights Reserved.

Tip

To rotate multiple selected objects as a group, use the options on the Rotate Form or select the *Rotate Around Combined Center* check box in the *Layout Editor Options* form.

Rotating Objects: Virtuoso Photonics Solution (ICADVM20.1 Photonics Only)

Note: You will need the `Virtuoso_Photonics_Option` license to use the functionality discussed in this section. For information on obtaining the required license, contact your local Cadence representative.

You can rotate instances by any angle if it has a parameter that can be used for rotation.

Note: The value of the cdsRotateInstParam parameter on the instance or master is considered as the rotation parameter. If the cdsRotateInstParam is not specified, the value specified in the `rotateInstParam` environment variable is considered for rotation.

For orthogonal rotation of an instance, the *Rotation* field on the *Attribute* tab of the Edit Photonics Pin Properties form is updated.

For non-orthogonal rotation of an instance, the *rotation* parameter on the *Parameter* tab of the Edit Photonics Pin Properties form is updated. The rotation parameter is defined by the rotateInstParam environment variable.

Chopping Objects

The *Chop* command lets you cut away part of an object or cut an object into pieces. If a non-ROD object has a net name and it is cut into pieces, each remaining piece of the object retains the net name.

Note: Partial chopping of instances and vias is not supported.

Support for Objects

© 2020 All Rights Reserved.

ROD objects	Most shapes lose ROD attributes when they are chopped. For multipart paths, the effect of a chop depends on what parts are choppable and how you chop them. For details, see How Chopping Affects Multipart Paths. When you chop the master rectangle of a multipart rectangle, all subrectangles are deleted and the shapes resulting from chopping the master rectangle are regular, unnamed database shapes. When you chop any ROD object, all alignments to that object are deleted.
Blockages	Chopping a blockage object may result in more than one blockage object. If the original blockage object had an owner, then all the resulting blockage objects will have the same owner. One owner will then have multiple associated blockages.

© 2020 All Rights Reserved.

Boundaries	The *Chop* command does not support snap boundary objects. ■ For P&R, area and cluster boundaries, the command allows rectilinear chops only, therefore only horizontal, vertical and orthogonal snap modes are allowed. Any other snap mode will override to orthogonal. ■ For P&R boundary, after a chop, the resultant P&R boundary snaps to the grid. They are snapped to the grid specified by the environment variables, boundarySnapXGrids or boundarySnapYGrids. If the environment variables are not specified, the resultant P&R boundary is snapped based on the least common multiple of the global active grids. ■ After a chop, the resultant P&R boundary must remain a single object. If you chop the P&R boundary into more than one piece, the chop does not take place and the CIW issues the following warning: *WARNING* Can't chop the PR Boundary into two or more pieces. ■ The *Chop* command supports editing of IOBox and CoreBoxSpec when P&R boundary is chopped. If chopping results in a non-rectangular P&R boundary, the rectangle containing all P&R boundary points is considered. The relative distance of IOBox and CoreBoxSpec from the P&R boundary is maintained in all directions, even until the IOBox and CoreBoxSpec shapes are reduced to zero area. ■ After chopping a cluster boundary, each cluster boundary formed by the chop will remain connected to the original cluster. If you chop a cluster boundary object, each new cluster boundary has an extension added to its name, with a resulting informational message appearing in the CIW.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Chop` command is not supported for curved shapes.

To cut hierarchical objects, such as parameterized cells, use the *Yank* command and see Copying and Cutting Through the Hierarchy. For information about chopping multipart paths, see Modifying Corners.

To cut away part of an object:

1. Choose *Edit – Basic – Chop* [`Shift-c`]. Press F3 to open the form.

© 2020 All Rights Reserved.

The Chop Form opens.

2. Set *Chop Shape* to *rectangle, line,* or *polygon*. If you select *line* or *polygon*, set the *Snap Mode*.
3. You can split an object by turning off the *Remove chop* option in the Chop form and using a rectangle or polygon chop shape. This saves both the area you cut and the original objects.
4. Select one or more objects.
5. Click to enter the point of the chop.

- Depending on the options that are set, the object area is cut away or the object is split.

Create a line that intersects a rectangle.

The rectangle is split into three rectangles.

- If you cut a hole in an object, it is redrawn as a polygon with a cut line.

- Paths convert to polygons when *Chop Shape* is set to *polygon* or *line* and *Snap Mode* is set to *anyAngle* or *diagonal*.

© 2020 All Rights Reserved.

- If you cut a path with a diagonal line, the resulting object is a polygon.

Cutting the path with a diagonal line

After cutting, the path has been converted to a polygon.

For more information about how parent-child relationship behaves when either the parent or the child is chopped, see Attaching Objects.

Chopping Vias

The *Chop* command also lets you cut away part of a via object or cut a via object into pieces. A few cases of how a via is chopped are described below.

- When a via is chopped with the *Chop Shape* set to *rectangle*, the via is chopped like other shapes, as shown in the figure below.

Rectangular Chop

Chopped via

© 2020 All Rights Reserved.

- When a via is chopped with the *Chop Shape* set to *rectangle* and the *Remove chop* check box selected, two distinct vias are created, as shown in the figure below.

Rectangular chop with *Remove Chop* check box selected

Chopped vias - Two distinct vias created

- When a via is chopped with the *Chop Shape* set to *rectangle* and the *Remove chop* check box not selected, three distinct vias are created, as shown in the figure below.

Rectangular chop with *Remove Chop* check box not selected

Chopped vias - Three distinct vias created

Note: If the *Chop Shape* is set to *rectangle or line*, two distinct vias for each original via are created. However, if the *Chop Shape* is set to *polygon*, then *n* distinct vias are created. However, a limitation in case of chopping vias is that if any region (of the chop shape) is non-rectangular, then a via is not created in the non-rectangular region.

If you are chopping a via stack, each via in the viaStack is chopped independently.

© 2020 All Rights Reserved.

Merging Objects on a Layer

You can merge one or more objects created on the same layer into one object. This is sometimes referred to as performing a logical AND on the objects.

Support for Objects	
ROD objects	The result of merging ROD objects is a ROD object. Merging a ROD object with any non-ROD shape produces a non-ROD shape. If the ROD objects being merged are on the same net, the resultant ROD object retains the net. If the ROD objects being merged are on different or no nets, the resultant ROD object will not be on any net. The result of merging two “identical” MPPs is an MPP only when the merging of their master paths results in a path and not a polygon. If the result of merging the master paths is a polygon, the resultant is a ROD object. However, the subparts are not ROD objects; only the polygon on the base layer of the MPP is a ROD object.
Blockages	The *Merge* command merges blockage objects in the same way it merges polygons. ■ If one or all selected blockages are attached to the same instance, the resulting merged blockage will remain attached to the instance. ■ If the blockages are attached to different instances, a scroll list displays so that you can choose which instance will own the resulting merged blockage. One of the options is no attachment.
Boundaries	The *Merge* command does not support P&R and snap boundary objects. ■ If an area boundary object merge involves more than one constraint group, a scroll list will pop up, allowing you to choose the constraint group for the merged area boundary. You must also rename the area boundary. ■ If a cluster boundary object merge involves more than one cluster, a scroll list will pop up, allowing you to pick a destination cluster. You must also rename the cluster boundary.

© 2020 All Rights Reserved.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Merge` command is not supported for curved shapes.

To merge objects,

1. Choose *Edit – Basic – Merge* [`Shift-m`].
2. Select one or more objects on the same layer. The objects must touch or overlap each other.

   The objects are merged.

   You can click and drag to create a selection box and merge several objects at once. Only those objects that are on the same layer and that touch will be merged.

Select a group of objects while using *Merge*.

Objects on the same layer are merged.

Video

For details on how to use the *Merge* command in pre-select and post-select modes, see the video *Using the Merge Command in Pre-select and Post-select Modes*.

How Merging Converts Objects

When you merge objects using Merge or Layer Generation, most merged objects are converted to polygons. If you merge a group of polygons and paths, the resulting object is a polygon.

However, when you merge paths of the same width abutting each other, they merge into a single path. When you merge overlapping paths, they merge into a polygon or rectangle.

Overlapping or abutting rectangles of the same width adjoining end to end merge into rectangles.

© 2020 All Rights Reserved.

When you merge objects, the resulting polygon should not have more than 4000 points (vertices).

Note: When ROD objects are merged, the resulting shape is not a ROD object

A polygon and a path before merging.

After merging, the new object is a polygon.

How Merging Converts Paths

Merging Collinear Paths

When you merge abutted collinear paths, the merged path must have the same width as the original path and should be on the same layer. If the path centers are not aligned, the merged path becomes a polygon. To ensure the path centers are aligned, you can change the x points in the *Edit Path Properties* form for each vertical path or the y points for each horizontal path.

Abutting paths with the same width before merging.

After merging, the new object is a path.

When you merge overlapping collinear paths, the merged path must have the same width and be on the same layer.

Overlapping collinear paths with the same width before merging.

After merging, the new object is a path.

© 2020 All Rights Reserved.

Merging Non-Collinear Paths

When you merge overlapping horizontal and vertical paths, the following conditions should be met for the resulting object to be a path:

- For each end point of the paths at the overlapping ends, this criterion should be met:

  center line of the other path =< end point of the path =< outer edge of the other path

- For each outer edge of the paths at the overlapping ends, the outer edge should not extend beyond the outer edge of the other path.

If either of these conditions is not met, the resulting object becomes a polygon.

In the following figure, both the above conditions are satisfied by the overlapping paths. Therefore, the resulting merged object is a path. The path ends move to form a right angle to make the path design rule correct.

In the following figure, the outer edge of the horizontal path extends beyond the outer edge of the vertical path. Therefore, the resulting object is a polygon.

© 2020 All Rights Reserved.

You can merge collinear paths and collinear multipart paths with multiple end types. The supported end types are truncate, extend, round, and variable. Some examples are shown below.

A: The two paths to the left are extend and round paths. The merged path is retained as an extend path with the same offset at the top as the extend path. The 0.5 width of the round path is retained at the end of the new extend path.

B: The two middle paths are round and truncate.The merged path is variable with the truncate end of the original bottom path retained and the 0.5 width of the round end retained at the end of the new variable path.

C: The two paths to the right are truncate and variable. The merged path is a variable path with the truncate end at the top path retained and the 0.7 offset retained of the original bottom path.

All paths being merged must be of the same width. The following two tables summarize the results of merging various path ends:

- In the first table, the resultant object of all merging combinations (columns 1 and 2) is a path (column 3). The table provides examples of begin extension, end extension, or width in the original paths to indicate the parameter values retained in the resultant merged path.

© 2020 All Rights Reserved.

w represents one half the path width of the round or extend path.
b represents begin extension.
e represents the end extension of any variable path end.

Type of Path 1		Type of Path 2		Type of Resultant Merged Path	
Truncate	(b0:e0)	Truncate	(b0:e0)	Truncate	(b0:e0)
Truncate	(b0:e0)	Extend	(w0.5:w0.5)	Extend	(b0:w0.5)
Truncate	(b0:e0)	Variable	(b2:e2)	Variable	(b0:e2)
Extend	(w0.5:w0.5)	Extend	(w0.5:w0.5)	Extend	(w0.5:w0.5)
Extend	(w0.5:w0.5)	Truncate	(b0:e0)	Truncate	(w0.5:e0)
Extend	(w0.5:w0.5)	Variable	(b2:e2)	Variable	(w0.5:e2)
Round	(w0.5:w0.5)	Round	(w0.5:w0.5)	Round	(w0.5:w0.5)
Variable	(b1:e1)	Variable	(b2:e2)	Variable	(b1:e2)
Variable	(b1:e1)	Truncate	(b0:e0)	Truncate	(b1:e0)
Variable	(b1:e1)	Extend	(w0.5:w0.5)	Extend	(b1:w0.5)

Note: A truncate path always has zero **b** and **e**.

- The second table reflects that the merging of any path with a round path results in a polygon.

Type of Path 1	Type of Path 2	Resultant Merged Object
Truncate	Round	Polygon
Extend	Round	Polygon
Round	Truncate	Polygon
Round	Extend	Polygon
Round	Variable	Polygon
Variable	Round	Polygon

© 2020 All Rights Reserved.

Yanking and Pasting Objects

The `Yank` command performs a “cookie cutter” copy. `Yank` copies objects, and parts of objects, into a temporary buffer. The `Paste` command places copies of objects from a temporary buffer.

Support for Objects	
ROD objects	The Yank and Paste commands do not support ROD objects. The Yank command copies only the shapes, but not the names of the shapes or the alignments. When you paste, the result is unnamed, unaligned, non-ROD shapes
Blockage	The Yank and Paste commands do not support blockage objects. The *Yank* command copies only the shapes and ignores the blockage objects.
Boundaries	The Yank and Paste commands do not support boundary objects. The *Yank* command copies only the shapes and ignores the boundary objects.

Yank and Paste are similar to Copy except that objects are chopped by the Yank shape. Yank/Paste is a “cookie cutter” copy. The `Yank` command is available both in read-only and edit modes.

Note: The `Yank` command is also supported for mosaics.

Yanking and Pasting Objects in Pre-Selection Mode

You can pre-select objects before you invoke the `Yank` command. In the pre-selection mode, only fully selected objects are yanked. Partially selected objects are ignored by the `Yank` command in the pre-selection mode.

To copy an object, group of objects, or an instance:

1. Select the object that you want to copy.

The selected object is highlighted.

2. Choose *Edit – Basic – Yank* [`y`].

© 2020 All Rights Reserved.

3. Choose *Edit – Basic – Paste* [`Shift+y`].

   Outlines of the objects you yanked follow the pointer.

4. Click to place the copied objects.

Note: In the pre-selection mode, the `Yank` command does not support boundaries (PR, Snap, Area, and Cluster), blockages, guides, and steiners. In case the selected set contains any of the above-mentioned objects, a warning message is displayed in the CIW.

You can copy a group of objects as well using the `Yank` and `Paste` commands.

- When a group of objects are selected, all the members of the group are yanked. When you paste, the newly created objects become members of the newly created group.
- When you want to edit the contents of a group (*Edit – Hierarchy – Edit In Place*), select the shapes that you want to yank. When you paste, the newly created shapes remains a member of the group from which they were copied.

- When you want to edit the contents of a group (*Edit – Hierarchy – Edit In Place*), select the shapes that you want to yank. When you paste after you return (*Edit – Hierarchy – Return to Level*), the newly created shape is not a member of the group from which they were copied.

© 2020 All Rights Reserved.

- When the transparent group mode is enabled, select the shapes that you want to yank. When you paste, the newly created shape is not a member of the group from which they were copied.

For information about how parent-child relationship behaves when either the parent or the child is yanked, see Attaching Objects.

Yanking and Pasting During Edit In Place

If you are editing a cell instance in place, you can select the objects to be yanked and then paste these objects. You can also paste in the top cellview even after you return from Edit In Place, using *Edit - Hierarchy - Return*.

Yanking and Pasting Objects in Post-Selection Mode

To copy part of an object, group of objects, or an instance:

1. Choose *Edit – Basic – Yank* [`y`]. Press F3 to open the form.

   The Yank Form opens.

   In the Yank form the Yank Shape is set by default to create a rectangle.

2. If you are in full selection mode, press `F4` to go to partial selection mode.

3. Create the yank shape around the objects you want to copy.

   All objects and parts of objects inside the box are copied into a special yank buffer. If an instance is inserted by the *Yank* shape, the objects in it are brought to the top level (flattened).

4. Choose *Edit – Basic – Paste* [`Shift+y`].

© 2020 All Rights Reserved.

Outlines of the objects you yanked follow the pointer.

5. Click to place the copied objects.

The copies appear where you clicked.

Note: You can paste an object whose layer is not visible in the *Layers* panel of the Palette assistant. However, you cannot yank an object whose layer is not visible in the *Layers* panel of the Palette assistant and whose object type is not visible in the *Objects* panel of the Palette assistant.

Editing Instance Properties

For information about properties, attributes and about how to edit properties, see Editing and Defining Properties.

Reshaping Objects

The *Reshape* command lets you change the shape of a selected object.

Support for Objects	
ROD objects	Except for multipart paths and multipart rectangles, all ROD objects that can be reshaped remain ROD objects after a reshape operation. When you reshape any ROD object, all alignments to that object are deleted.
Rows	The *Reshape* command does not support rows.
Groups	The *Reshape* command does not support group objects.
Blockages	The *Reshape* command reshapes blockages in the same way it reshapes a polygon.

© 2020 All Rights Reserved.

Boundaries	The *Reshape* command reshapes P&R and cluster boundaries in the same way it reshapes a polygon, using points. The *Reshape* command supports reshaping of IOBox and CoreBoxSpec when P&R boundary is reshaped. The relative distance of IOBox and CoreBoxSpec from the P&R boundary is maintained. The *Reshape* command does not support snap boundary objects.

Reshaping Polygons

To reshape a polygon,

1. Choose *Edit – Advanced – Reshape* [`Shift-r`]. Press F3 to open the form.

   The Reshape Form opens.

2. Select a polygon.
3. Turn on *Reshape Type: line*.
4. Create the new section of the polygon.

The first and last points must touch the original shape.

5. Double-click when you are finished entering points.
6. Click the middle mouse button to toggle between highlighting the new shape and highlighting both the old and new shape.

Highlight the new shape to replace the original shape.

Highlight both shapes to reshape the original shape.

7. When the shape you want is highlighted, click.

Adding a Rectangle to a Polygon

To add a rectangle to a polygon,

1. Choose *Edit – Advanced – Reshape* [`Shift-r`]. Press F3 to open the form.

© 2020 All Rights Reserved.

The Reshape Form opens.

2. Select a polygon.
3. Turn on *Reshape Type: rectangle*.
4. Create a rectangle that intersects the polygon.
5. Click the middle mouse button to toggle between highlighting the new shape and highlighting both the old and new shape.

Highlight the new shape to replace the original shape.

Highlight both shapes to reshape the original shape.

6. When the shape you want is highlighted, click.

Reshaping a Path

To reshape a path,

1. Choose *Edit – Advanced – Reshape* [`Shift-r`].

   The Reshape Form opens.

2. Select a path.
3. Turn on *Reshape Type: line*.
4. Starting from the centerline, create the new section of the path.

Start the new segment from the path centerline.

5. Double-click when you are finished entering points.
6. Click right to toggle between highlighting the reshaped path options.

Reshape options

© 2020 All Rights Reserved.

7. When the shape you want is highlighted, click.

Splitting Objects

The *Split* command lets you split and stretch a portion of an object. You usually use it to add a jog by stretching a section of a path or group of paths (a bus). You can also use Split to add a jog by stretching a section of a multipart path.

Support for Objects	
ROD objects	Except for ellipses, circles, donuts, multipart paths, and multipart rectangles, ROD objects remain ROD objects after a split operation. When you split any ROD object, all alignments to that object are deleted.
Blockages	The *Split* command does not support blockage objects. ■ In the pre-selection mode, the command will fail, and an message will appear in the CIW. ■ In the post-selection mode, the command will fail to select the blockage object.
Boundaries	The *Split* command does not support P&R and snap boundary objects. ■ In the pre-selection mode, the command will fail, and an informational error will appear in the CIW. ■ In the post-selection mode, the command will fail to select the boundary object. For area and cluster boundary objects, the split command splits the objects in the same way it splits polygons, using points.

Splitting an Object

To split and stretch an object,

1. Select the object you want to split.
2. Choose *Edit – Advanced – Split* [`Control-s`]. You can open the Split Form by pressing F3.

© 2020 All Rights Reserved.

3. Click the points of the split line.

You can use the right mouse button to toggle between the split segments. In the above example the split will alternately highlight the path section interior to the split lines, and the two segments of the path exterior to the split lines.

When the split shape is drawn clockwise the split command selects the parts of the object outside the split lines first. When the split shape is drawn counterclockwise, the split command selects the parts of the object interior to the split lines first. In the case of a single split line, toggling with the right mouse button will alternate between the sides of the object divided by the split line.

© 2020 All Rights Reserved.

4. Click the reference point for the stretch.

Splitting a Path **Splitting a PathSeg**

5. Click the new location for the stretch.

Splitting a Path **Splitting a PathSeg**

© 2020 All Rights Reserved.

The following figures illustrate splitting and stretching of multiple segments of a pathseg at the same time:

Creating the split line

Clicking on the reference stretch area

© 2020 All Rights Reserved.

Stretching the split portions

Resultant of the split and stretch

Using Split to Change the Angle of Objects

As you stretch objects, you can change the angle of the segments.

To stretch path segments to any angle,

1. Choose *Edit – Advanced – Split* [`Control-s`]. Press F3 to open the form.

   The Split Form opens.

2. Turn *Lock Angles* off.

© 2020 All Rights Reserved.

3. Create a cut line.

4. Stretch the highlighted path segment to the angle you want.

   Note: In the Stretch Form, ensure that the *Snap Mode* is set to *anyAngle* or *diagonal* to change the angle of the segment.

Stretch the highlighted segment to the angle you want.

   The following figure shows the resultant stretched object.

You can add a jog to several paths (a bus) so that it wraps around an object by splitting a section of the paths and stretching the split section.

Splitting Multiple Objects

To stretch several paths,

1. Select all paths and connected objects you want to stretch.
2. Choose *Edit – Advanced – Split* [`Control-s`].
3. Press F3 to open the form.

   The Split Form opens.

© 2020 All Rights Reserved.

4. Set the *Snap Mode* to *anyAngle*.
5. Click to create a line through the segments.

   To stretch a segment of each path, create a split line that crosses each path twice, once at each end of the segment you want to stretch.

   Create the split line at a 45-degree angle through the segments, so that the edges of the segments are offset from each other. This prevents the path segments from overlapping when you stretch them.

6. Double-click the last point of the split line. The reference area to be stretched is highlighted.

7. Click a starting point for the stretch.

© 2020 All Rights Reserved.

8. Click where you want to stretch the segments.

The following figure shows the resultant stretched objects:

The path segments are stretched. Also, these other objects move:

- Any objects attached to the segments (such as vias) that have been selected
- Any objects not recognized by the *Split* command (instances, labels, mosaics, bends, and tapers) but attached to the path, if they have been selected
- Objects aligned to a multipart path you stretch stay in alignment, even if you did not select them

Subrectangles and subpaths in multipart paths regenerate after the stretch.

© 2020 All Rights Reserved.

Modifying Corners

The *Modify Corner* command enables you to reshape the corner of a polygon to make a rounded or chamfered (45-degree) edge.

Support for Objects	
Rectangles	The *Modify Corner* command supports rectangles and the resultant modified shape is a polygon.
ROD objects	The *Modify Corner* command supports ROD polygons and ROD rectangles, except for master rectangles that are part of a path objects.
Blockages	The *Modify Corner* command does not support blockage objects. ■ In the pre-selection mode, the command will fail, and an informational error will appear in the CIW. ■ In the post-selection mode, the command will fail to select the blockage object.
Boundaries	The *Modify Corner* command does not support boundary objects. ■ In the pre-selection mode, the command will fail, and an informational error will appear in the CIW. ■ In the post-selection mode, the command will fail to select the boundary object.

Modifying the Corners of a Polygon

To reshape a corner,

1. Choose *Edit – Advanced – Modify Corner*.

   The Modify Corner Form opens.

2. Do one of the following:

   - Select *radial* and set *Radius* (the radius of the curve) and *Number of Sides* (the number of segments to create the curve).
   - Select *chamfer* and set *Distance*.

Note: The maximum value you can set is half the length of the shortest adjacent line segment. The layout editor applies this value or the maximum allowable value for each corner, whichever is shorter.

© 2020 All Rights Reserved.

3. Click the corner you want to change. To change more than one corner, `Shift`-click each additional corner.
4. Click *OK*.

Original object | Object with a radial corner | Object with a chamfer corner

Sizing Objects

The *Size* command reduces or enlarges objects by stretching each edge in or out by the given number of user units.

Support for Objects	
ROD objects	The *Size* command supports ROD objects. However, when you size a multipart path (MPP), the master path becomes a ROD polygon and inherits its name from the original MPP; all subparts become ordinary, unnamed database shapes. When you size any ROD object, all alignments to that object are deleted.

To size objects,

1. Select the objects you want to resize.
2. Choose *Edit – Advanced – Size*.

   The Size Form opens.
3. Select the *Increase Size by Value* or *Increase Size by Direction* (*Left*, *Right*, *Top*, or *Bottom*) to specify the number of units you want to stretch.

   A positive number enlarges the object, a negative number reduces it.

© 2020 All Rights Reserved.

4. Click *OK*.

Object before resizing.

Object after using *Size* with a value of 1 to grow each edge by 1 unit.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Size` command does not support curved shapes.

Attaching and Detaching Objects

Attaching Objects

When you attach one object to another, you create a parent-child relationship between the two objects. The attached object is the child of the object to which it is attached. The behavior of parent-child relationships is described in the subsequent sections:

Move Command

- When the parent moves, all child objects move with it.
- When a child moves, the parent does not move.

Delete Command

- When the parent is deleted, all child objects are deleted.
- When a child is deleted, the parent is not deleted.

© 2020 All Rights Reserved.

Chop Command

- When the parent is chopped such that there is only one resultant shape, the parent-child relationship is retained.
- When the parent is chopped such that there are more than one resultant shapes, one of the chopped shapes becomes the parent of the child.
- When a child is chopped such that there is only one resultant shape, the parent-child relationship is retained.
- When a child is chopped such that there are more than one resultant shapes, all the shapes attach as child to the parent.

Yank Command

- When the child is selected and the parent is not selected, the child is yanked. After the `Paste` command, the new object is independent.
- When the parent is selected and the child is not selected, the parent is yanked. After the `Paste` command, the new object does not have any child.
- When the child and parent are selected, both the child and parent are yanked. After the `Paste` command, the parent-child relationship is retained in the new object.

Support for Objects	
Blockages	■ Cyclical relationships are not allowed. For example, if object A is a parent to object B, then object B cannot be a parent for object A. ■ An instance/cluster cannot be a child to blockage objects. ■ If a blockage object is a child object to an instance/cluster, during attach the ownership information will update, but the explicit parent/child relationship will not be retained. ■ If a blockage object is attached as a child to any object other than the instance/cluster, the explicit parent/child relationship will be retained. ■ A blockage object can be a parent to any other objects, such as text labels. ■ A blockage object cannot be owned by an instance and be a child to any other object. In this case the owner/blockage relationship takes precedence to the parent/child relationship.

© 2020 All Rights Reserved.

Boundaries	The *Attach/Detach* commands fully support boundary objects. When the parent is deleted, the child is deleted also.

To attach objects,

1. Choose *Edit – Advanced – Attach/Detach* [v].

   The following message appears in the Command Interpreter Window:

   `Select child object to be attached or detached.`

2. Click the object you want to be the child object. A dotted line extends from the child to the pointer.

3. Move the pointer to the object you want to be the parent object and click.

   The child is attached to the parent.

You can attach all objects to a parent object by pressing `Shift` and clicking the middle mouse button.

Detaching Objects

To detach objects that have been joined using the *Attach* command,

1. Select all of the child objects you want to detach.

2. Choose *Edit – Advanced – Attach/Detach* [v].

   The following message appears in the CIW:

   ```
   To attach child object, click the parent object;
   to detach child object, click in empty area.
   ```

   A dotted line extends from the child to the pointer.

© 2020 All Rights Reserved.

3. Move the pointer to an empty area of your cellview and click.

   The child is detached from the parent.

4. If you selected more than one child object, click again in an empty area.

You can detach all objects from the parent object by clicking the middle mouse button in an empty area.

© 2020 All Rights Reserved.

Converting Instances to Mosaics

The *Convert To Mosaic* command converts selected instances into mosaics. The command works on both physical and non-physical instances, fluid guard rings, and Pcell instances.

It is available in two modes, *Convert Instances into Mosaics* or *Convert Instances into Single Mosaic*. You can select the mode by pressing `F3` while the command is running or selecting *Convert To Mosaic* from the *Instance* context-sensitive menu to open the Instance To Mosaic form:

- **Convert Instances into Mosaics** is the default and converts a number of selected instances into an equivalent number of 1x1 mosaics.

Each mosaic inherits the name, orientation, and origin of the instance from which it was derived. Cluster and group memberships and parent-child relationships are retained, along with any halo blockages and text displays associated with the original instance.

Note: Connectivity information and any process rule overrides on the original instances are lost. Additionally, if the original instance was a member of any constraints, you must check and re-create these manually where necessary. *Convert to Mosaic* does not check or correct any constraints.

This option is also available on the *Edit Instance Properties* form. Click the *Convert to Mosaic* button on the *Edit Instance Properties* form to convert instances into

© 2020 All Rights Reserved.

mosaics. Also, if *Common* editing mode is on, multiple instances are converted into multiple mosaics.

- **Convert Instances into Single Mosaic** converts a number of selected instances of the same master into a single mosaic. For example, in the diagram below four instances of the same master, `A`, are selected and converted into a single 2x2 mosaic.

In this mode, the selected instances must all have the same master and must be placed at regular intervals in both X and Y directions (the intervals need not be the same in both directions). If the selected instances are Pcells, then all the parameter values must match.

The name of the new mosaic is generated automatically by the system and reported in the CIW. It has the same origin as the lower left instance in the original selected set.

Note: Parent-child relationships, properties, and text displays are **not** maintained in the resultant mosaic.

© 2020 All Rights Reserved.

If the selected set includes objects other than instances, these are left untouched when the command is complete.

Example 1

Example 2

To convert instances into mosaics:

1. From the layout window menu bar, choose *Edit – Convert – To Mosaic*.

   Note: You can also select the instances before you start the command.

2. Press `F3` to open the Instance To Mosaic options form.

3. Choose the *Mode*.

   Note: The default is controlled using the convertEachInstanceIntoMosaic environment variable.

© 2020
All Rights Reserved.

4. In the canvas, select the instances to be converted and press `Return`.

   The selected instances are converted into one or more mosaics depending on the mode selected.

Note: If the *Instance Editing Mode* in the Layout Editor Options form is set to *P&R Boundary* or *Snap Boundary*, when converting intances to mosaics, the *Delta X* and *Delta Y* is set from the P&R Bounday or Snap Boundary, and not from the BBox.

Converting Mosaic to Instances

The *Convert To Instance* command converts selected mosaic to instances.

Convert Mosaic into Instances converts the selected mosaic to instances as illustrated below.

Each instance inherits the name, orientation, and origin of the mosaic from which it was derived.

Converting Circles, Donuts, and Ellipses to Polygons

Circles, donuts, and ellipses are stored in the database as true conics with centerpoints and radii. You might need to convert conics to polygons before streaming out your database for chip production.

Support for Objects

© 2020 All Rights Reserved.

ROD objects	The *Convert To Polygon* command fully supports ROD objects, except for rectangles and multipart rectangles. The *Convert To Polygon* command has no affect on rectangles and multipart rectangles. When you convert a multipart path, the polygon resulting from the master path is a ROD object and takes the name of the multipart path. Subparts, if any, become unnamed, ordinary database shapes (paths and rectangles). Subparts are not converted to polygons. When you convert any ROD object (except rectangles and multipart rectangles) to a polygon, all alignments to the converted object are deleted.
Blockages	The *Convert to Polygon* command does not support blockage objects. ■ In the pre-selection mode, the command will fail, and an informational error will appear in the CIW. ■ In the post-selection mode, the command will fail to select the blockage object.
Boundaries	The *Convert to Polygon* command does not support boundary objects. ■ In the pre-selection mode, the command will fail, and an informational error will appear in the CIW. ■ In the post-selection mode, the command will fail to select the boundary object.

To convert a conic to a polygon:

1. Choose *Edit – Convert – To Polygon*.
2. Click the conic you want to convert.

   The conic is redrawn as a polygon, using many straight-line segments to create each curve.
3. Continue clicking on each conic you want to convert.
4. Press `Esc` to stop converting shapes to polygons.

   No conversion happens if you run the *Convert to Polygon* command on a rectangle or a polygon. A message appears in the CIW if you attempt to convert a rectangle or polygon to a polygon.

© 2020 All Rights Reserved.

Note: You can change the number of polygon segments used to create the converted conics by setting the *Conic Sides* field in the Layout Editor Options Form. The default number of segments is 20.

Setting Conic Sides

You can set the number of segments used when recreating conics when you convert them to polygons. The minimum number of conic sides is 3, and the maximum number of conic sides is 4000.

To set the number of conic sides,

1. Choose *Options – Editor* [`Shift-e`].

   The Layout Editor Options Form opens.

2. In the *Conic Sides* field, type a value.
3. Click *OK*.

Converting Shapes Pathsegs

The *Convert To PathSeg* command converts selected shapes, polygons, and paths to pathsegs.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Convert To PathSeg` command is not supported for curved shapes.

Converting Polygons to Paths

The *Convert - To Path* command converts polygons and rectangles into paths. This command works in both the pre-selection and post-selection modes. However, only supported objects, polygons or rectangles, are converted into paths.

The *Convert - To Path* command can convert the polygons to paths, if the following criteria is met:

- Polygons must have a consistent width to be converted to a path.
- The edges of polygons must be orthogonal or 45 degrees.
- The number of vertices of polygons must be even.
- Polygons with rounded ends cannot be converted to a path.

© 2020 All Rights Reserved.

The figure below shows examples of polygons that can be converted to a path.

The figure below shows examples of polygons that cannot be converted to paths.

The first polygon in the above figure, Z shape, cannot be converted to a path. This is because it does not have exact matching vertices on the inside and outside of the bends of the polygon. The inside of the bends has one vertex and the outside has two vertices. The second polygon, I shape, has rounded ends, so this can also not be converted to a path. The third polygon, mirrored C, has anyAngle segments, so this cannot be converted to a path.

To convert a polyon or rectangle to a path:

1. Choose *Edit – Convert – To Path*.
2. Click the polygon or rectangle you want to convert.

   The polygon or rectangle is converted to a path.
3. Continue clicking each polygon or rectangle you want to convert.
4. Press `Esc` to stop converting polygons or rectangles to paths.

© 2020 All Rights Reserved.

Note: Alternatively, you can right-click a selected polygon in the design area and click *Convert To Path* from the context-sensitive menu.

Note: You can partially select the point or edge of a polygon, which is then used as the starting point for deducing the path from a polygon or rectangle. This enables you to control the direction of path.

(ICADVM20.1 Virtuoso RF Solution Only) The `Convert To Path` command is not supported for curved shapes.

Changing Origin

For information about the *Edit – Advanced – Move Origin* command, see Changing the Origin of a Cellview.

Aligning Objects

You can align objects by using the *Edit – Quick Align* command or the *Edit – Advanced – Align* command. The two commands enable you to perform similar tasks but differ in their usage. You can perform the following tasks by using these commands:

- Align objects to a point
- Align objects to an edge
- Align objects to a layer
- Align objects to a target axis

Both the commands work in both pre and post-selection modes.

The *Quick Align* command enables you to quickly align objects, in a more interactive manner than the *Align* command. In comparison with the *Align* command, the *Quick Align* command provides the following additional features:

- Enables smart snapping of edges, centerlines, and points within the specified aperture
- Enables cyclic highlighting of the reference and target objects within the aperture distance from the pointer by using the `Spacebar` key
- Enables the alignment with respect to reference or target objects across the hierarchy
- Supports the spacing rules defined in the technology file
- Enables alignment by stretching partially selected selected objects

© 2020 All Rights Reserved.

The *Quick Align* command enables you to align an edge, a centerline, or a point to another edge, centerline, or point, whereas the *Align* command aligns an object bounding box to the bounding box of another object.

This topic explains how you can align objects:

- By Using the Quick Align Command
- By Using the Align Command

By Using the Quick Align Command

The *Quick Align* command enables you to use objects, instances, and groups as the reference set and also use them as the target to which the reference set is to be aligned. By using the *Quick Align* command, you can select the edges, centerlines, and points on objects in an instance or a group, which may be present at any level in the hierarchy.

Note: The reference and target object can be the same object in the *Quick Align* command.

© 2020 All Rights Reserved.

The *Quick Align* command supports design-rule-driven (DRD) *Enforce* mode. You can specify the DRD rules that need to be enforced when the *Quick Align* command is active, through the *DRD Options* form accessed from the *Options - DRD Edit* menu.

For more information about *DRD Options* form, see Design Rule Driven Form Descriptions in the *Virtuoso Design Rule Driven Editing User Guide*.

The following sections demonstrate the resultant of the *Quick Align* command in various scenarios. These sections illustrate the command use with pre-selection of objects. Similar results are obtained with post-selection of objects, except that in post-selection, the command remains active and the objects do not remain selected after the alignment.

© 2020 All Rights Reserved.

This topic covers the following:

- Highlighting when the Quick Align Command is Active
- Customizing the Quick Align Settings
- Aligning Edges and Centerlines to a Target Edge
- Aligning Points to a Target Point
- Aligning Objects to a Target Axis

Highlighting when the Quick Align Command is Active

The Quick Align snapping mechanism uses different object highlighting schemes to set the reference and select the target. You can use the environment variable, QAHoldTimeMilliSec to specify the time required to hold the pointer over an object. If the pointer remains stationary over an object for longer than the specified time, the potential points, edges, and centerlines in the object are highlighted.

Reference Highlighting

Reference highlighting scheme is dependent on the following factors:

- Selection mode (Full Select or Partial Select)
- Edge + point mode or edge only mode

Additionally, highlighting of reference is different after the reference is set.

The following figure shows pre-select highlighting when the selection mode is Full Select and edge + point is active.

Edge + Center Point *Vertex* *Nexus*

Note: Nexus is the midpoint of the given shape. In case of a polygon, nexus is the center of its bounding box.

© 2020 All Rights Reserved.

The following figure shows pre-select highlighting when the selection mode is Partial Select and edge + point is active.

Edge + Center Point *Vertex* *Nexus*

The following figure shows pre-select highlighting in the edge only mode.

Full Select *Partial Select*

The figure below shows highlighting after the reference is set for both Full and Partial selection modes. Highlighting is same for nexus in both cases.

Full Select Edge + Center Point *Partial Select Edge + Center Point* *Nexus* *Full Select Edge Only* *Partial Select Edge Only*

Note: The reference point is discarded when the `Backspace` key is pressed.

Target Object Highlighting

The figure below shows highlighting for the target.

© 2020 All Rights Reserved.

Edge Center *Edge* *Center Point* *Nexus* *Vertex*

Note: If quick align is invoked during EIP, then you can snap the target at a level higher than the current edit level.

Customizing the Quick Align Settings

1. Choose *Edit – Quick Align* or press `a`.
2. Press `F3`.

   The Quick Align Form opens.

3. Select the *Copy reference* check box to create a copy of the objects during alignment.
4. Select the *Snap Target* from the options, *Edges and Points*, *Edges*, or *Points*.

© 2020 All Rights Reserved.

5. Select the *No Spacing*, *Min Spacing*, or *User Spacing* option to set up the spacing to be used between the reference set and the target edge, centerline, or point. You can also access these options from the *Spacing* sub-menu on the Quick Align Context-Sensitive Menu.

   Note: The spacing of the reference object layer from the target determines the spacing of the other objects in the reference set from the target.

   The figures below show alignment results when the *No Spacing*, *Min Spacing*, and *User Spacing* options are selected while aligning the reference object to the target edge. The reference object layer is `Poly` and that of the target is `Oxide`. The `minSpacing` value specified in the technology file between the two layers is `0.3`.

No Spacing

Min Spacing

User Spacing = `1.0`

You can specify both positive and negative values in the *User Spacing* field. Some examples of resultants with different reference and target edges are depicted in the figures below.

© 2020 All Rights Reserved.

- Positive User Spacing

Reference Edge | Target Edge | Resultant

© 2020 All Rights Reserved.

- Negative User Spacing

6. You can select the *Snap to targets in reference layer* option on the Quick Align Context-Sensitive Menu.

The following example illustrates a scenario when the *Snap to targets in reference layer* option is selected. The reference set comprises objects on `Metal1`, `Metal2`, and `Metal3` layers. The target objects exist on `Metal1` and `Metal2` layers. If the reference edge is on the `Metal1` object, edges, centerlines, and points on only the `Metal1` target object can be selected for aligning the reference set. Similarly, if the reference edge is on

© 2020 All Rights Reserved.

a `Metal2` object, edges, centerlines, and points on only the `Metal2` target object can be selected for aligning the reference set.

7. Select the *Snap to edges only* option to highlight only the edges and centerlines in the reference set and target object.

8. To align the reference objects to a target axis, select *Vertical* or *Horizontal* from the *Target Axis* sub-menu on the Quick Align Context-Sensitive Menu and click in the design display area.

© 2020 All Rights Reserved.

The target axis is created at the point of click.

Note: You can create the target axis, horizontal or vertical, before or after creating a selection set or setting a reference point.

a. Specify a value in the *Target Axis X= or Y=* field in the Quick Align Form.

Note: You can press Bindkey A + H to display horizontal target axis and A + V to display the vertical target axis.

The target axis is created at the specified distance from the X or Y-axis. In the figure below, a value of 2 is specified in the *Target Axis X= or Y=* field.

9. In the *Snap Target Visualization* section, set the *Width*, *Color*, and *Line Style* of the highlight edge and point.

© 2020 All Rights Reserved.

Some samples are shown below.

Width = 0, Color = yellow, Line Style = thickline
This is the default setting.

Width = 1, Color = red, Line Style = dashed

Width = 2, Color = magenta, Line Style = solid

Note: The following figures show some the points, edges, and centerline highlights in a

© 2020 All Rights Reserved.

path.

Highlighted Edge and Points

Highlighted Edges Only

Highlighted Centerlines

Use the highlighted points and edges to select the required reference and target edges, centerlines, or points.

Note: When snapping labels in the `Quick Align` command, snapping happens only to the origin of the labels.

Aligning Edges and Centerlines to a Target Edge

- In Full Selection Mode
- In Partial Selection Mode
- Setting the Reference and Target on the Same Edge

© 2020 All Rights Reserved.

You can also align edges and centerlines to a target point.

In Full Selection Mode

To align edges of objects in a reference set in full selection mode to a target edge or centerline:

1. On the *Options* toolbar, ensure that the *Full Select* button is on. If not, press `F4` to enable it.
2. Select the reference set objects.

3. Select *Edit – Quick Align* or press `a` to start the *Quick Align* command.
4. Press `F3` or right-click and select *Options* from the Quick Align Context-Sensitive Menu.

   The *Quick Align* form appears.
5. Set up the fields and options as required.
6. In the design display area, move the pointer around the selected reference set.

   The potential points, edges, and centerlines in the objects in the reference set are highlighted. By default, points, edges, and centerlines highlight. To highlight only the

© 2020 All Rights Reserved.

edges and centerlines, right-click and select *Snap to edges only* from the Quick Align Context-Sensitive Menu.

7. Click to set a highlighted edge as the reference edge.

Note: The reference edge should be part of the selected reference set.

8. Move the pointer close to the target object.

© 2020 All Rights Reserved.

The potential edges and centerlines parallel to the reference edge highlight. The figures below show the potential edges and centerline highlighted when the pointer is moved close to the target object.

Potential Target Edge 1 Highlighted

Potential Centerline Highlighted

Potential Target Edge 2 Highlighted

9. Do one of the following to select a highlighted target edge:

   a. Click.

© 2020 All Rights Reserved.

The object with the reference edge moves to align with the target edge. All the objects in the reference set move by the same distance. The relative spacing amongst the reference set objects is maintained.

The figure below shows the alignment of the reference set when the potential target edge 1, as shown in step 8, is set as the target edge.

Click to Select the Target Edge

The pre-selected reference set objects remain selected after the alignment.

b. Double-click.

The object with the reference edge moves to align with the target edge. All the objects in the reference set move by varying distances to align to the target edge, as the reference edge. The relative spacing amongst the reference set objects is lost.

The figure below shows the alignment of the reference set when the potential target edge 1, as shown in step 8, is set as the target edge.

Double-Click to Select the Target Edge

The pre-selected reference set objects remain selected.

© 2020 All Rights Reserved.

c. Double-click at a point when no target edge is highlighted.

The object with the reference edge moves to align to the point. All the objects in the reference set move by varying distances to align to the target point, as the reference edge. The relative spacing amongst the reference set objects is not maintained.

The figure below shows the alignment of the reference set to the point.

Double-Click for Aligning to a Point

The pre-selected reference set objects remain selected.

If you quick align an edge of a wire (pathseg) in full selection mode to a target edge, the wire moves to align the selected edge to the target edge, as shown below.

In Partial Selection Mode

To align edges of objects in a reference set in partial selection mode to a target edge or centerline:

1. On the *Options* toolbar, ensure that the *Partial Select* button is on. If not, press `F4` to enable it.

© 2020 All Rights Reserved.

2. Partially select the set of reference objects. Use `Shift` + click to add to the selection and `Ctrl` + click to remove from the selection.

3. Select *Edit – Quick Align* or press `a` to start the *Quick Align* command.
4. Press `F3` or right-click and select *Options* from the Quick Align Context-Sensitive Menu.

   The *Quick Align* form appears.
5. Set up the fields and options as required.
6. In the design display area, move the pointer around the partially selected reference set.

   The partially selected objects determine the alignment axis. The potential points, edges, and centerlines, along one axis, in the objects in the reference set highlight. By default, points, edges, and centerlines highlight. To highlight only the edges and centerlines, right-click and select *Snap to edges only* from the Quick Align Context-Sensitive Menu.
7. Click to set a highlighted edge as the reference edge.

   Note: The reference edge should be part of the selected reference set.
8. Move the pointer close to the target object.

© 2020 All Rights Reserved.

The potential edges and centerlines parallel to the reference edge highlight. The figures below show the potential edges and centerline highlighted when the pointer is moved around the target object.

Potential Target Edge 1 Highlighted

Potential Centerline Highlighted

Potential Target Edge 2 Highlighted

9. Do one of the following to select the highlighted target edge:

   a. Click.

      The object with the reference edge stretches to align with the target edge. All the objects in the reference set stretch by the same amount. The reference edge determines the stretch direction. The relative spacing amongst the reference set objects is maintained.

© 2020 All Rights Reserved.

The figure below shows the alignment of the reference set when the potential target centerline, as shown in step 8, is set as the target.

Click to Select the Target Centerline

The pre-selected reference set objects remain selected after the alignment.

b. Double-click.

The object with the reference edge stretches to align with the target edge. All the objects in the reference set stretch by varying amounts to align to the target edge, as the reference edge. The reference edge determines the stretch direction. The relative spacing amongst the reference set objects is lost.

The figure below shows the alignment of the reference set when the potential target centerline, as shown in step 8, is set as the target.

Double-Click to Select the Target Centerline

The pre-selected reference set objects remain selected after the alignment.

c. Double-click at a point when no target edge is highlighted.

The object with the reference edge stretches to align to the point. All the objects in the reference set stretch by varying amounts to align to the target point, as the

© 2020 All Rights Reserved.

reference edge. The reference edge determines the stretch direction. The relative spacing amongst the reference set objects is not maintained.

The figure below shows the alignment of the reference set to the point.

The pre-selected reference set objects remain selected after the alignment.

If you quick align an edge of a wire (pathseg) in partial selection mode to a target edge, the selected edge moves to align to the target edge and the remaining pathsegs of the wire stretch to move along with the aligned pathseg, as shown below.

Setting the Reference and Target on the Same Edge

In full selection mode, if you set an edge of an object as the reference and then set the same edge as the target, the object moves based on the value specified in the *User Spacing* field. If you specify a positive value, the object moves in the positive direction. If you specify a

© 2020 All Rights Reserved.

negative value, the object moves in the negative direction. If *User Spacing* is `0`, no change is made to the object.

In partial selection mode, if you set an edge of an object as the reference and then set the same edge as the target, the selected edge stretches based on the value specified in the *User Spacing* field. If you specify a positive value, the edge is stretched in the positive

© 2020 All Rights Reserved.

direction. If you specify a negative value, the edge is stretched in the negative direction. If *User Spacing* is `0`, no change is made to the object.

Aligning Points to a Target Point

- In Full Selection Mode
- In Partial Selection Mode

You can also align points to target edges and centerlines.

In Full Selection Mode

To align points of objects in a reference set in full selection mode to a target point:

© 2020 All Rights Reserved.

1. On the *Options* toolbar, ensure that the *Full Select* button is on. If not, press `F4` to enable it.
2. Select the reference set objects.

3. Select *Edit – Quick Align* or press `a` to start the *Quick Align* command.
4. Press `F3` or right-click and select *Options* from the Quick Align Context-Sensitive Menu.

   The *Quick Align* form appears.
5. Set up the fields and options as required.
6. In the design display area, move the pointer around the selected reference set.

   The potential points, edges, and centerlines in the objects in the reference set highlight. By default, points, edges, and centerlines highlight. Ensure that the *Snap to points and edges* option is selected from the Quick Align Context-Sensitive Menu.
7. Click to select a reference point.

   Notice that the selected reference point is marked by a diamond shape.
8. Move the pointer close to the target object.

© 2020 All Rights Reserved.

The potential points and edges in the target object appear.

9. Do one of the following to select a highlighted target point:

a. Click.

The object with the reference point moves to align with the selected target point. All the objects in the reference set move by the same distance. The relative spacing amongst the reference set objects is maintained.

The pre-selected reference set objects remain selected after the alignment.

b. Double-click.

The object with the reference point moves to align with the selected target point. All the objects in the reference set move to align to the reference and target point. The relative spacing amongst the reference set objects is lost.

Note: The points on the objects in the reference set that align to the reference and target points are relative to the selected reference point.

© 2020 All Rights Reserved.

c. Double-click at a point when no target point is highlighted.

The object with the reference point moves to align with the point. All the objects in the reference set move to align to the reference and double-click point. The relative spacing amongst the reference set objects is lost.

Note: The points on the objects in the reference set that align to the reference and target points are relative to the selected reference point.

The pre-selected reference set objects remain selected after the alignment.

In Partial Selection Mode

To align points of objects in a reference set in partial selection mode to a target point:

1. On the *Options* toolbar, ensure that the *Partial Select* button is on. If not, press `F4` to enable it.
2. Partially select the set of reference objects. Use `Shift` + click to add to the selection and `Ctrl` + click to remove from the selection.

© 2020 All Rights Reserved.

3. Select *Edit – Quick Align* or press `a` to start the *Quick Align* command.
4. Press `F3` or right-click and select *Options* from the Quick Align Context-Sensitive Menu.

   The *Quick Align* form appears.
5. Set up the fields and options as required.
6. In the design display area, move the pointer around the partially selected reference set.

   The partially selected objects determine the alignment axis. The potential points, edges, and centerlines in the objects in the reference set along one axis highlight. By default, points, edges, and centerlines highlight. Ensure that the *Snap to points and edges* option is selected from the Quick Align Context-Sensitive Menu.
7. Click to select a reference point.

8. Move the pointer close to the target object.

   The potential points and edges in the target object highlight. Notice that the selected reference point is marked by a diamond shape.
9. Do one of the following to select a highlighted target point:

   a. Click.

© 2020 All Rights Reserved.

The object with the reference point stretches to align with the selected target point. All the objects in the reference set stretch by the same amount. The relative spacing amongst the reference set objects is maintained.

The pre-selected reference set objects remain selected after the alignment.

b. Double-click to select the target point.

The object with the reference point stretches to align with the target point. All the objects in the reference set stretch by varying amounts to align to the target point. The relative spacing amongst the reference set objects is not maintained.

© 2020 All Rights Reserved.

The pre-selected reference set objects remain selected after the alignment.

Aligning Objects to a Target Axis

The support for aligning objects in a reference set to a target axis is available only in post-selection mode.

- In Full Selection Mode
- In Partial Selection Mode

In Full Selection Mode

To align objects in a reference set in full selection mode to a target axis:

1. On the *Options* toolbar, ensure that the *Full Select* button is on. If not, press `F4` to enable it.
2. Select *Edit – Quick Align* or press `a` to start the *Quick Align* command.
3. Right-click and select *Vertical* or *Horizontal* from the *Target Axis* sub-menu on the Quick Align Context-Sensitive Menu. Alternatively, press `v` or `h` to set the target axis.
4. In the design display area, click to place the target axis.
5. Select the reference set objects.

6. In the design display area, move the pointer around the selected reference set.

   The potential points, edges, and centerlines in the objects in the reference set highlight. By default, points, edges, and centerlines highlight. To highlight only the edges and centerlines, right-click and select *Snap to edges only* from the Quick Align Context-Sensitive Menu.

7. Do one of the following to align the reference set with the target axis:

   a. Click to select a reference edge, centerline, or point.

© 2020 All Rights Reserved.

The object with the reference edge moves to align with the target axis. All the objects in the reference set move by the same distance. The relative spacing amongst the reference set objects is maintained.

The reference edge aligns with the target axis. The relative spacing amongst the reference set objects is maintained.
The post-selected reference set objects are de-selected after the alignment.

b. Double-click to select a reference edge, centerline, or point.

The object with the reference edge moves to align with the target axis. All the objects in the reference set move by varying distances to align to the target axis, as the reference edge. The relative spacing amongst the reference set objects is lost.

The reference edge aligns with the target axis. The relative spacing amongst the reference set objects is lost.
The post-selected reference set objects are de-selected after the alignment.

© 2020 All Rights Reserved.

In Partial Selection Mode

To align objects in a reference set in partial selection mode:

1. On the *Options* toolbar, ensure that the *Partial Select* button is on. If not, press `F4` to enable it.
2. Select *Edit – Quick Align* or press `a` to start the *Quick Align* command.
3. Right-click and select *Vertical* or *Horizontal* from the *Target Axis* sub-menu on the Quick Align Context-Sensitive Menu. Alternatively, press `v` or `h` to set the target axis.
4. In the design window, click to place the target axis.

   To place the target axis at a specified distance from the X or Y-axis, specify the value in the *Target Axis X= or Y=* field in the *Quick Align* form.
5. Partially select the set of reference objects. Use `Shift` + click to add to the selection and `Ctrl` + click to remove from the selection.

6. In the design display area, move the pointer around the selected reference set.

   The potential points, edges, and centerlines in the objects in the reference set highlight. By default, points, edges, and centerlines highlight. To highlight only the edges and centerlines, right-click and select *Snap to edges only* from the Quick Align Context-Sensitive Menu.
7. Do one of the following to align the reference set with the target axis:

   a. Click to select a reference edge, centerline, or point.

© 2020 All Rights Reserved.

The object with the reference edge stretches to align with the target axis. All the objects in the reference set stretch by the same amount. The relative spacing amongst the reference set objects is maintained.

The reference edge aligns with the target axis.

The post-selected reference set objects are de-selected after the alignment.

b. Double-click to select a reference edge, centerline, or point.

The object with the reference edge stretches to align with the target axis. All the objects in the reference set stretch by varying amounts to align with the target axis, as the reference edge. The relative spacing amongst the reference set objects is lost.

The reference edge aligns with the target axis.

© 2020 All Rights Reserved.

The post-selected reference set objects are de-selected after the alignment.

© 2020 All Rights Reserved.

The table below summarizes the behavior of the *Quick Align* command in full and partial selection modes. The behavior is the same for pre and post-selected objects.

Action	Result in Full Selection Mode	Result in Partial Selection Mode
Aligning Edges and Centerlines		
Click to select the target edge	All the objects in the reference set “move” and align to the target edge. The relative spacing amongst the reference set objects is maintained.	All the objects in the reference set “stretch” and align to the target edge. The relative spacing amongst the reference set objects is maintained.
Double-click to select the target edge	All the objects in the reference set “move” and align to the target edge. The relative spacing amongst the reference set objects is lost.	All the objects in the reference set “stretch” and align to the target edge. The relative spacing amongst the reference set objects is lost.
Aligning Points		
Click to select the target point	All the objects in the reference set “move”. The reference point coincides with the target point. The relative spacing amongst the reference set objects is maintained.	All the objects in the reference set “stretch” and align to the target point. The reference point coincides with the target point. The relative spacing amongst the reference set objects is maintained.
Double-click to select the target point	All the objects in the reference set “move”. Points on all the objects in the reference set coincide with the target point. The relative spacing amongst the reference set objects is lost.	All the objects in the reference set “stretch” and align to the target point. Points on all the objects in the reference set coincide with the target point. The relative spacing amongst the reference set objects is lost.

By Using the Align Command

The *Align* command lets you align objects or shapes in a cellview. You can align objects to either a point, another object, or a layer. You can also align a layer to another layer or other objects. When aligning objects in a chain, the entire chain gets aligned with the selected object. You can add objects to an already aligned group.

© 2020 All Rights Reserved.

This topic covers the following:

- Aligning Objects to a Point
- Aligning Objects to an Object
- Aligning Objects to a Layer
- Aligning Layer to Layer
- Aligning a Layer to Objects
- Adding Objects to a Pre-Selected Aligned Group
- Aligning Objects by using the Align Toolbar

Caution

The Align command does not utilize spacing rules from the technology file and does not obey user-defined constraints with the exception of group constraints. Rule and constraint violations can occur as a result.

If nothing is selected in the cellview, the *Align* command prompts you to select one or more objects. You can area select or use the *Shift* + left mouse button to select additional objects. Use the *Return* key or double-click the mouse button to complete the command. To specify a different reference, click the *Set New Reference* button in the Align Form. The layout editor prompts you to select a new reference to which you want to align objects. You can exit the command by clicking the *Cancel* button.

Tip

You can quickly align an object by placing the pointer over the object and then pressing the *Return* key or double clicking. Conversely, to prevent an object from aligning when you press the *Return* key or double-click, ensure that the pointer points at empty space in the design canvas. You can also use the *Return* key or double-click for aligning multiple selected objects.

Aligning Objects to a Point

You must always choose a reference point. The selected shapes align to the selected reference point as per the other options specified in the *Align* form. If you specify a spacing value for point alignment, the first object always overlaps the reference point and the spacing value is used to align the remaining objects. After alignment, all selected shapes are de-selected.

© 2020 All Rights Reserved.

1. Select *Edit – Advanced – Align*.

   The Align Form opens.

2. Choose the *Align Direction*.

   If you make an error in the *Reference* section of the form, click the *Set New Reference* button, and then correct your entries.

3. Choose the *Point* option.

   You are prompted to "`Point at a reference point to align to`:". Click the point with your pointer or enter the coordinates of the point in the CIW. After you have chosen a point and an alignment direction, a pointer line appears indicating the beginning point of the alignment and the alignment direction. In the example below, the direction is horizontal.

Pointer indicating beginning point and alignment direction.

4. In the *Align Using* section of the form, choose *Object*. In the example shown below, the objects are instances, so the *Object* radio button is chosen.

5. Choose the edge of the object you want to align from the *Use* cyclic field. In the example, *Lower Left* is chosen in the *Use* cyclic field.

© 2020 All Rights Reserved.

6. To evenly space the objects in the *Align Direction*, click the *Apply Spacing* check box and enter the value in user units by which you want to separate the objects. If you do not use *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced, as shown in the example below:

7. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.

8. Choose *Between Objects* or *Pitches*, depending on the spacing you want. In the example, *Between Objects* is chosen, with a spacing of two user units.

9. Choose how you want the objects sorted in the *Sort Objects By* cyclic field.

   You are prompted to "`Select object(s) to be aligned:`". You can select the objects by clicking each object one at a time, while holding down the `Shift` key, or you can select the objects by drawing a selection box around them.

10. Click *Apply*.

    In the example, the reference point is -157:-12 and the *Align Direction* is Horizontal. The *Align Using* is Instance, the *Use* cyclic is set to *Lower Left*. *Apply Spacing* is set to `2.0` user units *Between Objects*, and the objects are sorted by *Instance Name*. The result is shown below.

© 2020 All Rights Reserved.

If *Orthogonal Spacing* is also selected and set to 2.0 user units, objects are aligned along the Y-axis as well.

Aligning Objects to an Object

The align command supports top-level pins, shapes, instances, devices, rows, cluster boundaries, and groups. Area halos, layer halos, boundaries and blockages, if selected as part of a set of objects, are ignored by the align command.

1. Select *Edit – Advanced – Align*.

   The Align Form opens.

2. Choose the *Align Direction* in the Reference section of the form.

3. Choose the *Object* radio button in the Reference section of the form.

   You are prompted to "`Select a reference object to align to`"

   If you make an error in the reference section of the form, click the *Set New Reference* button, then correct your entries.

4. Select the type of object you want to align to. In the example below, *Group* is chosen.

© 2020 All Rights Reserved.

5. Click the object in the cellview window you want to align other objects to.

   The alignment pointer appears.

6. In the *Align Using* section of the form choose *Object*.
7. Choose the type of object you are aligning to the reference object. In the example the objects are shapes, so the *Any Shape* cyclic field is chosen.
8. Choose the edge of the object you want to align from the *Use* cyclic field. In the example, *Left Edge* is chosen.
9. To evenly space the objects in the *Align Direction*, click the *Apply Spacing* selection box and enter the value in user units in the amount by which you want to separate the

© 2020 All Rights Reserved.

objects. If you do not use *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced.

10. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.

11. Choose Between *Objects* or *Pitches*, depending upon the spacing you want. In the example, *Between Objects* is chosen, with a spacing of two user units.

12. Choose how you want the objects sorted in the *Sort Objects By* cyclic field.

You are prompted to "`Select object(s) to be aligned`". You can select the objects by clicking on each object one at a time, while holding down the Shift key, or you can select the objects by drawing a selection box around them.

13. Click *Apply*.

In the example the reference object is a group, the *Align Direction* is Vertical, and the *Use* field is set to Left Edge. The *Align Using* is Any Shape, the *Use* cyclic is set to Left

© 2020 All Rights Reserved.

Edge. *Apply Spacing* is set to 2.0 user units between objects, and the objects are sorted by Order Selected. The result is shown below.

© 2020 All Rights Reserved.

If *Orthogonal Spacing* is also selected and set to 2.0 user units, objects are aligned along the X-axis as well.

Aligning Objects to a Layer

The align command supports top-level pins, shapes, instances, devices, rows, cluster boundaries, and groups. Area halos, layer halos, boundaries and blockages, if selected as part of a set of objects, are ignored by the align command.

1. Select *Edit – Advanced – Align*.

   The Align Form opens.

2. Choose the *Align Direction* in the *Reference* section of the form.
3. Choose the *Object* radio button in the *Reference* section of the form.

© 2020 All Rights Reserved.

You are prompted to "`Select a reference object to align to`"

If you make an error in the *Reference* section of the form, click the Set New Reference button, then correct your entries.

4. Select the type of object you want to align to. In the example below, Instance is chosen.
5. Click the object in the cellview window you want to align other objects or layers to.

   The alignment pointer appears.

6. In the *Align Using* section of the form, choose *Layer*.
7. Choose the layer you are aligning to the reference object. In the example, the `metal1 drawing` layer is the alignment layer.
8. Choose the edge of the object or layer you want to align from the *Use* cyclic field. In the example, *Top Edge* is chosen in the *Use* cyclic field.
9. To evenly space the objects in the *Align Direction*, click the *Apply Spacing* selection box and enter the value in user units in the amount by which you want to separate the objects. If you do not use the *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced.
10. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.
11. Choose *Between Objects* or *Pitches*, depending upon the spacing you want. In the example, *Between Objects* is chosen, with a spacing of `5.0` user units.
12. Choose how you want the objects sorted in the *Sort Objects By* cyclic field.

© 2020 All Rights Reserved.

You are prompted to "`Select object(s) to be aligned`". You can select the objects by clicking on each object one at a time, while holding down the Shift key, or you can select the objects by drawing a selection box around them.

13. Click *Apply*.

In the example the reference object is an instance, the *Align Direction* is Horizontal, and the *Use* field is set to Top Edge. The *Align Using* is Layer `metal1 dg`, the *Use* cyclic is set to `Top Edge`. The *Apply Spacing* is selected, with 5.0 user units *Between Objects*. Objects are sorted by Order Selected.

Aligning Layer to Layer

1. Select *Edit – Advanced – Align*.

The Align Form opens.

2. Choose the *Align Direction* in the *Reference* section of the form. In the example, the Vertical Align Direction is used.

3. Choose the *Layer* radio button in the *Reference* section of the form.

4. Choose the layer you want to use to align the layer to from the cyclic field.

In the example below, `poly1 dg` is chosen.

5. Choose the layer edge you want the objects to align to.

In the example, *Left Edge* is chosen in the *Use* cyclic field.

You are prompted to "`Select a reference object to align to`"

If you make an error in the *Reference* section of the form, click the *Set New Reference* button, then correct your entries.

6. Click the object in the cellview window you want to align other objects or layers to.

The alignment pointer appears.

© 2020 All Rights Reserved.

7. In the *Align Using* section of the form choose Layer.
8. Choose the layer you want to align to the reference layer.

   In the example below, `poly1 dg` is chosen from the cyclic field.
9. Choose the edge of the object or layer you want to align from the *Use* cyclic field. In the example, *Left Edge* is chosen in the *Use* cyclic field.
10. To evenly space the objects in the *Align Direction*, click the *Apply Spacing* selection box and enter the value in user units in the amount by which you want to separate the objects. If you do not use the *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced.
11. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.
12. Choose how you want the objects sorted in the *Sort Objects By* cyclic field. In the example, *Instance Name* is used.

© 2020 All Rights Reserved.

You are prompted to "`Select object(s) to be aligned`". You can select the objects by clicking on each object one at a time, while holding down the Shift key, or you can select the objects by drawing a selection box around them.

13. Click *Apply*.

In the example the reference layer is Layer of material `poly1 dg`, the *Align Direction* is Vertical, and the *Use* field is set to Left Edge. The *Align Using* is Layer, the *Use* cyclic is set to Left Edge. The *Apply Spacing* is selected, with 5.0 user units *Between Objects*. Objects are sorted by *Instance Name*.

Aligning a Layer to Objects

1. Select *Edit – Advanced – Align*.

The Align Form opens.

2. Choose the *Align Direction* in the *Reference* section of the form. In the example, the Horizontal Align Direction is used.

3. Choose the *Layer* option in the *Reference* section of the form.

You are prompted to "`Select a reference object to align to`"

If you make an error in the *Reference* section of the form, click the *Set New Reference* button, then correct your entries.

4. Choose the layer edge you want the objects to align to. In the example, *Top Edge* is used.

© 2020 All Rights Reserved.

5. Select the object containing the layer you want to align to. In the example below, `pdiff dg` is chosen in the *Use* cyclic field.
6. Click the object in the cellview window you want to align other objects or layers to.

   In the example below, `pdiff dg` is chosen. The alignment pointer appears.

7. In the *Align Using* section of the form, choose *Object*.
8. Choose the objects you are aligning to the reference layer.
9. Choose the edge of the object or layer you want to align from the *Use* cyclic field. In the example, *Top Edge* is chosen in the *Use* cyclic field.
10. To evenly space the objects in the *Align Direction*, click the *Apply Spacing* check box and enter the value in user units in the amount by which you want to separate the objects. If you do not use the *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced.
11. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.

© 2020 All Rights Reserved.

12. Choose how you want the objects sorted in the *Sort Objects By* cyclic field. In the example, *Order Selected* is used.

You are prompted to "`Select object(s) to be aligned`". You can select the objects by clicking on each object one at a time, while holding down the *Shift* key, or you can select the objects by drawing a selection box around them.

13. Click *Apply*.

In the example the reference layer is an `pdiff dg`, the *Align Direction* is Horizontal, and the *Use* field is set to Top Edge. The *Align Using* is Instance, the *Use* cyclic is set to Top Edge. The *Apply Spacing* is selected, with 5.0 user units *Between Objects*. Objects are sorted by *Order Selected*.

Adding Objects to a Pre-Selected Aligned Group

To add other objects to the aligned group,

1. Use the `Shift` + click to select the unaligned objects.
2. In the Align form, click *Set New Reference*.

   The prompt area of the cellview window asks you to:

   `Select a reference object to align to:`

3. Click in the layout window at the point to which you want the objects to align.

   The unaligned device(s) align(s) with the original selected alignment group.

   Before:

© 2020 All Rights Reserved.

After:

Aligning Objects by using the Align Toolbar

For performing quick alignment operations by using the *Align* command, use the *Window - Toolbars - Align* toolbar. Using the toolbar, you can perform left, right, top, and bottom alignment of the selected objects. You can also align them vertically or horizontally. The toolbar also provides the option of specifying alignment and orthogonal spacing values.

Tip

To specify more advanced align options, such as the reference point, object, or layer, the align direction, and the sort order, use the *Edit – Advanced – Align* form.

To align the left, right, top, or bottom edges of the selected objects,

© 2020 All Rights Reserved.

1. Select the objects to be aligned.

2. Click *Align Left*, *Align Right*, *Align Top*, or *Align Bottom* on the toolbar for the required alignment. The selected objects are aligned to the left, right, top, or bottom edge of the imaginary bounding box around the selected set.

3. To evenly space the objects, select *Align Spacing* and enter the value in user units of the amount by which you want to separate the objects. If you do not use the *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced.

4. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.

5. To align objects of a specific reference layer, select the *Use Reference Layer* option and select a reference layer.

© 2020 All Rights Reserved.

In the following example, the selected objects have been aligned first to the left and then to the top. The objects have not been spaced.

Result of Using *Align Left* from the Toolbar, *Align Spacing* = 0

Result of Using *Align Top* from the Toolbar, *Align Spacing* = 0

© 2020 All Rights Reserved.

Alignment in the following examples uses spacing.

Result of Using *Align Right* from the Toolbar,
Align Spacing = 1.0

Result of Using *Align Bottom* from the Toolbar,
Align Spacing = 1.0,
Orthogonal Spacing = 1.0

To align the objects horizontally or vertically,

© 2020 All Rights Reserved.

1. Select the objects to be aligned.

2. Click *Align Vertical* or *Align Horizontal* on the toolbar as required.
3. To evenly space the objects, select *Align Spacing* and enter the value in user units of the amount by which you want to separate the objects. If you do not use the *Apply Spacing*, the objects retain their original spacing. They are aligned, but not evenly spaced.
4. To space the objects perpendicular to the *Align Direction* as well, select the *Orthogonal Spacing* option. This option is available only if *Apply Spacing* is selected.

© 2020 All Rights Reserved.

If *Align Spacing* is not on, the centre of the imaginary bounding box around the selected set is considered as the reference point. Objects are then aligned along the imaginary vertical or horizontal line passing through this center point.

Result of Using *Align Vertical* from the Toolbar,
Align Spacing = 0

Result of Using *Align Horizontal* from the Toolbar,
Align Spacing = 0

If *Align Spacing* is on, the object closest to the centre of the imaginary bounding box around the selected set is regarded as the reference object. Remaining objects are aligned vertically or horizontally with respect to the reference object, separated by *Align Spacing* distance and *Orthogonal Spacing*, if specified.

Video

For details on how to use the smart snapping of ruler in the quick align command, see the video *How to Use Smart Snapping of Ruler in Quick Align Command*.

© 2020 All Rights Reserved.

Editing Slots

For information about adding slots to existing rectangles or paths, see Creating Slots. For information about adding slots while creating rectangles or paths, see Creating Rectangles and Creating Paths, respectively.

Note: The *Edit Slot* command applies only to slotted shapes that are created by using the *Create Slot*, *Create Rectangle*, or *Create Path* commands.

To edit slots in an existing rectangle or path:

1. Click the rectangle or path containing the slots. You can select multiple shapes containing slots to simultaneously edit the slots in all of them. The selected shapes can be a mix of ROD objects and polygons.
2. Choose *Edit – Advanced – Slot*.

   The Edit Slot Form opens and is populated with the values of the slots in the selected shapes.

   Alternatively, you can first open the form, and then select one or more shapes for editing the slots they contain.

3. Update the values in the different fields as required. In addition, from the *Shape Type* list, select *Preserve*, *Convert to Polygon*, or *Convert to ROD* based on the required resultant shape type.
4. Click *Apply*.

   The values specified in the form are applied to the slots in the selected shapes.

© 2020 All Rights Reserved.

5. Press *Esc* or click *Cancel* in the form to finish editing the slots.

Hierarchical Editing

For information about hierarchical editing commands, see Chapter 5, "Working With Hierarchical Designs":

- *Descend Edit* and *Descend Read*: see Descending into a Cellview
- *Return* and *Return To Level*: see Descending into a Cellview and Returning to a Different Hierarchy Level
- *Edit in Place*: see Editing a Cell in Place
- *Tree*: see Using the Tree Form
- *Make Editable*: see Making Hierarchy Editable
- *Refresh*: see Refreshing Memory after Editing
- *Make Cell*: see Making Cells
- *Flatten*: see Flattening Instances

Editing Groups

For information about how to create a group, see Creating Groups.

You can perform the following edit operations on a group:

- Adding Objects to a Group
- Adding Objects to a Group By Using the Select Target Group Form
- Ungrouping All Objects in a Group
- Removing Objects from a Group
- Editing Group Contents

Adding Objects to a Group

To add an object to a group:

1. Choose *Edit – Group – Add to Group*.

© 2020 All Rights Reserved.

You are prompted to "`Point at the target group`:"

2. Click the group you want to edit.

   Alternatively, you can select the group you want to edit and right-click to display the *Group* context menu. Select *Add to Group* from the context menu.

   You are prompted to "`Select the figure(s) to add to group:`".

3. Click the objects you want to add to the group.

   The objects are added to the group. The solid line surrounding the group changes to include the new object or objects.

4. Press the `Esc` key.

5. The dotted line surrounds all the objects when you move your pointer over the group.

Adding Objects to a Group By Using the Select Target Group Form

To add an object to a group using the *Select Target Group* form, follow these steps:

1. Choose *Edit – Group – Add to Group*.

2. Press the `F3` key.

   The *Select Target Group* form appears.

© 2020 All Rights Reserved.

3. Enter the name of the group in the *Group Name* field.
4. Click the *Filter* button and the group name is added to the information box.

   You can continue to add groups to the information box by adding the group name, or you can click the Filter button to add all of the groups into the information box.
5. Click the name of the first group you want to zoom to.
6. Click the Zoom To group radio button.
7. Move the scale slider to zoom in closer or zoom out from the group.

   The scale set at 1 is the equivalent of the fit edit command. The scale set at 50 is the farthest you can zoom out in the cellview.

   You are prompted to "`Point at the target group`:"
8. Click the group you want to edit that appears in the information box.

   You are prompted to "`Select the figure(s) to add to group:`".
9. Click the Select button on the form, or click the group in the cellview.
10. Click the objects you want to add to the group.
11. The objects are added to the group. The solid line surrounding the group changes to include the new object or objects.
12. Press `Esc` to exit the command.

Ungrouping All Objects in a Group

To ungroup all objects from a group, follow these steps.

1. Choose *Edit – Group – Ungroup*.

   You are prompted to "`Point at the group to ungroup it`:"
2. Click the group you want to ungroup.

   Alternatively, you can select the group and right-click it. Select *Ungroup* from the *Group* context menu.

   The group is ungrouped. The line surrounding the group is deleted, and no longer appears when you place your pointer over the objects that were in the group.

Note: When you ungroup objects from a group, the child object of this group will not get deleted after you ungroup.

© 2020 All Rights Reserved.

Removing Objects from a Group

To remove an object from a group, follow these steps:

1. Choose *Edit – Hierarchy – Edit In Place*. Alternatively, you can right-click the group to display the *Group* context menu and select *Edit In Place.*

   You are prompted to "`Point at a shape in the cellView to be edited in place:`". The *Edit – Group – Remove from Group* remains grayed out until you have completed this step.

2. Click the group you want to edit.

3. Choose *Edit – Group – Remove from Group*.

   You are prompted to "`Select the figure(s) to remove from current group:`".

4. Click the objects you want to remove from the group.

   The object or objects are removed from the group. The solid line surrounding the group changes to exclude the deleted object or objects.

5. Click the `Escape` key.

6. Choose *Edit – Hierarchy – Return to Level* or right-click anywhere in the design canvas and select *Return* from the *Layout* context menu.

7. The dotted line surrounds only the remaining objects when you move your pointer over the group.

Editing Group Contents

You can edit the contents of a group either by using the *Edit In Place* command or by switching over to the transparent group mode.

Using Edit In Place

1. Click the group whose contents you want to edit.

2. Choose *Edit – Hierarchy – Edit In Place*. Alternatively, you can right-click the group to display the *Group* context-sensitive menu and select *Edit In Place.*

3. Perform the desired edit operations on the objects within the group.

4. Choose *Edit – Hierarchy – Return to Level* or right-click anywhere in the design canvas and select *Return* from the *Layout* context menu.

© 2020 All Rights Reserved.

Note: In a row type of FigGroup, you can edit or stretch the objects within the group when the selection mode is set to partial. By using this feature, you can stretch multiple rows or rails at the same time. For more information about FigGroup types, see the *Cadence Integrator's Toolkit Database Reference*.

Using Transparent Group Mode

The transparent group mode enables you to edit a user-defined group without doing EIP into the group. In this mode, the contents of user-defined groups become available for selection and editing at the top-level, while being retained as objects of their respective groups. You can edit objects in nested groups also at the top-level itself, without performing multiple EIPs.

Note: You can enable this mode only for the active window.

To edit groups in transparent group mode:

1. Activate the transparent group mode in one of the following ways:
   - In the Display Options Form, select the *Transparent Group* option in the *Display Controls* section.
   - On the *Options* toolbar (*Window - Toolbars - Options*), click the *Transparent Group* icon .
   - Set the window-specific environment variable in the CIW:

     ```
     window = hiGetCurrentWindow()
     window~>transparentGroup = t
     ```

   This makes all the user-defined groups in the design editable. In the transparent group mode, the pointer changes to white .

   Tip

   You can set the bannerFields environment variable to display `(T)` in the status banner whenever you activate the Transparent Group mode.

2. Perform the desired edit operations on the objects within a group(s).
3. After you are done with editing, you can leave the transparent group mode enabled for the current window session or disable it in one of the following ways:
   - In the *Display Options* form, deselect the *Transparent Group* option in the *Display Controls* section.
   - On the *Options* toolbar, toggle the *Transparent Group* icon on the *Options* toolbar to OFF.

© 2020 All Rights Reserved.

- Set the window-specific environment variable in the CIW:

```
window = hiGetCurrentWindow()
window~>transparentGroup = nil
```

Caution

If you intend to retain the Transparent Group setting ON for the remaining window session, Cadence recommends that you switch on the visibility (Vis) of** Groups **in the** Objects **assistant. This will display a boundary around groups even in Transparent Group mode and will prevent you from inadvertently modifying the objects belonging to any group.

In transparent group mode, contents of user-defined groups are available for selection and editing without doing EIP into groups. You can, therefore, select objects belonging to multiple groups and edit them simultaneously. For example, you can move shapes belonging to two different groups at the same time: Activate the transparent group mode, select shapes irrespective of the group to which they belong, and move them by the desired value. Similarly, you can select objects belonging to different groups and chop them simultaneously.

You cannot merge objects belonging to different groups. You also cannot attach an object from one group as the owner of an object belonging to another group. If you run *Make Cell* on objects belonging to different groups, the selected objects are removed from their respective groups. The resultant instance is placed at the top-level and is not included in any group.

In the transparent group mode, you can perform group-related commands, such as creating groups and adding, removing, and ungrouping objects in a group.

Selecting Objects

This section covers the following topics:

- Selection Basics
- Pre-Selection and Post-Selection of Objects
- Selecting Objects
- Making Objects Selectable and Unselectable
- Selecting a Group of Objects
- Selecting Vias in a Via Stack
- Selecting Multipart Rectangles

© 2020 All Rights Reserved.

- Selecting Path Components
- Selecting Path Segments

Selection Basics

There are two selection modes: full and partial. In full selection mode, you can select an entire object. In partial selection mode, you can select entire objects, edges, or corners of objects. Press `F4` to toggle the selection mode. The selection mode is displayed in the status banner and the Status Toolbar.

To Select...	Do This
One object	Click the object.
One edge or corner	Press `F4` to turn on partial selection. Then click the edge or corner.
Fully, a partially selected object	Press the `j` key. If the selected set contains all fully selected objects, pressing `j` extends the selection based on the current granularity.
A group of objects	Click and drag to create a selection box around the group.
An additional object	Press `Shift` and click inside the object.
	Press the `a` key and click the object.
An additional edge or corner	Press `F4` to turn on partial selection. Then press `Shift` and click the edge or corner.
An additional group	Press `Shift` and click and drag to create a selection box around the objects.
	Press `Shift-a`, then create a selection box around the objects.
A group of edges or corners	Press `F4` to turn on partial selection. Then create a selection box around the edges or corners.
All objects	Press `Control-a` or choose *Edit – Select – Select All.*
An object located under other objects or cycle through overlapping objects	Select the top object, press `Control-y`. Continue to press `Control-y`, or click right, until you select the object you want.

© 2020 All Rights Reserved.

Pre-Selection and Post-Selection of Objects

You can select objects either before or after you start a command.

If you select the object before starting the command (pre-selection mode),

- Editing commands do not automatically repeat, even if repeat mode is set on.
- The resultant objects remain selected after the command completes.
- Some editing commands prompt you for a starting point, called the reference point, for the edit.

If you select the command and then the object (post-selection mode),

- Editing commands automatically repeat if repeat mode is on (the default).
- The resultant objects are deselected after the command completes.
- Editing commands do not necessarily prompt you for a reference point. If you click to select the first object, the editor uses that point as the reference point.

Selecting Objects

You can select objects by defining the selection shape.

- Selecting Objects By Defining an Area
- Selecting Objects by Drawing a Line
- Fully Selecting Partially Selected Objects

Selecting Objects By Defining an Area

To select objects within an enclosed area:

1. Select *Edit – Select – Select By Area*.
2. Press `F3`.

© 2020 All Rights Reserved.

The Select Area Shape Type Form appears.

3. Select the shape type as *rectangle* or *polygon*.
4. In the design window, draw the shape selected in the form.

   Any objects enclosed within the drawn shape are selected.

   Note: The selection of fully or partially enclosed objects within the drawn shape is based on the settings in the *Area Selection Controls* section in the Selection Options Form.
5. You can continue to select more shapes by drawing a rectangle or polygon around them.
6. To add objects to the previous selection, press `Shift` and draw the shape around the additional objects to be selected.
7. To remove objects from the previous selection, press `Ctrl` and draw the shape around the objects to be removed from the selected set.
8. Press `Esc` to cancel the command.

Selecting Objects by Drawing a Line

To select objects by using a line:

1. Select *Edit – Select – Select By Line*.
2. In the design window, click at the first point of the line.
3. Click at the next point of the line.
4. You can continue to click to define more points of the line.
5. Double-click to end the line.

   Any objects overlapping the line are selected.
6. You can continue to select more shapes by drawing a line to overlap them.
7. To add objects to the previous selection, press `Shift` and draw a line to overlap the additional objects to be selected. Keep `Shift` pressed until you double-click to end the line.

© 2020 All Rights Reserved.

8. To remove objects from the previous selection, press `Ctrl` and draw a line to overlap the objects to be removed from the selected set. Keep `Ctrl` pressed until you double-click to end the line.
9. Press `Esc` to cancel the command.

Fully Selecting Partially Selected Objects

To fully select objects that are partially selected:

1. Select *Edit – Select – Extend selection to object* [`J`].
2. In the design window, select the partially selected object that you want to fully select.
3. Press `Esc` to cancel the command.

Making Objects Selectable and Unselectable

You can use the following methods to control the selectability of objects:

- By Using the Layers Assistant
- By Using the Protection Commands

© 2020 All Rights Reserved.

By Using the Layers Assistant

You can use the *Layers* assistant to control the selectability of objects on specific layer-purpose pairs. To control the selectability of individual objects, see By Using the Protection Commands.

© 2020 All Rights Reserved.

By Using the Protection Commands

You can use the commands on the *Edit – Select* menu to set individual objects as selectable or unselectable (selection protection). To control the selectability of objects based on the LPP they are on, see By Using the Layers Assistant.

Choose from the Edit – Select Menu	To...
Set Selection Protection	Protect objects from being selected.
Clear Selection Protection	Start an enter function that removes protection from selected objects. Start this function, then either click or use area selection to specify the objects you want to make selectable again.
Clear All Selection Protection	Make all currently protected objects selectable again. **Note:** Only the objects that were marked protected by using the *Set Selection Protection* command are made selectable. The selectability of objects set using the *Layers* assistant is not affected.
Override Selection Protection (Toggle Command)	The command appears as a check box on the submenu. When on, to select objects that have been marked protected. In this mode, you can perform basic edit operations on the protected objects. When off, to not allow selection of protected objects. **Note:** If you turn off this command after selecting a protected object, the protected object is deselected.

© 2020 All Rights Reserved.

Choose from the Edit – Select Menu	To...
Highlight Protected Objects (Toggle Command)	The command appears as a check box on the submenu. When on, to display a halo around objects that are marked protected. This is the default mode. When off, to remove the halo around objects that are marked protected.
Selection Protection Options	Configure the settings for highlighting the objects that are marked protected. This command opens the Selection Protection Highlight Options Form.

These commands work in both pre and post-selection modes. The *Clear Selection Protection* command works in pre-selection mode only when *Override Selection Protection* is on.

Selecting a Group of Objects

The following selection from *Options – Selection* on the cellview window. You can toggle back and forth between full and partial selection using the `F4` function key, which will switch between the full and partial modes selected on the Selection Options Form. The following diagram illustrates the seven selection modes:

You can select a group of objects by creating a selection box around them. This is sometimes referred to as window selection.

There are currently seven modes of area selection:

- Full selection which fully selects objects you completely enclose with the area selection box. This is called *Enclosed Figures* in the *Selection Options* form.
- Full selection which fully selects objects you completely or partially enclose with the area selection box. This is called *Enclosed and Crossed Figures* in the *Selection Options* form.
- Full selection which fully selects objects you partially enclose with the area selection box and does not select fully enclosed objects. This is called *Crossed Figures* in the *Selection Options* form.

© 2020 All Rights Reserved.

- Partial selection which fully selects all objects you completely enclose with the area selection box, vertices of any objects you partially select with the area selection box, and edges of any objects you enclose with the area selection box. This is called *Vertex* in the *Selection Options* form.
- Partial selection which fully selects all objects you completely enclose with the area selection box, and the edges of any objects you partially enclose with the area selection box. This is called *Enclosed Edges* in the *Selection Options* form.
- Partial selection which fully selects the edges of any objects you enclose with the area selection box, and does not select fully enclosed objects. This is called *Crossed Edges* in the *Selection Options* form.
- Partial selection which fully selects all objects you completely enclose with the area selection box, the vertices of any objects you partially enclose with the area selection box, and the edges of any objects you enclose with the area selection box. This is called *Enclosed and Crossed Edges* in the *Selection Options* form.

You can quickly toggle between the full and partial selection modes, by clicking the *Full Select* icon or the *Partial Select* icon on the *Options* toolbar or by pressing `F4`.

Parameterized cells can only be fully selected. You fully select them when you completely enclose them with the area selection box, irrespective of the selection mode setting.

© 2020 All Rights Reserved.

Full Selection Options

Figure 1

Full selection, enclosed

Figure 2

Full selection, enclosed & crossed

Figure 3

Full selection, crossed

© 2020 All Rights Reserved.

Partial Selection Options

Partial selection, vertices

Partial selection, enclosed

Partial selection, crossed

Partial l selection, enclosed & crossed

Unselected Object

Selected Object

Search Area

Selected Edge

Selecting Vias in a Via Stack

To select all the vias in a via stack, use one of the following methods:

- Select one via in the via stack, and press Ctrl + o to select the remaining vias.

© 2020 All Rights Reserved.

- Click and drag to create selection box around the via stack.
- Enable one of the following options to switch on the via stack selection mode:
  - *Via Stack* option in the Selection Options Form (*Options – Selection*)

  - *Via Stack* icon on the *Options* toolbar
- Choose *Via with matching sizes between common layers* option in the Selection Options Form (*Options - Selection*) to select only those vias that have the same size on common layers.

© 2020 All Rights Reserved.

Note: Full and partial selection modes do not have an impact on via stack selection. You can select the complete via stack even when the *Partial Select* button on the *Options* toolbar is on.

Selecting Multipart Rectangles

A multipart rectangle is a single object composed of multiple parts on the same or different layers. The parts consist of one or more named master rectangles and one or more arrays of unnamed subrectangles. Each named master rectangle is a separate object with ROD attributes, created at level zero in the hierarchy. Each unnamed sub-rectangle is an ordinary, unnamed database shape with no ROD attributes, created at level zero in the hierarchy.

The multipart rectangle below has one master rectangle and two arrays of subrectangles. Both arrays of subrectangles are offset from the master rectangle.

Selecting a master rectangle of a multipart rectangle is similar to selecting a regular rectangle: you point and click or create a selection box in a similar manner.

- In full selection mode, when you select any part of a master rectangle or its associated subrectangles, the whole master rectangle and all of its subrectangles are selected. The master rectangle is highlighted on the current selection layer, while the associated arrays of subrectangles are highlighted on different layers. This lets you see which part is the master and which parts are subrectangles.
- In partial selection mode (press `F4`), you can select an edge or vertex of a master rectangle.

For more information about multipart rectangles, see Multipart Rectangles and the `rodCreateRect` function in the *Virtuoso Relative Object Design User Guide.*

© 2020 All Rights Reserved.

Selecting Path Components

- Selecting a Whole Path
- Selecting Path Ends
- Selecting a Path Vertex

Selecting a Whole Path

To select a whole path,

➤ Do one of the following:

❑ In full selection mode, point anywhere on the path and click.

As you point, the whole path is highlighted with dotted lines.

Point anywhere on the path and click.

Highlighting shows the whole path is selected.

❑ In partial selection mode, point to the outer edge of any path segment and click.

As you point, the whole path is highlighted with dotted lines.

Point to the outer edge of a path segment and click.

Highlighting shows the whole path is selected.

© 2020 All Rights Reserved.

- In full or partial selection mode, create a selection box around the whole path.

Create a selection box around the whole path.

Highlighting shows the whole path is selected.

Selecting Path Ends

To select a path end,

➤ Do one of the following:

- In partial selection mode, point to an end of the path centerline and click.

  As you point, the end of the centerline is highlighted with dotted lines.

Point to an end of the path centerline and click.

Highlighting shows the path end is selected.

- In partial selection mode, create a selection box around an end of the path.

Create a selection box around an end of the path.

Highlighting shows the path end is selected.

© 2020 All Rights Reserved.

Selecting a Path Vertex

A vertex is a point on the path centerline where two segments join.

To select a vertex,

➤ Do one of the following:

❑ In partial selection mode, point to the vertex and click.

As you point, the vertex is highlighted with dotted lines.

Point to the vertex and click.

Highlighting shows the vertex is selected.

If you are not able to click a vertex, the current setting of the *Gravity* option is too restrictive. In the Layout Editor Options form, turn off *Gravity*.

❑ In partial selection mode, create a selection box around the vertex.

Create a selection box around the vertex.

Highlighting shows the vertex is selected.

Selecting Path Segments

To select a single path segment,

➤ Do one of the following:

❑ In partial selection mode, point to the centerline in the middle of a segment and click.

© 2020 All Rights Reserved.

As you point, the centerline of the segment is highlighted with dotted lines.

Point to the path centerline in a segment and click.

Highlighting shows the path segment is selected.

- In partial selection mode, create a selection box around a segment. Make sure both end points of the segment centerline are inside the box.

Create a selection box around the centerline of a segment.

Highlighting shows the path segment is selected.

To point select more than one path segment, ensure that the *Spine* option in the *Selection Options* section of the Selection Options Form (*Options - Selection*) is selected. All pathsegs connected to and of the same width as the first pathseg clicked get selected. This behavior differs for the full and partial selection modes. The following example illustrates this.

© 2020 All Rights Reserved.

- In full selection mode, clicking anywhere on the pathsegs 1, 2, 3, or 4 selects all pathsegs forming the spine. The width of the pathsegs 5 and 6 is different from the pathsegs 1 and 4 where the click points A, B, and C lie. Therefore, pathsegs 5 and 6 do not get selected.

Point of Click	A	B	C
Pathsegs Selected	1+2+3+4	1+2+3+4	1+2+3+4

- In partial selection mode, clicking at point A selects only pathseg 1. Clicking at point B, which is at the outer edge of the pathseg, selects all pathsegs forming the spine. Clicking

© 2020 All Rights Reserved.

at point C selects both pathsegs 3 and 4 because they are of the same width and have the same direction.

Point of Click	A	B	C
PathSegs Selected	1	1+2+3+4	3+4

If you area select a spine such that the area selection box does not enclose all the pathsegs of the spine, then neither the spine nor the pathsegs within the area selection box are selected. pathsegs within the area selection box are selected only if the *Partial Select* mode is on. To prevent issues when the spine comprises a large number of pathsegs, you can configure the environment variable maxPathSegInSpineAreaSelection.

Deselecting Objects

You deselect objects almost the same way you select objects, except you press `Control` as you click an object, corner, or edge.

© 2020 All Rights Reserved.

Here is a quick reference to all of the ways you can deselect objects.

To Deselect...	Do This
All objects	click an empty portion of the design, or press `Control-d` or choose *Edit – Select – Deselect All* or right-click and select *Deselect All* from the context menu.
One or more objects	Press `Control` and click the object or right-click and select *(De)Select Under Cursor* from the context menu and select the objects you want to deselect from the sub-menu.
A group	Press `Control` and click and drag to create a deselection box around the objects.

© 2020 All Rights Reserved.

8

Creating and Editing Multipart Paths

This chapter shows you how to use the Virtuoso® Layout Suite layout editor to create a multipart path (MPP), how to edit and delete subparts, how to save an MPP as a template in your technology file or in an ASCII file so you can use it again, and how to load MPP templates from ASCII files. This chapter discusses the following topics:

- Creating Multipart Paths
- Flow for Creating Multipart Paths
  - Defining the Master Path
  - Defining Connectivity for the Master Path
  - Adding a New Subpart
  - Defining Connectivity for a Subpart
  - Editing a Subpart
  - Deleting a Subpart
  - Creating a Multipart Path in a Cellview
  - Saving Multipart Path Values as a Template
  - Loading an ASCII Multipart Path Template File
- Creating an MPP Guard Ring
- Setting Multipart Path Environment Variables
- Selecting Multipart Paths
- Moving Multipart Paths
- Splitting Multipart Paths
- Reshaping a Multipart Path
- How Chopping Affects Multipart Paths

© 2020 All Rights Reserved.

Creating Multipart Paths

You can use the *Create Multipart Path* command to create named paths that have relative object design (ROD) information associated with them. You can create simple one-part paths or complex paths containing several parts, such as a guard ring, transistor, bus, or shielded path. A path that consists of more than one part is called a *multipart path*.

A multipart path (MPP) consists of a single master path and one or more subparts. The master path is the defining object; all subparts are based upon and exist in relation to the master path. You can define any number of subparts for a multipart path. Subparts can be any or all of the following: offset sub-paths, enclosure sub-paths, and sets of sub-rectangles. Also, you can define connectivity for any or all parts of a multipart path.

Note: In the layout editor, a multipart path (master path and all its subparts) is treated as a single object.

A ROD multipart path has many advantages over a regular path; for example, you can access it by name, even through levels of hierarchy, and you can chop it.

You define a multipart path by filling in the fields of the *Create Multipart Path* form and the *ROD Subpart* form. After specifying information in the forms, you click in the current layout cellview window to define the points for the master path of the MPP. You can change the values in the forms while you are clicking points, until you double-click to complete the MPP.

When you define a new type of MPP, you might want to save the values in the forms as a template so that you can create other MPPs that are same or similar. A *template* is the set of values that specify a particular multipart path, such as a guard ring. You can define as many MPP templates as you wish. You save templates either to your technology library, if you have write permission, or to an ASCII file.

For a complete description of ROD objects and multipart paths, see the *Virtuoso Relative Object Design User Guide.*

With the *Create Multipart Path* command, you can:

- Create a single-part path using the form defaults.
- Choose an MPP template using the *MPP Templates* field.
- Modify the values in the *Create Multipart Path* and *ROD Subpart* forms to create a different MPP.
- Save the current form values as a template in your technology file (if you have write permission) or in an ASCII file.

© 2020 All Rights Reserved.

- Load an ASCII MPP template file from your local hierarchy to add the templates to the list in the MPP Templates field. For additional ways to load ASCII MPP template files, see the `leDefineMPPTemplate` SKILL function in the Custom Layout SKILL Function Reference.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The `Create Multipart Path` command does not support curved shapes.

About Creating a Multipart Path in a Layout Window

In a layout cellview window, you create the master path for an MPP just as you create a regular path: by clicking to enter a series of points. You define the characteristics of the master path and of all its subparts by filling in the fields in the *Create Multipart Path* and *ROD Subpart* forms. As you draw an MPP in a cellview, you can make changes to the forms and see the results of your changes immediately, until you double-click to complete the MPP.

Note: You can make as many changes as you want to an MPP using the *Create Multipart Path* form and the *ROD Subpart* form prior to entering the last point of your MPP in the layout window. Once you enter the last point, you cannot use these forms to change the MPP. You can change a completed MPP using the *Edit Properties* form.

If you are creating a multipart path with abutted ends and the centerline of the master path intersects itself, the system will not create the path. A dialog box appears, saying that no path was created because it would have been self-intersecting. The system does not allow you to create a path whose master path centerline intersects itself, and does not allow you to create a path with one or more self-intersecting sub-paths.

When you first start the layout editor software, the *Template Name* and *MPP Template* fields in the *Create Multipart Path* form are set to *New* and the values of the other fields are system defaults, unless the `mppTemplate` environment variable is set. If `mppTemplate` is set, *New* is replaced with a specific template name, and the other fields are set to the template values.

For some fields, the system looks at your technology file for the design rule for the layer and, if it is defined, uses it as the default. For example, the system looks for the `minWidth` rule for the current layer to use as the default for the *Width* field.

You can create an MPP in a layout cellview window using the default values without even opening the form, or you can open the form and change the values. Or you can choose to use the values from an MPP template that was saved earlier. When you choose an MPP template from the *MPP Template* field, the system fills in the rest of the fields in the *Create Multipart Path* form and *ROD Subpart* form with the values specified in the template. The system assigns system default values to fields for which values are not specified in the template. You can use the template values as they are or modify them as desired.

© 2020 All Rights Reserved.

Note: If you want to define connectivity for any part of an MPP, you must open the *Create Multipart Path* form, and in the *Connectivity* section, type in a net name.

If you want to designate a particular MPP template as your default rather than use the system default values, you can do so by setting the `mppTemplate` layout editor environment variable. For information about how to set this environment variable, see Setting Multipart Path Environment Variables.

About Saving a Multipart Path as a Template

You can save the values currently in the *Create Multipart Path* and *ROD Subpart* forms as a template with a unique name. This lets you create the same or similar MPPs. You can save MPP values as templates in the following two ways:

- To the temporary technology file in virtual memory
  If you have write permission, you can also save to your technology library on disk
- To an ASCII file in your local hierarchy

When you click the *Save Template* button on the *Create Multipart Path* form, the system displays the *Save Multipart Path Templates* form. This form lets you save any template named in the *Template List* field or save all templates named in the *Template List* field.

The remainder of this chapter describes how to define, change, and save MPP templates using the *Create Multipart Path* form. Instead of using a graphical user interface to define an MPP, you can create an ASCII file to define one or more MPP templates, using the `leDefineMPPTemplate` function. For information about this function, see `leDefineMPPTemplate` in the *Custom Layout SKILL Functions Reference*.

© 2020 All Rights Reserved.

Flow for Creating Multipart Paths

The tasks involved in creating an MPP are shown in a suggested flow below. To see the steps for a task, click the task name.

You must define the master path. Optionally, you can specify connectivity for it.

Optionally, you can specify any number of subparts. You can specify connectivity for any or all subparts.

Note: You must click *OK* or *Apply* in the ROD Subpart form to complete adding, editing, or deleting a subpart.

Optionally, you can save your MPP form values as a template.

© 2020 All Rights Reserved.

Defining the Master Path

1. To open the Create Multipart Path form, in the layout window, choose *Create – Multipart Path* and press `F3`.

   The Create Multipart Path Form opens.

   The first time you open form, the values of fields will default to the values set in the technology file and can vary depending on which layer is current in the Layers Panel. If there is no rule defined in the technology file, the system uses a system default

   Note: When you select *Create – Multipart Path*, you might see warning messages about no minimum width rule defined for the first valid layer listed in the *Layers* assistant.

   You can use a template to create your MPP or create a new one from scratch. The templates already defined in your technology library are listed in the MPP Template field. You can also load MPP templates from ASCII files to add them to the templates listed in the MPP templates field and to the temporary technology file in virtual memory.

   Note: Templates do not contain net names because each MPP you create from the same template could be on a different net. When you choose a template that contains pin connectivity, the system displays a message at the bottom of the cellview window prompting you to enter a valid net name. You must enter a name in the *Net Name* field for each part (master path and/or subparts) that has pin connectivity. If you click in the cellview window before entering the required net name(s), the system beeps.

2. To load templates from ASCII MPP template files, see Loading an ASCII Multipart Path Template File.

3. To create a new MPP in the current layout window (and a new MPP template), or to use an existing template, do one of the following:

   - To create a new MPP from scratch, without using an existing template,

     - When you do not want to save your MPP as a template, in the *MPP Template* field, click *New*.

       Choosing *New* resets all values in the Create Multipart Path and ROD Subpart forms to the system defaults and sets the *Template Name* field to *New*.

     - When you want to save your MPP as a template, for *Template Name*, type a unique name, such as `myMPP`.

       When you create a new MPP without using a template, you need to set the layer and purpose in the *Layers* assistant to the value you want for the master path.

     - Check the *Layers* assistant for the current layer and purpose for the master path, and modify it if desired.

© 2020 All Rights Reserved.

- To use an existing MPP template,
  - Select the name of the template you want from the *MPP Template* field.

    When you choose an existing MPP template, the system changes the current layer and purpose in the *Layers* assistant to the layer and purpose defined in the MPP template for the master path.

    If the layer specified in the template is not a valid layer, the system displays a dialog box asking if you want the system to add the master path layer to the *Layers* assistant. If you do not, the system uses the current layer in the *Layers* assistant.
  - If a dialog box asks if you want to add the template layer to the list of valid layers, click *OK* or *Cancel*.

    If you want a different layer, you can change the layer in the *Layers* assistant.
  - Check the *Layers* assistant for the current layer and purpose for the master path, and modify it if desired.

Now you can modify the values in the *Create Multipart Path* form.

The *ROD Name* field specifies a unique name for the ROD object information about the MPP in the layout window. You can access information about the MPP through multiple levels of hierarchy using this name. The default ROD object name consists of the prefix `path` followed by a number.

4. For *ROD Name*, accept the default system-assigned name or type in a name that is unique in the current cellview.

5. For *Technology Library* select from the cyclic field the library in which you want to save the Multipart Path. This cyclic list contains a list of all the technology libraries in the effective technology libraries (those that are defined in the technologies data base) that contain only the multipart path templates defined in the data base.When a different technology library is selected in the *Technology Library* field the *MPP Template* field is updated to contain the templates contained in that library.

6. For *Choppable*, do one of the following:

- To make the master path and all subparts choppable, turn on *Choppable*.
- To make the master path not choppable (subparts can be choppable or not), turn off *Choppable*.

7. For *Width*, do one of the following:

- Type in a positive integer or floating-point number.

© 2020 All Rights Reserved.

- To use the `minWidth` rule for the master path layer from the technology file, do not change the *Width* field.
- If the `minWidth` rule is not defined in the technology file, and
  - This is the first time you have used this form in your current editing session and you do not specify a width, the system supplies a default and displays an warning saying that the default width is being used.
  - You have used this form earlier in your current editing session, the previous value for this field is displayed and the system displays a warning.

8. For *End Type*, choose a value. If you want to specify a beginning or ending extension, you must choose *variable*.

The *Offset* and *Justification* fields are interdependent. *Justification* determines the part of the master path (master path centerline, left edge, or right edge) that is offset from the points you enter (the point list), relative to the direction of the master path. The direction of the master path is determined by the sequence in which you enter its points.

9. For *Justification*, choose one of the following values:

- When *Offset* is `0`:

  center to make the master path centerline coincident with the point list

  left to make the right edge of the master path coincident with the point list

  right to make the left edge of the master path coincident with the point list

- When *Offset* is not `0`:

  center to offset the centerline of the master path from the point list

  left to offset the left edge of the master path from the point list

  right to offset the right edge of the master path from the point list

10. For *Offset*, do one of the following:

- To make the centerline of the master path coincident with the points you enter, leave *Offset* set to `0` and *Justification* set to *center*.
- To offset the master path from the point list, type a signed integer or floating-point number.

  The part of the master path that is offset from the point list is determined by the value of the *Justification* field. Whether you enter a positive or negative number depends

© 2020 All Rights Reserved.

on where you want the master path in relation to the point list. The following table lists various combinations of Offset with respect to the position of the master path.

Direction of Point List Segment	Positive Offset	Negative Offset
Positive along X axis →	Above	Below
Negative along X axis ←	Below	Above
Positive along Y axis ↑	Left	Right
Negative along Y axis ↓	Right	Left

For illustrations showing the location of a master path in relation to the point list, see "Examples of Offsetting the Master Path" in the *Virtuoso Relative Object Design User Guide*.

11. For *Begin Extension*, do one of the following:

- If you want the master path layer to begin at the first point you enter for the master path, leave *Begin Extension* set to `0`.

  To specify an extension, the *End Type* field must be set to *variable*.

- If you want the master path layer to extend beyond the first point you enter for the master path, type a positive integer or floating-point number.

12. For *End Extension*, do one of the following:

- If you want the master path layer to end at the last point you enter for the master path, leave *End Extension* set to `0`.

  To specify an extension, the *End Type* field must be set to *variable*.

- If you want the master path layer to extend beyond the last point you enter for the master path, type a positive integer or floating-point number.

13. For *Connectivity*, do one of the following:

- If you do not want to define connectivity, leave *Connectivity* set to *None* and return to Flow for Creating Multipart Paths to continue with the next step in the flow.

- If you want to associate the master path with a net or specify it as a pin, continue to the next section, Defining Connectivity for the Master Path.

© 2020 All Rights Reserved.

Defining Connectivity for the Master Path

You can associate the master path with a net, and you can designate the master path as a pin. Changes you make to the *Create Multipart Path* form immediately affect the master path in the layout window.

To define connectivity for a master path, do one of the following:

1. To assign the master path to a net, do the following:

   - For *Connectivity*, choose *Net*.

     The form expands to show the net connectivity field.

   - For *Net Name*, type the name of the net.

     If you want to assign the master path to a pin, continue with the next step; otherwise, you have completed assigning the master path to a net. Return to Flow for Creating Multipart Paths to continue with the next step in the flow.

2. To assign the master path to a pin, do the following:

   - For *Connectivity*, choose *Pin*.

     The form expands to show the net and pin connectivity fields.

   - For *Net Name*, type the name of the net.

     When you choose *Pin*, you must enter a net name or the system beeps when you click in the cellview window to begin creating the MPP.

   - For *I/O Type*, choose one of the following: *input*, *output*, *inputOutput*, *switch*, or *jumper*.

   - For *Access Direction*, choose the part(s) of the pin to which routing can be connected.

   - Turn on *Display Terminal Name* if you want to create a text-display object to display the net name for the pin.

     Note: If pin names are not visible in the cellview, the *Pin Names* option in the Display Options form is not turned on. To display all pin names, turn on *Pin Names* in the Display Options form.

     The system uses the values of four fields to calculate the position of the text-display object: *Reference Handle*, *Offset X*, *Offset Y*, and the *Justification* field on the Pin Name Display form.

© 2020 All Rights Reserved.

- Click the *Display Terminal Name Option* button if you want to set pin name display options, such as font, height, layer, and justification on the Pin Name Display form.
- For *Reference Handle*, choose where you want to align the pin name in relation to the master path or subpart.

  You can offset the pin name from the object in the direction of the X and/or Y axis.
- For *Offset X*, type a positive or negative floating-point number to offset the origin point of the pin name along the X axis from the point specified by *Reference Handle.*
- For *Offset Y*, type a positive or negative floating-point number to offset the origin point of the pin name along the Y axis from the point specified by *Reference Handle.*

You have completed defining connectivity. Return to Flow for Creating Multipart Paths to continue with the next step in the flow.

© 2020 All Rights Reserved.

Adding a New Subpart

You can add any number of subparts for the current master path in the layout window. If you want to use the subpart values again, you need to save them as a template along with the master path.

1. To specify a new subpart for the current master path, in the Create Multipart Path form, click the *Subpart* button.

   The ROD Subpart Form appears.

2. Click to choose the type of subpart you want to create: *Offset Subpath*, *Enclosure Subpath*, or *Subrectangle*.

   The system displays the subpart fields at the bottom of the form. If any subparts of the type you selected already exist, the system displays one line of information for each subpart in the scroll window at the top of the form.

3. To define a new subpart, go to one of the following topics:

   Defining an Offset Subpath

   Defining an Enclosure Subpath

   Defining One or More Sets of Subrectangles

Defining an Offset Subpath

You can define a subpath coincident with an edge of the master path, on the left or right side of the master path, or overlapping the master path. This is called an *offset subpath*.

You can specify the width of the offset subpath or let it default to the `minWidth` rule for the subpath layer from the technology file. The location of an offset subpath in relation to the master path depends on the values of the *Separation* and *Justification* fields, in relation to the direction of the master path.

For a detailed description of offset sub-paths, see Offset Subpaths in the *Virtuoso Relative Object Design User Guide*.

Set values for the offset subpath section as follows:

1. For *Layer*, choose the layer for the offset subpath.

2. Turn on *Choppable* if the master path is set to choppable; otherwise set the offset subpath to choppable or not choppable, as you wish.

© 2020 All Rights Reserved.

3. For *Begin Offset*, type a signed integer or floating-point number to specify the starting edge of the subpath in relation to the starting edge of the master path. A positive number extends the end of the subpath beyond the end of the master path; a negative number retracts the end of the subpath from the end of the master path.

4. For *End Offse*t, type a signed integer or floating-point number to specify the ending edge of the subpath in relation to the ending edge of the master path. A positive number extends the end of the subpath beyond the end of the master path; a negative number retracts the end of the subpath from the end of the master path.

5. For *Width*, do one of the following:

   - Type in a positive integer or floating-point number.
   - To use the `minWidth` rule for the subpath layer from the technology file, do not change the *Width* field.

   If the `minWidth` rule is not defined in the technology file, you must specify a width; otherwise, the width remains zero and no subpath is created.

6. For *Separation* and *Justification*, type a signed integer or floating-point number for *Separation* and choose a setting for *Justification*, as shown in the table in section Rectangles in the Corners of Subrectangle Segments.

   Whether you enter a positive or negative number depends on where you want to define the subpath in relation to the master path.

   For illustrations showing the location of offset subpaths in relation to the master path, see "Examples of Offset Subpaths" in the *Virtuoso Relative Object Design User Guide*.

7. For *Connectivity*, do one of the following:

   - If you do not want to define connectivity, leave *Connectivity* set to *None*.
   - If you want to associate the subpart with a net or specify it as a pin, go to Defining Connectivity for a Subpart.

8. To add the new offset subpath to the window at the top of the *ROD Subpart* form, click *Add*.

   The system displays a line of data for the new offset subpath in the scroll window.

   You **must** click *OK* or *Apply* at the top of the *ROD Subpart* form to make the new offset subpath part of the MPP.

9. To save the new offset subpath as part of the MPP, in the *ROD Subpart* form, click *OK* or *Apply*.

© 2020 All Rights Reserved.

So far, you are defining a new offset subpath only for the MPP in the current layout window. If you want to add the new offset subpath to an MPP template, you need to save your changes to the template.

10. To save your field values as an MPP template, go to Saving Multipart Path Values as a Template.

Note: You can also use the context sensitive menu in the *Property Editor* assistant to add an offset subpath.

You have completed adding an offset subpath to your MPP. Continue to the next section or return to Flow for Creating Multipart Paths to continue with the next step in the flow.

Defining an Enclosure Subpath

You can define a subpath with its centerline on the centerline of the master path and its width calculated using the width of the master path plus a positive or negative enclosure value. This is called an *enclosure subpath*. The enclosure determines by how much the subpath is enclosed by the master path or by how much the master path is enclosed by the subpath.

For a detailed description of enclosure subpaths, see Enclosure Subpaths in the *Virtuoso Relative Object Design User Guide*.

Set values for the enclosure subpath section as follows:

1. For *Layer*, choose the layer for the enclosure subpath.
2. Turn on *Choppable* if the master path is set to choppable; otherwise, set the enclosure subpath to choppable or not choppable, as you wish.
3. For *Begin Offset*, type a signed integer or floating-point number to start the edge of the subpath before or after the starting edge of the master path.
4. For *End Offset*, type a signed integer or floating-point number to end the edge of the subpath before or after the ending edge of the master path.

   To calculate the width of an enclosure subpath, the system **subtracts** two times the enclosure value from the width of the master path:

   ```
   Width of Enclosure Subpath = Width of Master Path - (2 * Enclosure Value)
   ```

5. For *Enclosure*, type an integer or floating-point number for the enclosure of the subpath in relation to the master path.
   - To define a subpath that is narrower than the master path, type a positive number.
   - To define a subpath that is wider than the master path, type a negative number.

© 2020 All Rights Reserved.

- To use the `minExtensionDistance` rule for the subpath layer from the technology file, keep the default.

  If `minExtensionDistance` for the subpath layer is not defined in your technology file, and you do not type a value, the system sets the value to zero.

For illustrations showing the location of enclosure subpaths in relation to the master path, see "Examples of Enclosure Subpaths" in the *Virtuoso Relative Object Design User Guide*.

6. For *Connectivity*, do one of the following:

   - If you do not want to define connectivity, leave *Connectivity* set to *None*.
   - If you want to associate the subpart with a net or specify it as a pin, go to Defining Connectivity for a Subpart.

7. To add the new enclosure subpath to the scroll window at the top of the ROD Subpart form, click *Add*.

   The system displays a line of data for the new enclosure subpath in the scroll window.

   You **must** click *OK* or *Apply* at the top of the ROD Subpart form to make the new enclosure subpath part of the MPP.

8. To save the new enclosure subpath as part of the MPP in the cellview, click *OK* or *Apply*.

   So far, you are defining a new enclosure subpath only for the MPP in the current layout window. If you want to add the new subpath to an MPP template, you need to save your changes to the template.

9. To save your addition to an MPP template, go to Saving Multipart Path Values as a Template.

Note: You can also use the context sensitive menu in the *Property Editor* assistant to add an enclosure subpath.

You have completed adding an enclosure subpath to your MPP. Continue to the next section or return to Flow for Creating Multipart Paths to continue with the next step in the flow.

Defining One or More Sets of Subrectangles

You can define one or more *sets of subrectangles* in relation to the centerline of any master path. A set can contain one or more subrectangles.

You can specify the width and length of the subrectangles or let them default to the `minWidth` rule for the subrectangle layer from the technology file. You can offset the centerline of the

© 2020 All Rights Reserved.

subrectangles from the master path centerline, and you can offset the edge of the first and last subrectangles from the ends of the master path.

You can define connectivity for a set of subrectangles. When you specify that a set of subrectangles is a pin, each rectangle in the set becomes a pin. In the layout editor, the whole multipart path (master path and all subparts) is treated as a single object.

The location of a set of subrectangles in relation to the master path depends on the values of the *Separation* and *Justification* fields, in relation to the direction of the master path.

Rectangles in the Corners of Subrectangle Segments

When you create a subpart consisting of a set of subrectangles, the system creates a subrectangle in the corner formed by each segment of the subrectangle subpath, and then fills in the rest of the segment with subrectangles, following the spacing you specify.

There are several reasons that you might want to keep rectangles out of the corners of your subrectangle subpaths. For example, when you specify multiple subrectangle subpaths, placing rectangles in the corners of segments might cause individual subrectangles to be out of alignment vertically or horizontally with the subrectangles in the other subrectangle subpaths.

If you do not want rectangles in the corners of segments, you can keep them out by specifying the *Begin Segment Offset* and *End Segment Offset* options in the ROD Subpart form.

© 2020 All Rights Reserved.

The system measures the Segment Offset values from the edge of the master path, as shown below.

When setting values for the Segment Offset options, keep in mind:

- The *Begin Segment Offset* value does not apply to the first segment; the position of the first rectangle in the first segment is determined by the value of the *Begin Offset* option.
- The *End Segment Offset* value does not apply to the last segment; the position of the last rectangle in the last segment is determined by the value of the *End Offset* argument.
- The Segment Offset arguments work in combination with the *Begin Offset* and *End Offset* arguments.
- If the segment begins before the full distance specified by the *Begin Segment Offset* option, then the *Begin Segment Offset* is not applied, and a subrectangle could be placed at the corner of the subrectangle subpath.
- If the segment ends before the full distance specified by the *End Segment Offset* option, then the End Segment Offset is not applied, and a subrectangle could be placed at the corner of the subrectangle subpath.
- When applying the Offset options, the system will not create a spacing error. If the first rectangle in a subrectangle subpath segment would be too close to the last rectangle in the previous segment, the system discards the last rectangle in the previous segment.

When you specify the Offset options, you need to check the results to make sure the corners are as desired. If they are not, you should change the values of the Offset arguments and try again.

© 2020 All Rights Reserved.

For a detailed description of subrectangles, see Sets of Subrectangles in the *Virtuoso Relative Object Design User Guide*.

Fill in the subrectangle section as follows:

1. For *Layer*, choose the layer for the set of subrectangles.
2. Turn on *Choppable* if the master path is set to choppable; otherwise, set the subrectangles to choppable or not choppable, as you wish.
3. Turn on *Diagonal* if you want to create subrectangles on 45-degree path segments. This option works only when all subrectangles on the path are square.
4. For *Begin Offset*, type a signed integer or floating-point number to start the first subrectangle before or after the starting edge of the master path.
5. For *End Offset*, type a signed integer or floating-point number to end the last subrectangle before or after the ending edge of the master path.

   Before you set values for the *Begin Segment Offset* and *End Segment Offset* options, be sure to read the section, Rectangles in the Corners of Subrectangle Segments.
6. If you want to keep subrectangles out of the corners of segments, assign positive numbers for the *Begin* and *End Segment Offsets*:
   - For *Begin Segment Offset*, type a positive integer or floating-point number to begin the first subrectangle in each segment (except for the first segment), offset from the edge of the master path.
   - For *End Segment Offset*, type a positive integer or floating-point number to end the last subrectangle in each segment (except for the last segment), offset from the edge of the master path.

   For an MPP, the width of a subrectangle is parallel to the width of the master path.

© 2020 All Rights Reserved.

The length of a subrectangle is parallel to the master path centerline.

7. For *Width*, do one of the following:

- Type in a positive integer or floating-point number.
- To use the `minWidth` rule for the subrectangle layer from the technology file, keep the default.

If the `minWidth` rule is not defined in the technology file, and the subrectangle length is specified, the system uses the length. If no length is specified, you must specify a width; otherwise, the width remains zero and no subrectangles are created.

8. For *Length*, do one of the following:

- Type in a positive integer or floating-point number.
- To let the length default to the `minWidth` rule for the subrectangle layer from the technology file, do not change the *Length* field.

If the `minWidth` rule is not defined in the technology file, you must specify a length; otherwise, the length remains zero and no subrectangles are created.

For each segment, you control the space between rectangles and after the last rectangle with the two fields *Gap* and *Space*. *Space* determines the minimum distance between rectangles. *Gap* controls where the system places any space that remains after the rectangles are placed the minimum distance apart.

When the value of *Gap* is *distribute*, the system places subrectangles using the value of the *Space* option until there is no space for another subrectangle, then distributes the rest of the space as evenly as possible between the subrectangles in multiples of the grid space specified by the `mfgGridResolution` rule in the technology file. Any remaining space is placed after the last subrectangle in the segment.

© 2020 All Rights Reserved.

When the value of *Gap* is *minimum*, the system places subrectangles using the value of the *Space* option until there is no space for another rectangle, then leaves any excess space after the last subrectangle in the segment.

9. For *Space*, do one of the following:

- Type a positive integer or floating-point number for the distance between the edges of adjoining rectangles.
- To use the `minSpacing` rule for the subrectangle layer from the technology file, keep the default.

If you do not specify a value and `minSpacing` for the subpath layer is not defined in your technology file, the system sets the value to zero.

10. For *Gap*, choose *distribute* or *minimum*.

11. For *Separation* and *Justification*, type a signed integer or floating-point number for *Separation* and choose a setting for *Justification*, as shown in the table below.

Whether you enter a positive or negative number depends on where you want to define the set of subrectangles in relation to the master path.

For illustrations showing the location of sets of subrectangles in relation to the master path, see "Examples of Sets of Subrectangles" in the *Virtuoso Relative Object Design User Guide*.

Separation	Center Justification	Left Justification	Right Justification
Zero	Center of width of subrectangles on master path centerline	Left edge of master path coincident with right edge of subrectangles	Right edge of master path coincident with left edge of subrectangles
Positive number	Center of width of subrectangles on left side of master path centerline	Left edge of master path on right side of right edge of subrectangles	Right edge of master path on left side of left edge of subrectangles
Negative number	Center of width of subrectangles on right side of master path centerline	Left edge of master path on left side of right edge of subrectangles	Right edge of master path on right side of left edge of subrectangles

© 2020 All Rights Reserved.

When you specify that a set of subrectangles is a pin, each rectangle in the set becomes a pin. In the layout editor, the whole multipart path (master path plus all subparts) is treated as a single object.

12. For *Connectivity*, do one of the following:

- ❑ If you do not want to define connectivity, leave *Connectivity* set to *None*.
- ❑ If you want to associate the subpart with a net or specify it as a pin, go to Defining Connectivity for a Subpart.

13. To add the new subrectangle(s) to the scroll window at the top of the ROD Subpart form, click *Add*.

The system displays a line of data for the new set of subrectangles in the scroll window.

You **must** click *OK* or *Apply* at the top of the ROD Subpart form to make the new set of subrectangles part of the MPP.

14. To save the new set of subrectangles as part of the MPP in the cellview, click *OK* or *Apply*.

So far, you are defining a set of subrectangles only for the MPP in the current layout window. If you want to add the new set of subrectangles to an MPP template, you need to save your changes to a template.

15. To save your addition in an MPP template, go to Saving Multipart Path Values as a Template.

Note: You can also use the context sensitive menu in the *Property Editor* assistant to add a subrectangle to MPP.

You have completed adding a set of subrectangles to your MPP. Continue to the next section or return to Flow for Creating Multipart Paths to continue with the next step in the flow.

© 2020 All Rights Reserved.

Defining Connectivity for a Subpart

You can define connectivity for any MPP subpart by associating it with a net. You can also designate a subpart as a pin. The changes you make do not affect the MPP in the layout window until you click *OK* or *Apply* in the *ROD Subpart* form.

Note: You must type the name of the net in the *ROD Subpart* form for each subpart that has connectivity. If you do not type a name, the subpart is not associated with a net, and the system beeps when you click to start creating the MPP in a cellview window.

To define connectivity for a subpart, do one of the following:

1. To assign the subpart to a net, do the following:

   - For *Connectivity*, choose *Net*.

     The form expands to show the net connectivity fields.

   - For *Net Name*, type the name of the net.

   - To apply the net assignment to the subpart of the MPP in the layout window, click *OK* or *Apply*.

   You have completed assigning the subpart to a net. Return to Flow for Creating Multipart Paths to continue with the next step in the flow.

2. To assign the subpart to a pin, do the following:

   - For *Connectivity*, choose *Pin*.

     The form expands to show both the net and pin connectivity fields. For *Net Name*, type the name of the net.

     You must enter a net name for every subpart that has pin connectivity or the system beeps when you click in the cellview window to begin creating the MPP.

   - For *I/O Type*, choose one of the following: *input*, *output*, *inputOutput*, *switch*, or *jumper*.

   - For *Access Direction*, choose the part(s) of the pin to which routing can be connected.

   - Turn on *Display Terminal Name* if you want to create a text-display object to display the net name for the pin.

     Note: If pin names are not visible in the cellview, the *Pin Names* option in the *Display Options* form is not turned on. To display all pin names, turn on *Pin Names* in the *Display Options* form.

© 2020 All Rights Reserved.

The system uses the values of four fields to calculate the position of the text-display object: *Reference Handle*, *Offset X*, *Offset Y*, and the *Justification* field on the Pin Name Display form.

- Turn on *Display Terminal Name Option* if you want to set pin name display options, such as font, height, layer, and justification on the Pin Name Display form.
- For *Reference Handle*, choose where you want to align the pin name in relation to the master path or subpart.

  You can offset the pin name from the object in the direction of the X and/or Y axis.
- For *Offset X*, type a positive or negative floating-point number to offset the origin point of the pin name along the X axis from the point specified by *Reference Handle.*
- For *Offset Y*, type a positive or negative floating-point number to offset the origin point of the pin name along the Y axis from the point specified by *Reference Handle.*

  To make all the current values in the ROD Subpart form, including the connectivity values, part of the MPP in the layout window, you **must** click *OK* or *Apply*.
- To save the subpart values as part of the MPP in the layout window, in the ROD Subpart form, click *OK* or *Apply*.

The system changes the MPP in the layout window to include the subpart connectivity.

You have completed defining connectivity for the subpart. Return to the topics listed for Flow for Creating Multipart Paths to select another topic.

Editing a Subpart

The subparts of an MPP are listed at the top of the ROD Subpart form. You can change an existing subpart by changing the values for the selected subpart in the ROD Subpart form.

Remember, you must make all edits to subparts before entering the last point of your MPP in the layout window. You cannot change an MPP after clicking on its last point; instead, you must delete the whole MPP and recreate it. It is safest to save your form values as a template before drawing an MPP in your cellview. Then, if you need to edit a subpart, you can recreate the MPP from the template and use the ROD Subpart form to edit the subpart.

To edit a subpart, follow these steps.

1. In the *Create Multipart Path* form, click the *Subpart* button.

   The *ROD Subpart* form appears.

© 2020 All Rights Reserved.

Existing subparts are listed in the scroll window at the top of the form.

2. Highlight the subpart you want to edit.

   The system fills in the fields of the *ROD Subpart* form with the values of the subpart you selected.

3. Change the values in the *ROD Subpart* form as desired.

4. To make your changes affect the data listed for the subpart at the top of the ROD Subpart form, click *Edit*.

   The system changes the highlighted subpart data to match the new form values.

5. To make your subpart changes affect the MPP in the cellview, in the *ROD Subpart* form, click *OK* or *Apply*.

   So far, your changes affect only the MPP in the current layout window. If you want your changes to affect an MPP template, you need to save your changes to the template.

6. To save your changes to an MPP template, go to Saving Multipart Path Values as a Template.

You have completed editing a subpart. Return to the list of topics for the Flow for Creating Multipart Paths to select another topic.

Deleting a Subpart

You can delete any subpart of an MPP. Subparts are listed at the top of the ROD Subpart form.

Remember, you must make all deletions to subparts before entering the last point of your MPP in the layout window. You cannot change an MPP after clicking on its last point; instead, you must delete the whole MPP and recreate it. It is safest to save your form values as a template before drawing an MPP in your cellview. Then, if you need to delete a subpart, you can recreate the MPP from the template and use the ROD Subpart form to delete the subpart.

To delete a subpart, follow these steps.

1. In the *Create Multipart Path* form, click the *Subpart* button.

   The *ROD Subpart* form appears.

   Existing subparts are listed in the scroll window at the top of the form. For a description of the data shown in the window, see one of the following:

© 2020 All Rights Reserved.

2. Highlight the subpart you want to delete.
3. To delete the highlighted subpart from the data listed at the top of the ROD Subpart form, click *Delete*.

   The highlighted subpart is deleted from the list of subparts in the ROD Subpart form.
4. To delete the subpart from the MPP in the cellview, in the ROD Subpart form, click *OK* or *Apply*.

   The highlighted subpart is deleted from the MPP in the current cellview.

   So far, your changes affect only the MPP in the current layout window. If you want to remove the deleted subpart from an MPP template, you need to save the current form values to the MPP template, overwriting the old values.
5. To remove the deleted subpart from an MPP template, go to Saving Multipart Path Values as a Template.

Note: You can also use the context sensitive menu in the *Property Editor* assistant to delete a subpath.

You have completed deleting a subpart. Return to the list of topics for the Flow for Creating Multipart Paths to select another topic.

© 2020 All Rights Reserved.

Creating a Multipart Path in a Cellview

After you specify values for the fields in the Create Multipart Path and ROD Subpart forms or choose an MPP template, you are ready to create an MPP in your layout window.

About Connectivity

MPP templates do not contain net names because each MPP you create from the same template could be on a different net.

When you choose an MPP template that has pin connectivity defined, the system displays a message at the bottom of the cellview window prompting you to enter a valid net name. If the MPP template does not have pin connectivity, and you want to associate the MPP with a net, type the net name in the *Net Name* field for the master path or for any subpart, as desired.

Note: When there is pin connectivity, you must enter the name of the net in the *Net Name* field for each part (master path and/or subparts) before you begin to create the MPP in the layout window. If you click in the layout window before entering the net name(s), the system beeps.

As you draw an MPP, you can change the values in the *Create Multipart Path* and *ROD Subpart* forms and see immediate changes to the MPP, until you double-click to enter the last point. After you enter the last point to complete an MPP, you can no longer change that MPP using the Create forms. You can, however, use the *Edit MPP Path Properties* form to change a completed MPP.

To draw an MPP in the layout window, follow these steps.

1. Click to enter the first point.
2. Move the pointer to the next point and click.
3. Continue to move the pointer and click to enter points.
4. To enter the last point of the path, either double-click or press `Return`.

You have completed drawing the MPP in your layout window. Return to the topics listed for the Creating and Editing Multipart Paths to select another topic.

© 2020 All Rights Reserved.

Saving Multipart Path Values as a Template

Whenever you create a new type of MPP that you might want to use again, you can save the values currently in the *Create Multipart Path* and *ROD Subpart* forms as a template with a unique name. This lets you create the same or similar MPPs. You can save MPP templates in two ways:

- To an ASCII file in your local hierarchy
- To the temporary technology file in virtual memory
  If you have write permission, you can also save to your technology library on disk.

Saving Templates to an ASCII File

Remember that whenever you load MPP templates from ASCII files, they overwrite the templates in the technology file in virtual memory that have matching names. Later, when you save changes to your technology library on disk, the templates in virtual memory overwrite templates with matching names in the technology library on disk. If you do not want this to happen, see "Problem: Warning in CIW: MPP template name is replacing an existing MPP template" in the *Virtuoso Relative Object Design User Guide*.

When you save to an ASCII file, you can enter a new file name or overwrite an existing file. The system does not append to an existing ASCII file.

To save to one or all templates to an ASCII file, follow these steps.

1. In the *Create Multipart Path* form, click the *Save Template* button.

   The *Save Multipart Path Templates* form appears.

2. For *Template List*, choose the name of the MPP template you want to save or choose *All Templates* to save every template in the list.

3. Turn on *ASCII File*.

   If you use the name of an existing ASCII file, it will be overwritten.

4. Type the ASCII file name (or path and file name) or keep the default.

   If you want to designate a particular value as the default for the *ASCII File* field, you can set the `mppASCIIFileName` layout editor environment variable. For information about how to set this environment variable, see Setting the mppASCIIFileName Environment Variable.

5. Click *OK*.

   If the system overwrote an existing ASCII file, the following message appears in the CIW:

© 2020 All Rights Reserved.

```
Template templateName is replacing an exisitng template by the same name.
```

When you save an MPP to an ASCII file, the system also updates the temporary technology file in virtual memory.

Saving Templates to Your Technology Library

The Save Multipart Path Templates form allows you to save the template definitions to an ASCII file or directly to the technology library you select from the Technology Library cyclic field.

If you do not have write permission to the technology library, and you choose the Technology Library option, the field is grayed out and will not allow you to save the templates. Choose the ASCII File option instead.

To save to one or all templates to the temporary technology file in virtual memory, follow these steps.

1. In the *Create Multipart Path* form, click the *Save Template* button.

   The *Save Multipart Path Templates* form appears.

2. For *Template List*, choose the name of the MPP template you want to save or choose *All Templates* to save every template in the list.

3. Select the *Technology Library* radio button.

4. Click *OK*.

   To make the template part of your binary technology library on disk, you must save your technology library changes before you exit the editing session.

Saving to Your Technology Library on Disk Now

If you have not saved the changes to your binary technology library during your editing session, the system displays a dialog box when you exit the software asking if you want to save the changes. You can save your template changes to disk then or you can save them to disk now.

To save the changes to your binary technology library to disk now, follow these steps.

1. In the CIW, choose *Tools – Technology File Manager*.

   The *Technology Tool Box* appears.

2. Click *Save*.

© 2020 All Rights Reserved.

The *Save Technology File* form appears.

3. Choose the name of the library you want to save and click *OK*.

   The system displays a dialog box asking you to confirm that you want to save the specified technology library to disk.

4. In the *Technology File Save* form, click *Yes*.

You have completed saving the template (or all templates). Return to the topics listed for Creating and Editing Multipart Paths to select another topic.

© 2020 All Rights Reserved.

Loading an ASCII Multipart Path Template File

You can access MPP templates that were saved in ASCII files by loading the ASCII files from the *Create Multipart Path* form. The templates in each file you load are added to the MPP templates in the technology file in virtual memory and added to the list of template names in the *MPP Template* field on the *Create Multipart Path* form.

Important

When you load an ASCII MPP template file, the system checks the names of the ASCII MPP templates against the names of MPP templates that already exist in the technology file in virtual memory. If there is a name conflict, the system displays a message in the CIW saying that an MPP template in the ASCII file is replacing an existing MPP template. The existing template is overwritten in virtual memory only. If you do not want to overwrite the original MPP template in your binary technology library on disk, do not save changes to the technology file. You can also avoid overwriting existing templates by changing the duplicate names in your ASCII file. For more information about what to do, see "Problem: Warning in CIW: MPP template name is replacing an existing MPP template" in the *Virtuoso Relative Object Design User Guide*.

To load an ASCII MPP template file,

1. In the *Create Multipart Path* form, click *Load Template*.

   The *Load Multipart Path Template* form appears.

2. In the *Template File* field, type the path and file name for the ASCII MPP Template file you want to load and click *OK*.

   If you enter an invalid technology file name, the system displays a warning message in the CIW and no templates are loaded. Use the *Browse* button if you need assistance locating the template file name.

The system does not provide a default for the *Template File* field on the *Load Multipart Path Template* form. However, you can designate a file name or path and file name as the default for the *Template File* field on the Load Multipart Path Template form by setting the `mppASCIIFilename` layout editor environment variable. For information about how to set this environment variable, see Setting the mppASCIIFileName Environment Variable.

You have completed loading templates. Return to Defining the Master Path or to the topics listed for Creating and Editing Multipart Paths to select another topic.

© 2020 All Rights Reserved.

Creating an MPP Guard Ring

The tasks involved in creating an MPP guard ring include:

- Load a technology file into your local directory.
- Create an MPP template.

Note: An MPP guard ring template must contain at least one sub rectangle, and it should not have any connectivity in it.

- Use the MPP template in the *Guard Ring* form and fine-tune other settings.

The first two tasks are explained in detail in previous topics. To create an MPP guard ring,

1. Choose *Create – MPP Guard Ring.*

   The Guard Ring Form appears.

2. Select the MPP template you want to use for guard ring creation in the *Guard Ring Template* cyclic field.
3. Select the *Technology Library* in which the selected template is defined.
4. Specify the connectivity information for the MPP guard ring in the *Net Name* field.
5. Adjust the value in the *Enclose by* field to set the distance between the MPP guard ring and the selected object(s).
6. Select *Rectangular* or *Rectilinear* shape for the MPP guard ring.
7. In the layout window, select the object or objects around which the MPP guard ring will be created. A preview of the guard ring appears around the selected objects.
8. Click *Apply* to place the MPP guard ring around the selected objects.

© 2020 All Rights Reserved.

Setting Multipart Path Environment Variables

You can set two environment variables related to multipart paths:

- `mppTemplate`
- `mppASCIIFileName`

Setting the mppTemplate Environment Variable

You can control the default values for the fields in the *Create Multipart Path* form the first time the form is opened during an editing session by setting the layout editor environment variable mppTemplate.

The system default for `mppTemplate` is *New*. When you open the *Create Multipart Path* form, the system sets the *Template Name* and *MPP Template* fields to *New* and the remaining fields to their system default values. Whenever you click *New* in the *MPP Template* field, the system resets all fields in the *Create Multipart Path* and *ROD Subpart* forms to the system defaults.

To preset the *Template Name* and *MPP Template* fields to the name of an MPP template you have already saved, set `mppTemplate` to the template name. Then when you open the *Create Multipart Path* form for the first time during an editing session, the system sets the *Template Name* and *MPP Template* fields to the name of your template and uses the template values as defaults for the remaining fields.

When you choose a different value for the *MPP Template* field in the *Create Multipart Path* form, the system automatically resets the values of all the fields in the *Create Multipart Path* and *ROD Subpart* forms and updates the `mppTemplate` environment variable to match the template name you chose. For example,

- If the *MPP Template* field is set to *New*, which is the default, and you change it to `guardring`, the system resets all form fields to the values defined for the `guardring` template and updates the `mppTemplate` environment variable to `guardring`.
- Conversely, if the value of the *MPP Template* field is set to `guardring` and you change it to *New*, the system resets all fields to the system defaults and updates the `mppTemplate` environment variable to *New*.

The following sections describe how to check the current value of `mppTemplate` and how to set it.

© 2020 All Rights Reserved.

Checking the Value of mppTemplate

To determine the current value of the layout editor `mppTemplate` environment variable, follow these steps.

➤ In the CIW, type

```
envGetVal( "layout" "mppTemplate" )
```

Changing the Value of mppTemplate

➤ To change the setting of the `mppTemplate` environment variable in the CIW or in your local `~/.cdsenv` file, do one of the following:

- To change the `mppTemplate` environment variable temporarily, type the following in the CIW:

  ```
  envSetVal("layout" "mppTemplate" 'string "your_value")
  ```

  where *`your_value`* is the new value for `mppTemplate`.

- To change the value of the `mppTemplate` environment variable in your `~/.cdsenv` file, edit the file, search for `mppTemplate`, and change the value as desired.

  The default layout environment variable setting for `mppTemplate` in you `~/.cdsenv` file looks like this:

  ```
  layout   mppTemplate   string   "New"
  ```

Setting the mppASCIIFileName Environment Variable

You can set the layout editor environment variable `mppASCIIFileName` to control default values for the first time you open the following forms during an editing session:

- The *ASCII File* field in the *Save Multipart Path Templates* form
- The *Template File* field in the *Load Multipart Path Template* form

You can set this variable to a filename or to a path and filename. When you do not specify a path, the system uses your current directory.

The system lets you set `mppASCIIFileName` to a filename that does not exist yet. When you save a template to an ASCII file, the system creates the file. However, if you try to load before the file is created, the system displays a warning.

When the `mppASCIIFileName` environment variable is undefined or set to `nil`,

© 2020 All Rights Reserved.

- For the *ASCII File* field in the *Save Multipart Path Templates* form, the first time you open the form, the system supplies a default value that is based on your technology library name followed by

  `.mpp.templates.il`

  For example,

  `myTechLib.mpp.templates.il`

  You can change the *ASCII File* field by typing in it. If you do not type the `.il` suffix, the system appends it.

- For the *Load Multipart Path Template* form, the system does not supply a default value.

When you set the `mppASCIIFileName` environment variable to a specific filename or path and filename, the system supplies your value as the default for both the *ASCII File* field in the Save Multipart Path Templates form and the *Template File* field in the Load Multipart Path Template form.

When you change the value of the *ASCII File* field in the *Save Multipart Path Templates* form, the system automatically resets the value of the *Template File* field in the *Load Multipart Path Template* form to match and also updates the `mppASCIIFileName` environment variable to match. However, if you change the *Template File* field in the *Load Multipart Path Template* form, the system does not change the value of the *ASCII File* field in the *Save Multipart Path Templates* form and does not change the `mppASCIIFileName` environment variable.

The following sections describe how to check the current value of `mppASCIIFileName` and how to set it.

Checking the Value of mppASCIIFileName

To determine the current value of the layout editor `mppASCIIFileName` environment variable, follow these steps.

➤ In the CIW, type

```
envGetVal( "layout" "mppASCIIFileName" )
```

Changing the Value of mppASCIIFileName

➤ To change the setting of the `mppASCIIFileName` environment variable in the CIW or in your local `~/.cdsenv` file, do one of the following:

© 2020 All Rights Reserved.

- To change the `mppASCIIFileName` environment variable temporarily, type the following in the CIW:

  ```
  envSetVal("layout" "mppASCIIFileName" 'string "your_value.il")
  ```

  where *`your_value`* is the new value for `mppASCIIFileName`.

- To change the value of the `mppASCIIFileName` environment variable in your `~/.cdsenv` file, edit the file, search for `mppASCIIFileName`, and change the value as desired.

  The default layout environment variable setting for `mppASCIIFileName` in your `~/.cdsenv` file looks like this:

  ```
  layout    mppASCIIFileName    string     "techLibName.mpp.templates.il"
  ```

Selecting Multipart Paths

A multipart path is a single object composed of multiple parts on the same or different layers. The parts are a single master path and one or more subparts. The master path is the primary part; subparts exist in relation to the master path. A subpart can be an offset subpath, an enclosure subpath, or a set of subrectangles. Individual subrectangles are not individual shapes but are part of a specific set of subrectangles.

The multipart path below has one subpath and one set of subrectangles. Both the subpath and the set of subrectangles are offset from the master path.

Selecting multipart paths is similar to selecting regular paths: you point and click or create a selection box in a similar manner. In general, you cannot select subparts separately from the master path; however, you can select the chopped end of subpaths to stretch them.

© 2020 All Rights Reserved.

- In full selection mode, when you select any part of a multipart path, the whole multipart path is selected. The master path is highlighted on the current selection layer, while subparts are highlighted on a different layer. This lets you see which part is the master path and which parts are subparts.
- In partial selection mode (press F4), you can select an end, segment, or vertex of the master path.
- In partial selection mode, you can select the chopped end(s) of one or more subpath(s) and stretch them. When you stretch a chopped subpath, all other choppable subparts are stretched also.

Note: You cannot directly select a set of subrectangles, but selecting and stretching the chopped end of a subpath causes all chopped parts in the multipart path to be regenerated, including chopped sets of subrectangles. The system regenerates subrectangles along orthogonal segments only.

Selecting a Whole Multipart Path

In full selection mode, selecting multipart paths is similar to selecting regular paths. When you select any part of a multipart path, the whole multipart path is selected. The master path is highlighted on the current selection layer, while subparts are highlighted on a different layer. This lets you see which part is the master path and which parts are subparts.

To select a whole multipart path,

➤ Do one of the following:

- ❑ In full selection mode, point to any part of the multipart path and click.

© 2020 All Rights Reserved.

As you point, the whole path is highlighted with dotted lines.

Point to any part of the multipart path and click.

Highlighting shows the whole multipart path is selected.

- In full selection mode, create a selection box around any part of the multipart path.

Create a selection box around any part of the multipart path.

Highlighting shows the whole multipart path is selected.

Selecting Multipart Path Ends

In partial selection mode, you can select an end of a master path in a multipart path. You cannot select the end of a subpart, except for chopped subpath ends. If you attempt to select the non-chopped end of a subpart, the nearest end of the master path is selected.

To select an end of a master path,

➤ Do one of the following:

© 2020 All Rights Reserved.

- In partial selection mode, point to an end of the master path or an end of a subpath and click.

  As you point, the nearest end of the master path is highlighted with dotted lines.

Point to an end and click.

Highlighting shows the (nearest) master path end is selected.

- In partial selection mode, create a selection box around an end of the master path or around an end of a subpath.

Create a selection box around an end of the master path or subpath.

Highlighting shows the (nearest) master path end is selected.

Note: Even in partial selection mode, if you select one or more whole subrectangles, the whole MPP is selected.

Selecting Chopped Subpath Ends in Multipart Paths

In partial selection mode, you can select the end(s) of one or more chopped subpaths in a multipart path and stretch them.

© 2020 All Rights Reserved.

Note: You cannot directly select a set of subrectangles, but selecting and stretching the chopped end of a subpath causes all chopped parts in the multipart path to be regenerated, including chopped sets of subrectangles. The system regenerates subrectangles along orthogonal segments only.

For example, to select the end of one chopped subpath,

➤ Do one of the following:

- In partial selection mode, point to a chopped end and click.

  As you point, the chopped end is highlighted with dotted lines.

Point to a chopped subpath end and click.

Highlighting shows the chopped subpath end is selected.

- In partial selection mode, create a selection box around the chopped end of a subpath.

Create a selection box around a chopped end of a subpath.

Highlighting shows the chopped subpath end is selected.

© 2020 All Rights Reserved.

Selecting Multipart Path Segments

In partial selection mode, you can select one or more segments of the master path in a multipart path. You cannot select segments of a subpart. If you attempt to select a subpart segment, the nearest segment of the master path is selected.

Note: In partial selection mode, if you click the edge of a subrectangle, the nearest master path segment is selected.

To select a single master path segment,

➤ Do one of the following:

- In partial selection mode, point to the middle of the centerline in a master path segment or subpath segment and click.

  As you point, the nearest master path segment is highlighted with dotted lines.

Point to the centerline in a path segment and click.

Highlighting shows the (nearest) master path segment is selected.

© 2020 All Rights Reserved.

- In partial selection mode, create a selection box around a master path or subpath segment. Make sure both end points of the segment centerline are inside the box.

Create a selection box around the centerline of a path segment.

Highlighting shows the (nearest) master path segment is selected.

To select more than one master path segment,

➤ Do one of the following:

- In partial selection mode, create a selection box around multiple master path or subpath segments. Make sure the end points of each segment centerline are inside the box.

Create a selection box around the centerline of multiple segments.

Highlighting shows the (nearest) master path segments are selected.

© 2020 All Rights Reserved.

- In partial selection mode, point to the middle of the centerline in a master path or subpath segment and click. Press `Shift` and click to select additional segments.

As you point, the centerline of the nearest master path segment is highlighted with dotted lines.

Point to a segment centerline and click, then press `Shift` and click to select another segment.

Highlighting shows the path segments are selected.

Selecting a Multipart Path Vertex

For multipart paths, a vertex is a point on the master path centerline where two segments join. You can select a vertex of the master path in a multipart path. To do so, you either select the master path vertex directly or select the nearest vertex of a subpath.

To select a vertex of the master path,

➤ Do one of the following:

- In partial selection mode, point to a vertex on the centerline of the master path or centerline of a subpath and click.

© 2020 All Rights Reserved.

As you point, the nearest master path vertex is highlighted with dotted lines.

Point to a vertex on the master path or subpath and click.

Highlighting shows the (nearest) master path vertex is selected.

If you are not able to click a centerline vertex, the current setting of the *Gravity* option is too restrictive. In the Layout Editor Options form, turn off *Gravity*.

- In partial selection mode, create a selection box around a vertex on the centerline of a master path or subpath.

Create a selection box around a vertex of the master path or a subpath.

Highlighting shows the (nearest) master path vertex is selected.

Note: If you click the corner of a subrectangle, the nearest master path vertex is selected.

For more information about multipart paths, see Multipart Paths in the *Virtuoso Relative Object Design User Guide* and the `rodCreatePath` function in the *Virtuoso Relative Object Design SKILL Reference*.

© 2020 All Rights Reserved.

Moving Multipart Paths

When you move a multipart path within a cellview, its master path and subparts move together. You cannot move the master path separately from its subparts, nor can you move subparts separately from the master path.

When you move a multipart path that has objects aligned to it, the objects move to stay in alignment with the multipart path.

1. Choose *Edit – Move* [m].
2. In full selection mode, choose the multipart path by clicking on the master path or any subpart.

3. Click to place the multipart path; the aligned objects follow.

The multipart path and all objects aligned to it appear in the new location.

How Moving Affects Multipart Paths with Aligned Objects

When you move a multipart path, any objects aligned to the multipart path move with it. To retain alignments when you move a multipart path to a new cellview, all objects aligned with the multipart path must be part of the selected set. If aligned objects are not selected, they will not be moved and the alignments will be broken.

In the example below, the multipart path `path1` is the reference object. Via `B` in instance `nTrans1` is aligned to `path1`, with a separation of -10 units in the direction of the X axis.

© 2020 All Rights Reserved.

Therefore, `nTrans1` behaves as if it is aligned to `path1`. When you move `path1`, `nTrans1` moves with it.

The system automatically recalculates the alignment of objects when you open a layout cellview in edit mode or edit an aligned object in any way.

For an overview of relative object design alignment, see Aligning Named Objects in the *Virtuoso Relative Object Design User Guide*.

For multipart paths, see Multipart Paths and the `rodCreatePath` function in the *Virtuoso Relative Object Design User Guide*.

© 2020 All Rights Reserved.

Stretching Multipart Paths

You can stretch the ends, segments, and/or corners of a multipart path in the same way you stretch single-part paths; the master path and its subparts stretch together. You cannot stretch the master path separately from its subparts, nor can you stretch subparts (except chopped subpaths) separately from the master path.

If you attempt to stretch a subpart, the system applies the stretch to the whole multipart path. The only exception is that you can stretch the chopped ends of subpaths. When you stretch a chopped subpath, all other chopped subparts are stretched also.

Note: You cannot directly select a set of subrectangles, but selecting and stretching the chopped end of a subpath causes all chopped parts in the multipart path to be regenerated, including chopped sets of subrectangles. The system regenerates subrectangles along orthogonal segments only.

To stretch one segment in a multipart path,

1. Choose *Edit – Stretch* [s].

   Stretch automatically changes to partial selection mode and prompts you to select the object you want to stretch.

2. Select the segment you want to stretch by doing one of the following:

   - Click the middle of the centerline of the master path segment or on the comparable subpath segment.

The nearest master path segment is selected.

© 2020 All Rights Reserved.

- ❑ Create a selection box around the centerline of a segment.

Be sure to include the ends of the segment centerline.

When you use a selection box, *Stretch* prompts you for a reference point.

3. If you are prompted for a reference point, click where you want the stretch to begin.

The reference point does not have to be directly on the multipart path.

Stretch prompts you to point to the new location.

4. Point to a new location for the stretch.

© 2020 All Rights Reserved.

As you point, an outline of the master path segment shows you the new location.

5. Click to select the new location for the selected multipart path segment.

Stretching Chopped Subpath Ends in Multipart Paths

You can select and stretch the ends of chopped subpaths in a multipart path. You might want to do this to reconnect the ends. When you stretch the chopped end(s) of one or more subpaths, all chopped subparts stretch with the selected subpath(s).

Note: You cannot directly select a set of subrectangles, but selecting and stretching the chopped end of a subpath causes all chopped parts in the multipart path to be regenerated, including chopped sets of subrectangles. The system regenerates subrectangles along orthogonal segments only.

To stretch the chopped end of a subpath,

1. Choose *Edit – Stretch* [s].

   Stretch automatically changes to partial selection mode and prompts you to select the object you want to stretch.

© 2020 All Rights Reserved.

2. Select a chopped subpath end with a selection box or by clicking on it.

When you use a selection box, *Stretch* prompts you for a reference point.

3. If you are prompted for a reference point, click where you want the stretch to begin.

   The reference point does not have to be directly on the path end.

   Stretch prompts you to point to the new location.

4. Point to a new location for the stretch.

   As you point, an outline shows you the new location of the subpath end.

© 2020 All Rights Reserved.

5. Click to select the new location for the chopped subpath end.

You can reconnect chopped subparts by stretching one subpath end over to the other subpath end. Even if you overlap the ends, the system regenerates the subrectangles correctly.

6. Select the same chopped subpath end, and point to select the new location so that it overlaps the other chopped subpath end.

As you point, an outline shows you the new location of the subpath end.

© 2020 All Rights Reserved.

The system reconnects the subpaths and regenerates subrectangles to fill the reconnected segment.

Splitting Multipart Paths

You can add a jog to a section of a multipart path in the same way you add a jog to single-part paths; the jog affects the master path and all of its subparts. You might want to split and stretch a multipart path to direct it around an object or to add new connecting segments.

Any objects aligned with or attached to the part of the multipart path that moves, move with the stretch. You can split paths only in full selection mode.

Note: You cannot split or stretch the master path separately from its subparts, nor can you split or stretch subparts separately from the master path. If you attempt to split a subpart, the system applies the split to the whole multipart path.

To stretch a section of a multipart path segment,

1. Choose *Edit – Advanced – Split* [`Control-s`].

© 2020 All Rights Reserved.

2. Select the object to be split by clicking anywhere on the multipart path.

Split prompts you to draw a split line by clicking on points. To define the section you want to stretch, your split line must cross the centerline of the master path in two places.

3. To create a split line, click four points as shown below, then press `Return` to end the split line.

Split prompts you for a reference point.

© 2020 All Rights Reserved.

4. Click a starting point for the stretch.

Split prompts you for the new location for the stretch.

5. Point to a new location for the stretch.

An outline of the segment shows you the position of the stretch.

© 2020 All Rights Reserved.

6. Click to select the new location for the section being stretched.

Reshaping a Multipart Path

You can reshape a segment of a multipart path in the same way you reshape a segment of a single-part path; however, you must start the reshaped segment on the master path centerline.

The master path and its subparts reshape together. You cannot reshape the master path separately from its subparts, nor can you reshape subparts separately from the master path.

To reshape one segment in a multipart path,

1. Choose *Edit – Advanced – Reshape* [`Shift-r`].

   The Reshape Form appears.

2. Select a multipart path.

   You are prompted to enter the first point.

3. Turn on *Reshape Type: line*.

© 2020 All Rights Reserved.

4. Click the centerline of the master path to enter the first point.

As you move the pointer, an outline of the master path shows the new shape.

5. Double-click when you are finished entering points.

6. Click right to toggle between highlighting the new path and highlighting both the old and new path.

© 2020 All Rights Reserved.

7. When the shape you want is highlighted, click.

For multipart paths, see Multipart Paths and the `rodCreatePath` function in the *Virtuoso Relative Object Design User Guide*.

How Chopping Affects Multipart Paths

When you use *Chop* on a multipart path, the result depends on what parts of the multipart path are choppable. You specify whether a part is choppable when you create the multipart path.

- If you specify the master path as choppable, all of its subparts must be choppable also. When you specify the master path as not choppable, you can specify each of its subparts as choppable or not.
- If the master path is choppable, you can chop the whole multipart path into two or more separate multipart paths by chopping all the way through the master path at 90 degrees.
- If the master path is choppable and you chop at an angle other than 90 degrees and/or chop only part of the way through the master path, the chop changes all objects in the multipart path into ordinary paths, polygons, and rectangles. If this is not what you wanted, you can undo the chop with the *Edit – Undo* command.
- If the master path is not choppable, you can chop all subparts that are specified as choppable by chopping all the way through the master path at 90 degrees. All choppable subparts are chopped where you chop over the master path.
- When a chop affects sets of subrectangles that are choppable, the system regenerates subrectangles along orthogonal segments only.

© 2020 All Rights Reserved.

- If the master path is not choppable, the system will not allow you to chop at an angle other than 90 degrees or to chop only part of the way through the multipart path.

After you chop a multipart path, you can select and stretch the chopped ends of subpaths. Although you cannot directly select, stretch, or chop a set of subrectangles, all choppable sets of subrectangles are also chopped when you chop a subpath.

When you chop a multipart path into one or more separate multipart paths, the system assigns the name of the original multipart path to the first new multipart path. The system assigns unique names to the other new multipart paths, starting with `path0`, `path1`, etc.

Note: If you want to chop and rotate an MPP, rotate the MPP first, then chop it. Avoid rotating a chopped MPP as the chop holes might not rotate correctly.

Chopping Multipart Paths with Aligned Objects

When you chop through the (choppable) master path of a multipart path, alignments to other objects are lost.

The following example shows what happens to an aligned object after cutting away a section of a choppable master path.

© 2020 All Rights Reserved.

When you use the cutter to chop out part of the master path like this,

the result is two new, shorter multipart paths, neither of which are aligned to `rect1`.

The system keeps the name `pathA` with the first new multipart path and assigns a unique name in the format of `path`*n* to the second new multipart path.

Chopping a Multipart Path

The multipart path shown below has one subpath and one set of subrectangles.

© 2020 All Rights Reserved.

To cut away part of a multipart path,

1. Choose *Edit – Basic – Chop* [`Shift-c`].
2. Select the multipart path.
3. Create a chop rectangle around the master path to identify the parts you want to cut away.

To chop a multipart path or its choppable subparts, you must chop all the way through the master path. It is not necessary to chop through subparts.

The results of the chop depend on whether the master path is choppable or not, as shown below.

For multipart paths, see Multipart Paths and the `rodCreatePath` function in the *Virtuoso Relative Object Design User Guide*.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

9

Creating and Editing Fluid Guard Rings

For information about Creating and Editing Fluid Guard Rings, refer to the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

10

Editing and Defining Properties

This chapter contains these topics:

- Understanding Properties
- Editing Object Attributes
- Editing Object Connectivity
- Editing Cut Patterns
- Editing Multipart Paths
- Using Net Expressions and Inherited Connections

© 2020 All Rights Reserved.

Understanding Properties

The *Edit - Basic - Properties* command lets you edit the information that defines selected objects. Every object in your database has information associated with it. The Edit Properties Form displays this information, dividing it into the following categories:

- *Attribute* define the object. Each type of object has specific, built-in attributes. For example, a polygon always has a layer and a set of coordinates that define vertices. You can view and change attributes of an object.
- *Connectivity* shows information on how this object relates to a net and/or a terminal and associated net expressions.
- *Parameter* control values associated with a cell. Only parameterized cells (Pcells) can have parameters.
- *Property* is user-created information that further defines the object. However, some applications also add properties.
- *ROD* displays the relative object design (ROD) name, handle, and alignment information of the object or instance.

About Editing Properties

The *Edit Properties* form is a multipurpose, multi-tool form that is used to edit the properties of whatever you create. You can change the size of the *Edit Properties* form by using the environment variables propEditorWidth and propEditorHeight.

To open the *Edit Properties* form,

➤ Choose *Edit – Basic – Properties* [Q] before or after selecting an object.

The title and contents of the form vary with the object you select.

Displaying Properties

To display the properties of an object,

© 2020 All Rights Reserved.

1. Choose *Edit – Basic – Properties* [Q] or click the *Edit Properties* icon on the *Edit* toolbar.

   Alternatively, right-click at any free space in the design canvas to display the *Layout* context menu and select *Properties.*

2. Select one or more objects.

   The Edit Properties Form opens.

3. Click the appropriate tab in the form to see attributes, connectivity, parameters, or user-defined properties.

4. Click the appropriate icons on the form to perform actions on objects. You have the option to hide or show the selection tree, deselect highlighted ones from canvas, keep

© 2020 All Rights Reserved.

highlighted ones and deselect others from canvas, previous object, next object, and select all objects in the tree by selecting the appropriate icons.

Show/Hide Selection Tree: Lets you show or hide the selection tree

Deselect highlighted ones from canvas: Deselects the highlighted object from the canvas

Keep highlighted ones and deselect others from canvas: Retains the highlighted objects and deselects the other objects from the canvas

Previous: Selects the previous object in the selection tree

Next: Selects the next object in the selection tree

Select all objects in the tree: Selects all objects in the selection tree

Note: The number of objects selected is also displayed on the form 8/26 .

5. Click *Cancel* to close the form.

© 2020 All Rights Reserved.

Displaying Context-Sensitive Menu

Right-clicking in the selection tree to display context-sensitive menu in the Edit Properties form. The context-sensitive menu offers the following options:

Select All/Clear Selection: Selects/deselects all the objects in the selection tree.

Deselect: Deselects the highlighted object from the canvas.

Keep: Retains the highlighted objects and deselects the other objects from the canvas.

Zoom To Selected: Zooms to the selected object in the canvas.

Highlight Modified Fields: Highlights the modified fields in the Edit Properties form.

Dynamic Cross Highlight: Dynamically highlights the objects on the canvas.

Collapse All: Collapses all expanded tree nodes in the selection tree.

Expand Selected: Expands th selected node in the selection tree.

Note: You can use the Up and Down arrow keys on the keyboard to go to the next or previous object in the selection tree.

Editing the Properties of an Object

To edit the properties of one object,

1. Select the object.
2. Choose *Edit – Basic – Properties* [Q].

© 2020 All Rights Reserved.

The Edit Properties Form opens.

3. Type or select new values for the object.
4. Click *OK*.

The object is changed to show the new values.

The edited fields are highlighted in the Edit Properties form.

Note: If the values of the non-editable fields are updated because of the change in value of any editable field, such fields are highlighted in a darker shade of blue and gray color.

© 2020
All Rights Reserved.

Note: It will help in distinguishing the fields which are modified and non-modified due to changes.

Note: The *Parameter* tab shows the parameter values associated with parameterized cells (Pcells). For CDF instances, you can display the prompt names or CDF parameter names by selecting the option *Prompt Name* or *Display CDF Parameter Name*. By default, the prompt name is displayed for CDF instances.

Editing Properties of a Group of Objects

To edit properties of a group of objects,

1. Select all the objects you want to edit.
2. Choose *Edit – Basic – Properties* [Q].

   The Edit Properties Form opens.
3. Click the *Select all objects in the tree* icon or the *Select All* option from the contextual menu in the selection tree.

   The values for all properties common to the selected objects appear. If a property does not apply to all or is different for some of the selected objects, its value appears as *AS IS*, meaning this property cannot be changed on any of the selected objects.
4. Type or select new values for the objects.
5. Click *OK*.

Searching For and Replacing Properties

You can search for one or more objects with identical properties and replace those properties with a new value. This way you can quickly change a large number of objects.

© 2020 All Rights Reserved.

To search for objects and replace their properties,

1. Choose *Tools* – [`Shift+S`].
2. Use the Find/Replace Form to choose the object you want to search for and to set any search criteria.
3. In the *Replace* cyclic field, choose the type of property you want to replace.
4. Type or select the new value.

5. Click *Apply*.

   The layout editor highlights all the objects it finds. All the objects are now in a search group. The first object in the group is highlighted in a different color.
6. Click *Previous* or *Next* to select different objects in the search group.
7. To change the properties, do one of the following:
   - Click *Replace* to change the property for the first object in the search stack.
   - Click *Replace All* to change properties for all of the highlighted objects.
8. When you are finished changing objects, click *Cancel*.

Defining a New Cellview Property

A cellview property is information that you or a Cadence application assigns to the cellview. This information can be used to override default information or to add information for further processing by other applications. For example, when postprocessing cells, you could have a SKILL routine use a property to identify which cells to process.

To define a new cellview property,

1. Choose *File – Properties* [`Shift+Q`].

   The Edit Cellview Properties Form opens.
2. Click *Property*.

© 2020 All Rights Reserved.

The properties of the cellview appear. Most of these properties correspond to settings in the Display Options Form.

3. Click *Add*.

   The *Add Property* form appears.

4. Fill out the Add Property Form.
5. Click *OK*.

   Your new property appears at the bottom of the *Edit Cellview Properties* form.

Defining a New Object Property

An object property is information that you assign to a via, circle, donut, ellipse, instance, label, path, pin, polygon, or rectangle. This information can be used to identify an object or to add information for further processing by other applications.

To define a new object property,

1. Choose *Edit – Basic – Properties* [Q].
2. Select one or more objects.

   The Edit Properties Form for the first selected object appears.

3. Click *Property*.

   The properties of the object appear.

4. Click *Add*.

   The *Add Property* form appears.

5. Fill out the Add Property Form.
6. Click *OK*.

   Your new property appears at the bottom of the *Edit Properties* form.

Modifying a Property

To change a user-created property,

1. Click *Modify* in the Edit Cellview Properties Form or the Edit Properties Form.

   The Modify Property Form appears.

© 2020 All Rights Reserved.

2. Do one of the following:
   - If you want to change a cellview property, choose *File – Properties* [`Shift-q`].
   - If you want to change an object property, select the object and then choose *Edit – Basic – Properties* [`Q`].

   The Edit Cellview Properties Form or the Edit Properties Form opens.
3. Click *Property*.

   The properties of the object or cellview appear.
4. Click the property label you want to change.
5. Click *Modify*.

   The Modify Property form appears.
6. In the Modify Property Form, make your changes.
7. Click *OK*.

Deleting a Property

To delete a property,

1. Do one of the following:
   - If you want to delete a cellview property, choose *File – Properties* [`Shift+Q`].
   - If you want to delete an object property, select the object and then choose *Edit – Basic – Properties* [`Q`].

   The Edit Cellview Properties Form or the Edit Properties Form opens.
2. Click *Property*.

   The properties of the object or cellview appear.
3. Click the property label you want to delete.
4. Click *Delete*.

   The property is deleted from the form.

© 2020 All Rights Reserved.

Editing Object Attributes

The characteristics that define an object are known as the object's attributes. All design objects have default attributes, such as color.

To change the attributes of an object,

1. Choose *Edit – Basic – Properties* [Q].
2. Select the object.

   The *Edit Properties* form for that object appears.
3. In the *Edit Properties* form, select the *Attribute* tab.
4. Change any of the attributes. See Attribute Types for Objects.
5. Click *Apply*.

Editing Object Connectivity

The connectivity information defines how an object relates to a net and/or a terminal and the associated net expressions.

To change the connectivity of an object,

1. Choose *Edit – Basic – Properties* [Q].
2. Select the object.

   The *Edit Properties* form for that object appears.
3. In the *Edit Properties* form, select the *Connectivity* tab.
4. Specify the *Net Name* and *Net Criticality* information.
5. Click *Apply*.

Note: If you select multiple instances on the canvas, then you can set the net on a particular terminal for all the selected instances.

Editing Cut Patterns

To edit a cut pattern,

© 2020 All Rights Reserved.

1. Select a via array of five rows and eight columns, as an example.

2. Choose *Edit – Basic – Properties*.

   This opens the *Edit Via Properties* form.

3. In the *Cut* section, click the *Edit Cut Pattern* button.

   The Edit Cut Pattern Form appears.

4. Select *Wall Paper*.

5. Use the *Paper Size* left and down arrows to reduce the rows and columns to three and three.

6. Edit cut by cut by clicking on the squares to turn off the top row and the far right column, as shown below:

7. Click *OK*.

8. On the *Edit Via Properties* form, click *Apply*.

© 2020
All Rights Reserved.

The pattern then repeats across the entire cut array, starting from the lower left corner of the array and repeating upwards and to the right.

The following cut pattern is the result.

Editing Multipart Paths

A multipart path (MPP) is a single ROD object consisting of one or more parts at level zero in the hierarchy on the same or on different layers. A multipart path consists of a single *master path* and one or more *subparts*. The master path is an ordinary path; however, it is the defining part of a multipart path; all subparts are based on the master path.

© 2020 All Rights Reserved.

You can change the attributes of the MPPs master path and subparts through the Edit ROD Multipart Path Properties form. In this procedure, you change this data in the sample MPP shown here.

Label	Description	Original Data	Edited Data
A	Offset subpath begin and end offset	2	1
B	Offset subpath width	1	3
C	Enclosure subpath	0.6	-1
D	Subrectangle length and width	1	2

The *Edit ROD Multipart Path Properties* form lets you:

- View and change the attributes of the master path.
- View and change the attributes of the subparts, except for the layer.
- Change the ROD information for the entire multipart path including:
  - System handle
  - User handle
  - Alignment
- View and change the ROD properties.

To view the attributes of an MPP,

1. Select an MPP.
2. Choose *Edit – Basic – Properties* [Q].

© 2020 All Rights Reserved.

The *Edit ROD Multipart Path Properties* form appears.

For complete form field descriptions, see the ROD Subpart Form. For an example of how to edit an MPP, see Editing Multipart Paths.

Limitations to Editing Multipart Paths

There are a few attributes that cannot be edited using the Edit ROD Multipart Path Properties form.

- Adding or deleting subparts of an existing MPP

  To add or delete subparts of an existing MPP, you must use the *Create – Multipart Path* command. For example, if you want to delete all the subrectangles from an existing MPP, you would have to create a new MPP minus the subrectangles, rather than deleting the rectangles in the Edit ROD Multipart Path Properties form.

  However, you can create a new MPP quickly by loading a template containing all the data about the particular MPP. After loading the template, you can delete the subrectangle information.

- Changing subpart layers

  The *Layer* field displays the current layer for the selected subpart, but it is not editable. To change a subpart layer, you must create a new multipart path containing the subpart layer you want.

Edit the subparts of the sample by doing the following:

1. Select the MPP.
2. Choose *Edit – Basic – Properties*.

   The *Edit ROD Multipart Path Properties* form appears.
3. Click *Subpart.*

   The *Edit ROD Subpart* form appears. The three subpart choices appear in the middle of the form: *Offset Subpath*, *Enclosure Subpath*, and *Subrectangle*.

   Alternatively, you can use the context-sensitive menu in the *Property Editor* assistant to add an o*ffset subpath*, *enclosure subpath*, or *subrectangle.*
4. Click *Offset Subpath*.

   The form changes and the offset subpath fields appear.
5. Select the subpart in the scroll window at the top of the form.

© 2020 All Rights Reserved.

6. Change *Begin Offset* and *End Offset* from `2` to `1`.
7. Change *Width* from `1` to `3`.
8. Click *Edit* to apply the change.
9. Click *OK* to view the change.

   The MPP contains the edited offset subpath.

10. Click *Subpart* in the Edit ROD Multipart Path Properties form.

    The *Edit ROD Subpart* form appears.
11. Click *Enclosure Subpath.*

    The form changes and the enclosure subpath fields appear.
12. Select the subpart in the scroll window at the top of the form.
13. Change *Enclosure* from `0.6` to `-1`.
14. Click *Edit* to apply the change.
15. Click *OK* to view the change.

    The MPP contains the edited enclosure subpath.

© 2020 All Rights Reserved.

16. Click *Subpart* in the Edit ROD Multipart Path Properties form.

    The Edit ROD Subpart form appears.

17. Click *Subrectangle*.

    The form changes and the subrectangle fields appear.

18. Select the subpart in the scroll window at the top of the form.

19. Change the *Width* and *Length* from `1` to `2`.

20. Click *Edit* to apply the change.

21. Click *OK* to view the change.

    The MPP contains the edited subrectangles.

22. Click *OK* to close the *Edit ROD Multipart Path Properties* form. The three subparts of the MPP have been edited.

Using Net Expressions and Inherited Connections

Inherited Connections

Inherited connections is an extension to the connectivity model that allows you to create global signals and override their names for selected branches of the design hierarchy. This flexibility allows you to use:

- Multiple power supplies in a design
- Overridable substrate connections
- Parameterized power and ground symbols

© 2020 All Rights Reserved.

To learn about connectivity and naming conventions for inherited connections and how to add and edit net expressions in a schematic or symbol cellview, refer to the *Virtuoso Schematic Editor User Guide*.

Net Expressions in the Layout Editor Environment

A net expression is a special property placed on a net or terminal to define its connectivity. A net expression consists of a property name and a default net name. You use a net expression to override the connection made by a signal or terminal. This allows you to use, for example, multiple power supplies in your design. The signal is redefined due to the value assigned to it. Redefining the signal eliminates the problem of global nets being merged into a single, electrically-equivalent signal, which occurs when the signal traverses the design hierarchy.

You can create, edit, and delete net expressions for pins by changing the net expression information in the *Connectivity* category of the *Edit Properties* form. Once the net expression information is applied to a terminal, the net expression is displayed in the text labels of pins. The following must be true for the net expression to display:

- The pin is created with *Create Label* set on in the *Create Pin* form
- *Net Expressions* is set on in the *Display Options* form

  If *Net Expressions* is set off, the terminal name is displayed.

Note: Net expressions are not interpreted by the display code in the layout editor. The net expressions are displayed as is, but other tools, such as the Layout Versus Schematic (LVS) program, use and interpret net expressions. The net expressions on layout terminals must match those found in schematic views.

Creating a Net Expression

To create a net expression,

1. Select the pin whose terminal you want to assign a net expression to.
2. Open the Edit Properties Form.
3. Choose the *Connectivity* radio button.
4. Type the net expression override property name in the *Net Expression Property* field.
5. Type the default net name in the *Default* field. This is used when no override property is defined in the hierarchy above this point in the schematic view. If you do not enter a default net name, the terminal name is used.

© 2020 All Rights Reserved.

6. Click *Apply.*

   The pin label display changes to show the net expression property preceded by the @ symbol and the default net name preceded by the % symbol.

Note: The *Net Expressions* field in the Display Options Form must be set to on for the net expressions to display in your cellview.

Editing Net Expressions

To edit net expression information for pins,

1. Select the pin whose terminal's net expression you want to change.
2. Open the Edit Properties Form.
3. Choose the *Connectivity* radio button.
4. Change the information in the *Net Expression Property* and *Default* fields.

(Clearing the *Net Expression Property* field in the Edit Properties form deletes the net expression from your design.)

If several pins in your design have the same terminal name, you need update only one pin and the rest will update automatically. You see the edited text labels after you use *View – Redraw*.

Viewing Instances Containing Net Expressions

When you view an instance containing a pin whose terminal has a net expression, the label display is the terminal name, not the net expression. To see the net expression in a hierarchical design, you must descend into the cellview containing the net expression to see the full text display of the net expression.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

11

Using Connectivity

This chapter contains these topics:

- Overview of Pin Connectivity
- Propagating Nets
- Adding and Deleting Shapes on Nets
- Assigning and Unassigning Nets to Shapes
- Selecting Shapes Based on Nets
- Tracing Nets Using the Mark Net Command
- Probing Nets, Instances, Pins, and Shapes

See also *Implementing External Connections (Must Connects) in Parameterized Cells*. This is an application note on Cadence Online Support (COS), containing information and examples about how to create Cadence® SKILL language-based parameterized cells (Pcells) that use external connections.

© 2020 All Rights Reserved.

Overview of Pin Connectivity

Any object with pins is a device, which includes transistors, resistors, or capacitors. By default, pins are connected internally. The *Connectivity – Pins* command lets you tell the router the level at which to connect a pin or group of pins in a net, internally or externally.

- The *Connectivity – Pins – Must Connect* command tells the router to connect the selected pin or group of pins in a net externally (at a higher level). When you connect pins externally, you do not need to route between the pins inside the cell. See Defining Must Connect Pins.
- The *Connectivity – Pins – Strongly Connected* command tells the router to connect the selected pin or group of pins in a net internally (within the device). When you connect pins internally (strong connect), at least one pin in the group must be connected externally. See Defining Strongly Connected Pins.
- The *Connectivity – Pins – Weakly Connected* command tells the router to connect the selected pin or group of pins in a net externally. Similar to strongly connected pins, all weakly connected pins should also be connected together internally. Weakly connected pins have limited external connection to avoid specific internal connections (typically ones with high-resistance paths). See Defining Weakly Connected Pins.
- The *Connectivity – Pins – Pseudo Parallel Connect* command tells the router that the selected instance terminals (within an instance) do not need to be connected, even though they are on the same net. See Defining Pseudo Parallel Connected Net.

The ability to define connection types for pins can be useful if:

- You are a library developer designing a library device for use in a block or circuit design and need to provide layout designers with maximum flexibility by defining some connections internally and leaving other connections to be completed at the next level of design (see For Library Developers)
- You are a layout designer using a device designed by others in a layout and need to specify, later in the layout cycle, which pins to connect externally (see For Layout Designers)

The following examples show how the ability to define pins as must connects, strongly connected, weakly connected, or pseudo parallel connects can make the design process more efficient for both these job functions.

© 2020 All Rights Reserved.

For Library Developers

If you are working as a library developer, a typical device might be a multi-fingered FET like this.

- G(1) and G(2) are weakly connected (inside the device)
- G(3) and G(4) are weakly connected (inside the device)
- The pair G(1)/G(2) must be externally connected to the pair G(3)/G(4)
- S(1) must be externally connected to S(2)

To use this device in a design, if no external (must-connect) connections are defined, you must make internal (strongly connected) connections so the cell looks like this:

© 2020 All Rights Reserved.

Then, when this device is used in a design, the flight lines look like this:

If the layout designer makes connections between G and G(2) or G(4), the layout editor accepts the connection as complete. If the layout designer makes connections between S and S(1) or S(2), the layout editor accepts the connection as complete.

If you, as a library developer, do not want to complete the connections to the source or gate inside the device, but want the connections made at the next level of hierarchy, you do not wire the internal connections. Then, when the cell is used in a design, the flight lines look like this:

If the layout designer makes a connection between S and S(1), the software does not accept the connection as complete and displays a flight line to S(2). If the layout designer makes a connection between G and G(2), the software does not accept the connection as complete and displays a flight line to G(4).

If you make a connection between G and G(2) and G(4), the software accepts the connection as complete and displays no more flight lines on this net, because G(1) and G(3) are weakly connected to G(2) and G(4) respectively.

© 2020 All Rights Reserved.

If you make a connection between G and G(2), and between G(1) and G(3), this is considered an error and is indicated by a marker. The software would not accept the internal weak connection between G(1) and G(2) to complete the net.

For Layout Designers

If you are working as a layout designer, a typical design might be a two-row-high standard cell layout like this.

Floorplanning information regarding external connections is provided on the boundary of the design by an outside source, such as a circuit designer or library. Given that information early, you can identify which pins must be must connects, strong connects, or weak connects, and can use that information for device placement and wiring.

© 2020 All Rights Reserved.

If you define no external connections, the flight lines look like this:

The library developer usually does not connect the top two vdd! pins to the bottom two vdd! pins but does connect the top two vdd! pins together. The same is true for the gnd! pins. So the library developer defines the top two vdd! pins as being externally connected to the

© 2020 All Rights Reserved.

bottom two vdd! pins and the same for the gnd! pins. Then, when you are working inside the cell, the flight lines look like this:

© 2020
All Rights Reserved.

When you use the cell in the next level of hierarchy, the flight lines look like this. You must make the connections as shown by the flight lines. You are provided information regarding pin connections by an outside source, such as a circuit designer or library.

Defining Must Connect Pins

To define a set of pins on a net to be connected externally,

1. From the Layout window, choose *Connectivity – Pins – Must Connect*.

   To open the Define Net/Pin Connections Form, press F3.

   The Layout window prompts you to click a pin figure to select a net.

2. To select the net that includes the pins you want to connect externally, click a shape on that net.

   Note: You can select a net by selecting a shape in the layout before you choose the *Must Connect* command.

© 2020 All Rights Reserved.

Flight lines appear, connecting only the pins of the net you choose. The flight lines are drawn between pins that still need to be internally (strongly) connected. If incomplete nets are being displayed, the other net flight lines disappear.

Selecting a net does not make a connection change.

The Layout window prompts you to select one or more pins to be connected externally.

Note: Flight lines are visible in VLS XL and higher tiers.

3. To select the pins you want to connect externally, click pins, one or more, in the selected net.

   You can click a pin, `shift`-click subsequent pins, or select pins by area (click and drag with the right mouse button).

   Clicking on the pins of the selected net makes the connection change.

   - If you select a single pin, all other pins on the net are deselected and that pin is connected externally to all other pins on the net.
   - If you select multiple pins, those pins are connected strongly (internally) as a set and are connected as must connects (externally) to all other pins on the net.

   When you define the connection of each pin, the flight line display of the net immediately changes to show the connection.

   You can also use the *Define Must Connected Pins* form to select nets.

4. To open the *Define Must Connected Pins* form, press `F3` while the *Define External Pins* command is active.

   The Define Must Connected Pins form appears.

5. In the form, type the name of a net in the *Net* field or click the net names in the list box to select them.

   To select more than one net at a time to work on, you can

   - Choose multiple nets in the option form
   - Use drag by area when the command line prompts you to select a net

   Note: You must define external pin connectivity within a net. You cannot define external pin connectivity between nets.

6. When you are finished, press `Escape` to cancel the command.

   When you define pins as must-connects, the router routes them at the next level of hierarchy.

© 2020 All Rights Reserved.

Note: You can change from the *Must Connect* command to the *Strongly Connected* command to the *Weakly Connected* command to the *Pseudo Parallel Connect* command with the right mouse button. Must Connect pins must be changed to Strongly Connected before being changed to Weakly Connected.

Defining Strongly Connected Pins

To define a set of pins on a net to be connected strongly (internally),

1. From the Layout window choose *Connectivity – Define Pins – Strongly Connected*.

   To open the Define Net/Pin Connections Form, press F3.

   The Layout window prompts you to click a pin figure to select a net.

2. To select the net that includes the pins you want to connect, click a shape in that net.

   Note: You can select a net by selecting a shape in the layout before you choose the *Strongly Connected* command.

   Flight lines appear, connecting only the pins of the net you chose. The flight lines are drawn between pins that still need to be strongly connected. If incomplete nets are being displayed, the other net flight lines disappear.

   Selecting a net does not make a connection change.

   The Layout window prompts you to select two or more pins to be connected strongly.

   Note: Flight lines are visible in VLS XL and higher tiers.

3. To select the pins you want to connect strongly, click pins, one or more, in the selected net.

   You can click a pin, `shift-click` subsequent pins, or select pins by area (click and drag with the right mouse button).

   Clicking on the pins of the selected net makes the connection change.

   - If you select a single pin, the other pins are not deselected, and that pin is connected strongly to the current selection of strongly connected pins.
   - If you select multiple pins, those pins are connected strongly to all other strongly connected pins on the net.

   When you click pins to be strongly connected, the net flight line display immediately changes to show the new information.

   You can also use the Define Strongly Connected Pins form to select a net.

© 2020 All Rights Reserved.

4. To open the Define Strongly Connected Pins form, press `F3` while the *Strongly Connected Pins* command is active.

   The Define Strongly Connected Pins form appears.

5. In the form, type the name of a net in the *Net* field or click net names in the list box to select them.

6. In the layout, click the pins you want to define.

   When you add or remove pins to be strongly connected, the net flight line display immediately changes to show the new information.

   To select more than one net at a time to work on, you can

   - Choose multiple nets in the option form
   - Use drag by area when the command line prompts you to select a net

   Note: You must define strong pin connectivity within a net. You cannot define strong pin connectivity between nets.

7. When you have finished, press `Escape` to cancel the command.

   When you define pins as strongly connected, the router routes them within the design.

Note: You can change from the *Strongly Connected* command to the *Must Connect* command to the *Weakly Connected* command to the *Pseudo Parallel Connect* command with the right mouse button.

Defining Weakly Connected Pins

To define a set of pins on a net to be connected externally because the internal physical connection has a significant resistance,

1. From the Layout window choose *Connectivity – Define Pins – Weakly Connected*.

   To open the Define Net/Pin Connections Form, press F3.

   The Layout window prompts you to click a pin figure to select a net.

2. To select the net that includes the pins you want to connect externally for this reason, click a shape in that net.

   Note: You can select a net by selecting a shape in the layout before you choose the *Connectivity – Pins – Weakly Connected* command.

© 2020 All Rights Reserved.

Flight lines appear, connecting only the pins of the net you chose. Dotted flight lines are used to show the internal high-resistance path between the pins. If incomplete nets are being displayed, the other net flight lines disappear.

Selecting a net does not make a connection change.

The Layout window prompts you to select two or more pins to be connected.

Note: Flight lines are visible in VLS XL and higher tiers.

3. To select the pins you want to connect, click pins, one or more, in the selected net.

You can click a pin, `shift-click` subsequent pins, or select pins by area (click and drag with the right mouse button).

Clicking on the pins of the selected net makes the connection change.

- If you select a single pin, the other pins are not deselected, and that pin is connected externally to the current selection of externally connected pins.
- If you select multiple pins, those pins are connected externally to all other externally connected pins on the net.

When you click pins to be externally connected as weak-connects, the net flight line display immediately changes to show the new information.

You can also use the Define Weakly Connected Pins form to select a net.

4. To open the Define Weakly Connected Pins form, press `F3` while the *Weakly Connected Pins* command is active. The Define Weakly Connected Pins form appears.

5. In the form, type the name of a net in the *Net* field or click net names in the list box to select them.

6. In the layout, click the pins you want to define.

When you add or remove pins to be weakly connected, the net flight line display immediately changes to show the new information.

To select more than one net at a time to work on, you can

- Choose multiple nets in the option form
- Use drag by area when the command line prompts you to select a net

Note: You must define weak pin connectivity within a net. You cannot define strong pin connectivity between nets.

7. When you have finished, press `Escape` to cancel the command.

When you define pins as weakly connected, the router routes them within the design.

© 2020 All Rights Reserved.

Note: You can change from the *Weakly Connected* command to the *Must Connect* command to the *Strongly Connected* command to the *Pseudo Parallel Connect* command with the right mouse button. Must Connect pins must be changed to Strongly Connected before being changed to Weakly Connected.

Defining Pseudo Parallel Connected Net

For detailed information about pseudo-parallel connections, see the Pseudo-Parallel Connections section in the Virtuoso Layout Suite documentation.

Pseudo parallel connections are like must connects within an instance and represent cases where instance terminals do not need to be connected because current does not pass between them.

To define a set of instance terminals on a net for pseudo parallel connection,

1. From the Layout window, choose *Connectivity – Pins – Pseudo Parallel Connect*.

   To open the Define Pseudo Parallel connected net form, press F3.

   The Layout window prompts you to click to select a net.

   Note: You can select a net by selecting a shape in the layout before you choose the *Pseudo Parallel Connect* command.

2. To select the net that includes the instance terminals you want to connect, click a shape in that net.

   Flight lines appear, connecting only the pins of the net you choose. The flight lines are drawn between pins that can be pseudo parallel connected. If incomplete nets are being displayed, the other net flight lines disappear.

   Selecting a net does not make a connection change.

   If you did not select a net or instance terminal before you chose the command, the Layout window prompts you to select an instance terminal (instTerm).

   Note: Flight lines are visible in VLS XL and higher tiers.

3. To select the instance terminals you want to connect in pseudo parallel connection, click pins, one or more, in the selected net.

   You can click an instance terminal, `shift-click` subsequent pins, or select terminals by area (click and drag with the right mouse button).

   Clicking on the instance terminals of the selected net makes the connection change.

© 2020 All Rights Reserved.

When you define the connection of each terminal, the flight line display of the net immediately changes to show the connection.

You can also use the Define Pseudo Parallel Connected Pins options form to select nets.

4. To open the Define Pseudo Parallel Connected Pins form, press `F3` while the *Pseudo Parallel Connect* command is active.

   The Define Pseudo Parallel Connected Net form appears. The form lists all nets without I/O pins (except le_ex_# nets).

5. In the form, type the name of a net in the *Net* field or click the net names in the list box to select them.

   To select more than one net at a time to work on, you can

   - Choose multiple nets in the option form
   - Use drag-by-area when the command line prompts you to select a net

   Note: You must define pseudo parallel connectivity within a net. You cannot define pseudo parallel connectivity between nets.

6. When you are finished, press the `Escape` key to cancel the command.

   When you define pins as pseudo parallel connects, the router does not route them.

Note: You can change from the *Pseudo Parallel Connect* command to the *Must Connect* command to the *Weakly Connected* command to the *Strongly Connected* command with the right mouse button.

Checking Connectivity of Pins

To see whether pins on a net are defined for external connection,

1. From the Layout window *Connectivity* menu, choose *Pins – Must Connect*.
2. Press `F3` to see the *Connectivity – Must Connected Pins* form.

   The form appears, showing all nets that include terminals (possible connections to devices external to the device).

3. Click any of the nets shown in the form.

   Flight lines for the net appear on the layout (VLS XL and higher tiers), showing whether the terminals are defined as must-connect, strong-connect, or weak-connect.

© 2020 All Rights Reserved.

Propagating Nets

The *Propagate Nets* command maps terminals on an instance or a homogenous mosaic that you select to the nets in the cellview; that is, it defines how the instance or the instances in the homogenous mosaic are logically connected to the cellview. Using this command, you can assign or reassign a top-level net name for each terminal in the selected instances or the instances in the homogenous mosaic. For each such terminal, the command creates a top-level net, if necessary, and creates an instTerm on that net for the instance terminal. You use this command when you do not have a schematic for your layout cellview.

Routers use this logical connectivity information to physically connect the pins of the instance to the rest of the layout.

Promoting Net Information

Note: In the following steps, “top-level” refers to the cellview in which the instance is placed.

To promote net information from an instance in your layout,

1. Select one or more instances in the layout.
2. From the Layout window, choose *Connectivity – Nets – Propagate*.

   The Propagate Nets Form appears. If you select more than one instance, the *Previous* and *Next* buttons are displayed at the top of the form.

   The instance displayed in the form is highlighted in the layout. The form title shows the name and master cell of the instance currently highlighted.The form shows the terminals of the instance and the top-level nets they are mapped to, if any.

   Alternatively, with the instance selected, open the Property Editor assistant and select *Connectivity* from the *Property Category Selector* list. If you select multiple instances on the canvas, then you can set the net on a particular terminal for all the selected instances.
3. For each terminal, type in the top-level net to which you want to map the terminal.

   If a *Net Name* field is left blank, the corresponding terminal is not mapped to a top-level net. *Net Name* fields can be left blank, but the pins on the corresponding terminals will not be routed.

   If you want the net to have the same name as the terminal name, type in the terminal name or click *Set to Terminal Name*.

   If you want all the top-level nets to have the same name as the terminal name, click *Set All Net Names to Terminal Name*.

© 2020 All Rights Reserved.

If you want to undo all the changes to that instance, click *Defaults*. The net names revert to their current value.

If you want to clear all the net names, click the *Clear All Net Names* button.

4. Click *Previous* and *Next* to edit other instances.

   The form keeps track of all the changes you have made. You do not need to click *Apply* after each change before editing other instances.

5. Click *OK* to apply all the changes.

   All the changes you have made are saved to the database.

6. Choose *Options – Display*.
7. In the *Display Controls* area, turn on *Nets*.
8. Click *OK*.
9. Choose *View – Redraw*.

   The new net assignments are displayed in the Layout window.

Before assigning new nets with the *Propagate Nets* command.

After assigning new nets with the *Propagate Nets* command.

Adding and Deleting Shapes on Nets

The *Add Shape To Net* command lets you add selected shapes to nets that are attached to pins. An instance or mosaic can be added to a net as long as there is at most one terminal on the net. An instance cannot be added to a net if it contains instTerms. A shape cannot be

© 2020 All Rights Reserved.

added to more than one net. Terminals and nets are not deleted when the physical pin shape associated with them is deleted.

The *Delete Shape From Net* command lets you delete selected shapes from a net. Terminals and nets are not deleted when the physical pin shape associated with them is deleted.

Adding Shapes to a Net

To add a shape to a net,

1. Select one or more shapes.
2. Choose *Connectivity – Nets – Add Shape*.

   The Add Shape to Net Form appears.
3. Do one of the following:
   - To specify the net based on the selection, turn *Auto* on. For each shape, the software looks for a pin that overlaps the shape and adds the shapes to the net.
   - To specify the net manually, turn *Auto* off and type the net name.
4. Click *OK* to close the form and add the shapes to the nets.

Important

When there are zero-area dot pins on the edge of the shape to be added, the pin is used only if one of its access directions points toward the center of the shape.

Removing Shapes from a Net

To remove a shape from a net,

1. Select one or more shapes to be removed from a net.
2. Choose *Connectivity – Nets – Remove Shape*.

   The Remove Shape From Net Form appears.
3. If the net name shown is not the net you want the shapes to be deleted from, type the correct net name.
4. Click *OK* to close the form and delete the shapes from the specified net.

© 2020 All Rights Reserved.

Assigning and Unassigning Nets to Shapes

You can assign and unassign nets to shapes, instance pins, and routed objects like vias and pathsegs by using the `leHiAssignNet` and `leHiUnassignNet` functions, respectively. You can assign nets to a shape by tapping an already assigned shape, or by selecting a net from the drop-down list or by typing an existing net name in the Assign Net form. This command works in both the pre-selection and post-selection modes.

Note: This functionality works only in Layout XL for instance pins.

Assigning Nets to Shapes in Pre-Selection Mode

To assign a net to a shape in pre-selection mode,

1. Select the shapes to which you want to assign a net.
2. Run the `leHiAssignNet` command.
3. Do one of the following:
   - Tap an already assigned shape to pick the net.

     The net of the assigned shape is assigned to all the selected shapes.
   - Open the Assign Net form by pressing `F3`. From the *Net Name* drop-down list, select a net or type an existing net name, and click *Hide*.

     The net is assigned to all the selected shapes.
4. Press `Esc` to exit the command.

Assigning Nets to Shapes in Post-Selection Mode

To assign a net to a shape in post-selection mode,

1. Run the `leHiAssignNet` command.
2. Do one of the following:
   - Tap an already assigned shape to pick the net.
   - Open the Assign Net form by pressing `F3`. From the *Net Name* drop-down list, select a net or type an existing net name, and click *Hide*.
3. Select the shapes to which you want to assign a net.

   The net is assigned to all the selected shapes.

© 2020 All Rights Reserved.

4. Press `Esc` to exit the command.

You can use the `t` bindkey to assign another net to the shapes.

Unassigning Nets from Shapes in Pre-Selection Mode

To unassign a net from a shape in pre-selection mode,

1. Select the shapes.
2. Run the `leHiUnassignNet` command.

   The net is unassigned from all the selected shapes.

Unassigning Nets from Shapes in Post-Selection Mode

To unassign a net from a shape in post-selection mode,

1. Run the `leHiUnassignNet` command.
2. Select the shapes.

   The net is unassigned from all the selected shapes.

Selecting Shapes Based on Nets

You can use the net-based selection feature to select shapes based on their nets. The two ways to enable this feature are shown below:

- Using the *Net Based Selection* toolbar
  - **a.** Select the *Net Based Selection* toolbar from the *Window – Toolbars* menu.
  - **b.** Click the Net Based Selection icon on the toolbar.

- Using the *Net Based Selection* option on the Selection Options form

© 2020 All Rights Reserved.

Select the *Net Based Selection* option on the Selection Options form

You can also use the environment variable `netBasedSelectionOn` to enable the net-based selection feature.

When you enable the net-based selection feature, there are three ways in which you can specify the nets to select figures based on that net:

- Specify the nets in the text field on the *Net Based Selection* toolbar.

© 2020 All Rights Reserved.

- Select the *Tap Net* icon on the *Net Based Selection* toolbar to populate nets by tapping on an assigned shape in the canvas.

Press `Ctrl` and click to remove the net of the tapped shape from the list of nets.
Press `Shift` and click to add the net of the tapped shape to the list of nets.

- Select nets using the *Navigator* assistant.

When you select the *Reverse* option on the *Net Based Selection* toolbar, shapes, vias and instances with a net other than the net specified in the text field can be selected in the canvas. You can also use the environment variable `netBasedSelectionReverse` to specify the *Reverse* option.

Tracing Nets Using the Mark Net Command

The *Mark Net* command lets you visually trace the physical connectivity of a net(s) in a layout design without having to use a schematic. The command extracts the metal, via, poly and diffusion layer information from the technology file and highlights these in the layout as the

© 2020 All Rights Reserved.

net passes from one layer to the next through the hierarchy. The trace is not a selectable object: it is the highlighted net. The *Mark Net* command is available both in read-only and edit modes.

You can choose which and how many levels of hierarchy to traverse for highlight generation. After the hierarchy selection range is set, you cannot select a shape that is outside the start-stop hierarchical range. You can start the *Mark Net* command from any level of the hierarchy (from *Edit in Place*, for example), and the command will correctly trace the net throughout the designated hierarchical range. *Mark Net Stop Level*, specified in the MarkNet Options Form, can be lower than the *Display Level*, specified in the Display Options Form. However, if you set the *Mark Net Start Level* to greater than zero, it is possible to get incomplete net tracing. When the interconnect is interrupted by a shape on a hierarchical depth outside the specified range, the rest of the net is not highlighted even if the subsequent sections of the net are within the specified range.

A traced net remains highlighted as long as the cellview window remains open, unless you use the *Unmark Net* command to remove the highlighting. If you close the cellview window, the highlighting is deleted. However, as long as the cellview window is open, the nets remain highlighted even after the cellview window is redrawn, after any changes are made to the edit context, or after you exit the command. When the *Mark Net* command is re-invoked, it does not remove the existing highlighting. This allows you to incrementally mark nets, mark nets from any level of the hierarchy, and retain nets marked as you move throughout the hierarchy. You can mark nets in multiple cellview windows. You can also view the highlighted nets in the plot of a cellview. You can unmark a net later if required.

The highlighting is created relative to the current top-level cellview. If you highlight nets in separate cellview windows, the *Unmark Net* command removes all highlighting associated with the current top-level cellview. For example, `cellview Top` contains an instance of `cellview Bottom` and both the cellviews are open at the same time and you have used *Edit In Place* into the instance of `cellview Bottom` in `cellview Top`. The order in which you mark nets in the two cellview windows determines if *Unmark Net* removes the highlighting from one or both the cellview windows. If you mark the nets first in `cellview Bottom`, then this is the current top-level cellview. The marked nets will be highlighted in `cellview Top` as well. However, if you further mark nets in `cellview Top` and then *Unmark All Nets* in `cellview Bottom`, then it will unmark only those nets that were

© 2020 All Rights Reserved.

marked in `cellview Bottom`. You need to return to `cellview Top` and use *Unmark All Nets* to delete the additional highlighting in that window.

The final highlighted net is created by generating one large polygon representing the shapes on the net, so the highlight shows the outline of all the marked shapes; individual shapes on the net are not highlighted.

If the number of shapes on the net is 10,000 or less, a one-line message appears in the CIW, as in the following example:

```
Number of shapes processed by Mark Net: 5
```

If the number of shapes on the net is greater than 10,000, the output message in CIW displays each time 10,000 shapes are processed. The traced nets are also displayed incrementally, in sets of 10,000 shapes. When the last shape is processed, the final number of shapes processed is displayed, as in the following example:

```
Number of shapes processed by Mark Net: 10000...
Number of shapes processed by Mark Net: 20000...
Number of shapes processed by Mark Net: 30000...
Number of shapes processed by Mark Net: 30030
```

The *Mark Net* command does not highlight elliptical nets or circular vias.

For the *Mark Net* command to work, your technology file must list the interconnect layers in the `viaDefs` section of the technology file. Via definitions are of two types, `customViaDefs` and `standardViaDefs`. For more information about via definitions, see the *Virtuoso Technology Data ASCII Files Reference* for Via Definitions and Via Specifications. The *Virtuoso Technology Data User Guide* describes how to create and edit a technology file.

You can control the thread mode, single, auto or custom, of the Mark Net command using the environment variable, markNetNumThreads.

© 2020 All Rights Reserved.

Specifying Via, Connected, and Stop Layers for Tracing a Net

You can specify the via layers and the connected layers through which you want to trace a net. You can either use the via layers defined in the technology file or add your own via layers. To add a via layer, specify it as a (*layer1 viaLayer layer2*) combination on the Via Layers Tab of the MarkNet Options Form. Both the layers in the via layer combination can be specified as layer-purpose pairs (LPPs). To define connected layers, specify them in the *(layer1 purpose) (layer2 purpose)* format on the Connected Layers Tab of the MarkNet Options Form.

You can also specify the layers through which the *Mark Net* command must not trace a shape. These resistive layers, referred to as stop layers, help to block tracing through devices such as poly resistor and MIM capacitor. Specify the stop criterion in the *layer stopLayer* format on the Stop Layers Tab of the MarkNet Options Form. You can specify both *layer* and *stopLayer* as LPPs.

Note: To stop tracing a net at a particular via layer, Cadence recommends that you remove or disable that via layer entry on the Via Layers Tab, instead of defining it as a stop layer on the Stop Layers Tab. Having the entry only on one of the tabs prevents ambiguity and undesirable results.

In a new Virtuoso session, if the *Customized Via Layers*, *Customized Connected Layers*, and *Enable Stop Layers* check boxes are selected, the settings for the via, connected, and stop layers are read from the `markNetOptions` file stored at the following location in the technology library directory:

```
<technologyLibraryName>/.cadence/dfII/markNet/markNetOptions
```

After the *Mark Net* command completes its run or when you load new settings from a previously saved file, the via, connected, and stop layer settings are automatically stored in the `markNetOptions` file stored at the following location in your current working directory:

```
<currentWorkingDirectory>/.cadence/dfII/markNet/<technologyLibraryName>/
markNetOptions
```

Subsequently, layer settings read from the `markNetOptions` file in your current working directory override the settings read from the file stored in the technology library directory.

If you want a file containing your customized layer settings to be read each time you start a new Virtuoso session, use the markNetOptionsFile environment variable to specify the path or name of this file. The layer settings stored in the file specified using the `markNetOptionsFile` environment variable override the settings stored in the other two `markNetOptions` files.

The format of the `markNetOptions` file is as follows:

```
connectedLayers(
```


© 2020 All Rights Reserved.

```
; (layer1 purpose) (layer2 purpose) ... enabled
)
stopLayers(
; (layer purpose) (stopLayer purpose) enabled
)
```

The figure below shows a sample `markNetOptions` file. Notice that there are separate sections in the file for via, connected, and stop layers. These sections can be present in any order. You can edit the `markNetOptions` file manually.

```
;Do not remove or change these header lines.
;markNetOptions file version 1.0
viaLayers(
Metal1 Via1 Metal2 t
Metal2 Via2 Metal3 t
Metal3 Via3 Metal4 t
Metal4 Via4 Metal5 t
Metal5 Via5 Metal6 t
)
connectedLayers(
("Metal1" "net") ("Metal2" "net") nil
("Metal1" "net") ("Metal2" "drawing") nil
)
stopLayers(
("Metal1" "net") ("Metal2" "net") nil
("Metal1" "net") ("Metal2" "drawing") nil
("Metal1" "drawing") ("Metal2" "net") nil
Metal1 Metal2 nil
("Metal2" "net") ("Metal1" "net") nil
("Metal1" "drawing") ("Metal1" "pin") nil
("Metal1" "drawing") ("Metal2" "pin") nil
)
```

You can also manually save the settings for any or all via, connected, and stop layers to a file and location of your choice by using the *Save* feature in the MarkNet Options Form. By using the *Load* feature, you can later load the settings from the saved file.

In the figure below, two sets of shapes are shown, labeled (a) and (b). In (a), `M1 dg` overlaps `M2 net`, and in (b), and `M1 dg` overlaps `M2 pin`. `M1 dg` and `M2 net` are defined as

© 2020 All Rights Reserved.

connected layers, and `M2 pin` is defined as the stop layer for `M1 dg`. The *Mark Net* command will stop tracing the shape `M1 dg` when there is an overlapping `M2 pin` shape.

When you click `M1 dg` in (a), both `M1 dg` and `M2 net` are traced because they are defined as connected layers. Only `M2` purpose `pin` is defined as the stop layer for `M1 dg`, so tracing continues when the overlapping shape is `M2` purpose `net`. When you click `M1 dg` in (b), the tracing stops at the point the two shapes overlap because `M2 pin` is defined as the stop layer for `M1 dg`. The resultant tracing is shown in the figure below.

Tracing Nets

To trace a net,

1. Choose *Connectivity – Nets – Mark*.

© 2020 All Rights Reserved.

The following message appears in the status line in the design window:

```
Point at a net to mark.
```

2. Press F3 for the MarkNet Options Form to appear.
3. Specify the settings on the *Options* tab.

   a. Set *Hierarchy Range*.

      If you choose *User defined* from the drop-down list, the *Start* and *Stop* fields become editable. Specify the start and stop levels for starting and stopping net tracing.

   b. Specify the area to trace.

© 2020 All Rights Reserved.

By default, the entire cellview is traced. You can also specify a rectangular or polygonal area.

c. Ensure that the *Auto Cycle* check box is selected if you want the color of highlighted nets to automatically cycle through Cadence-defined LPPs.

Deselect the *Auto Cycle* check box if you want to manually specify the color for highlighted nets. This enables the *Highlight Color* cyclic field. Select the required LPP from the cyclic field.

d. Select the *Highlight Incrementally* check box if you want to view a progressive highlighting of the connectivity.

Tip

Instead of displaying the connectivity incrementally, you can use `Ctrl + c` to stop the *Mark Net* command from further processing shapes and to display the marked shapes processed until then.

e. Select the *Highlight Individual Shapes* check box to display a separate highlight for each shape.

f. Select the *Select From Overlaps* check box.

If this check box is selected, the *Choose LPP to Mark* form appears when you click at a point where shapes overlap. Select an LPP in this form to specify the shape from which to begin tracing.

g. Ensure that the *Customized Via Layers*, *Customized Connected Layers*, and *Enable Stop Layers* check boxes are selected to enable the settings specified on the *Via Layers*, *Connected Layers*, and *Stop Layers* tabs to be used for tracing nets. If any of these check boxes are not selected, the corresponding tab also gets disabled.

h. Select the *Retain Via Info for Save MarkNets* check box before running the *Mark Net* command if you later want to save via shapes in a cellview by using the *Connectivity – Nets – Save All MarkNets* command. By default, no via shapes are saved in the cellview used for storing the traced shapes.

© 2020 All Rights Reserved.

i. In the *Save/Load custom settings for Via*, *Connected and Stop Layers* section, specify the file name to which you want to save settings or from which you want to load the settings.

To save the settings to a file:

i) Click *Browse* and specify the location and name of the file.

By default, all the settings specified on the *Via Layers*, *Connected Layers*, and *Stop Layers* tabs are saved. If you do not want to save the settings for a layer type, deselect the corresponding check box.

ii) Click *Save To*.

To load the settings from a file:

i) Click *Browse* and specify the location and name of the file.

By default, the *Via Layers*, *Connected Layers*, and *Stop Layers* tabs are all populated with the settings stored in the file. If you do not want to load the settings for a layer type, deselect the corresponding check box.

ii) Click *Load From*.

© 2020 All Rights Reserved.

4. Click the *Via Layers* tab.

a. In the *layer1*, *viaLayer*, and *layer2* fields, specify the LPPs — the layer name in the *layer* column and the purpose name in the *purpose* column.

If you leave the *purpose* field blank, the *Mark Net* command interprets it as all purposes of the specified layer.

b. Click *Add*.

The new via layer in the `(layer1) (viaLayer) (layer2)` format is displayed in the *Via Layers* list box.

Note: If any of the layers is invalid or the specified via layer combination already exists, clicking *Add* does not add the via layer to the list box above. A warning

© 2020 All Rights Reserved.

message informing you about the same is displayed in the CIW.

By default, the *Enable* check box against the newly added via layer is selected and the via layer is used for tracing. You can deselect the *Enable* check box for any via layer that you do not want to be used for tracing. To select or deselect all the via layers in one go, select or deselect the *Enable* check box at the top of the *Enable* column.

c. Select the check box against a via layer in the *Delete* column and click *Delete* if you want to delete the via layer. Select the *Delete* check box at the top of the *Delete* column and click *Delete* if you want to delete all the via layers.

d. Select the *Reset to TechDB Defaults* check box if you want to use the default via layers defined in the technology file. All via layers defined in the technology file are displayed in the list box, and the *Enable* check box for each is selected by default.

Any via layers added by you are deleted from the list box. If you want, you can add your own via layers after populating the ones from the technology file.

© 2020 All Rights Reserved.

5. Click the *Connected Layers* tab.

   a. Click *Add More Entries*.

      A text field appears in the *Connected Layers* list box.

   b. Specify the connected layer combination in the text field.

© 2020 All Rights Reserved.

In the example shown in the figure below, the connected layers added imply that `metal2-drawing` is connected to `metal3-drawing` and `metal1-drawing4` is connected to `metal2-drawing4`.

You can specify more than two LPPs as connected layers. If you do not specify the purpose, all purposes of that metal layer are connected to the other specified LPP.

c. Click *Add More Entries* to add more entries for connected layers.

By default, the *Enable* check box against the newly added connected layer is selected and the connected layer is used for tracing. You can deselect the *Enable* check box for any connected layer that you do not want to be used for tracing. To select or deselect all the connected layers in one go, select or deselect the *Enable* check box at the top of the *Enable* column.

d. Select the check box against a connected layer in the *Delete* column and click *Delete* if you want to delete the connected layer. Select the *Delete* check box at the top of the *Delete* column and click *Delete* if you want to delete all the connected layers.

e. Select the *Reset to TechDB Defaults* check box if you want to use the default connected layers defined in the technology file. All connected layers defined in the technology file are displayed in the list box, and the *Enable* check box for each is selected by default.

Any connected layers added by you are deleted from the list box. If you want, you can add your own connected layers after populating the ones from the technology file.

© 2020 All Rights Reserved.

6. Click the *Stop Layers* tab.

a. In the *layer* and *stopLayer* fields, specify the LPPs — the layer name in the *layer* column and the purpose name in the *purpose* column.

If you leave the *purpose* field blank, the *Mark Net* command interprets it as all purposes of the specified layer.

b. Click *Add*.

The new stop layer in the `(layer) (stopLayer)` format is displayed in the list box above.

In the example shown in the figure below, the stop layers added imply that `pimplant-drawing` is a stop layer for all purposes on `y9`, `cont-drawing` is a

© 2020 All Rights Reserved.

stop layer for `metal1-drawing`, and all purposes on `y9` are stop layers for all purposes on `metal3`.

By default, the *Enable* check box against the newly added stop layer is selected and the stop layer is used while tracing. You can deselect the *Enable* check box for any stop layer that you do not want to be used while tracing. To select or deselect all the stop layers in one go, select or deselect the *Enable* check box at the top of the *Enable* column.

c. Select the check box against a stop layer in the *Delete* column and click *Delete* if you want to delete the stop layer. Select the *Delete* check box at the top of the *Delete* column and click *Delete* if you want to delete all the stop layers.

7. In the design window, click the shape to highlight the required net.

If you click to select an already highlighted net, the following warning message appears in the CIW:

```
*WARNING* Selected object is on an existing marked net.
```

The net is highlighted on the specified levels of the hierarchy.

8. Click a point on the second net that you want to highlight. The second net is highlighted on the specified levels of the hierarchy.

Saving Marked Nets

To save a traced net:

1. Trace a net using the steps described in Tracing Nets.

2. Select *Connectivity – Nets – Save All MarkNets*.

This opens the Save All Mark Nets Form.

© 2020 All Rights Reserved.

3. Specify the library, cell, and view names where you want to store the highlighted nets. Alternatively, click *Browse* to select the library, cell, and view names.

   If the specified cellview already exists, a warning message will be issued to confirm whether or not you want to overwrite the specified cellview.

Note: If you delete some shapes or objects from a highlighted net before saving it, the deleted shapes or objects will not be saved in the specified cellview. A warning message to this effect will also be issued.

Unmarking Nets

To unmark a traced net,

1. Choose *Connectivity – Nets – Unmark* to delete the highlighting of one or all of the marked nets.

   The Unmark Net Form appears.

2. To unmark one of the nets, click that net.
3. To unmark all the nets, click *Unmark All*.

To unmark all of the marked nets,

1. Choose *Connectivity – Nets – Unmark All* to delete the highlighting of all the marked nets.

Probing Nets, Instances, Pins, and Shapes

Probing allows you to find and highlight certain types of objects from among all the objects present in a design. It lets you highlight the logical connectivity between objects. These objects include nets, terminals, paths, and instances. Probing also lets you select objects to highlight their corresponding elements in a different representation of the same design (schematic and layout windows).

For information regarding the probing SKILL functions, see the *Virtuoso Design Environment SKILL Reference*.

To select the objects to be highlighted when you probe/select a net, instance, pin, or shape,

1. Choose *Options – Highlight*.

   This displays the Highlight Options Form.

© 2020 All Rights Reserved.

2. Select *Flightlines*. All shapes and opens on the probed net will be highlighted.
3. To highlight pins and instances on the probed/selected nets, select the *Pins* and *Instances* check boxes.
4. To highlight only the shapes and pins on the visible layers, select the *Visible Shapes Only* option.
5. Set the *Hierarchy Depth* to specify the depth up to which you want to display net probing.
6. Select *Dynamic Highlight* to highlight any object over which you hover pointer.
7. Select the drawing layer to be used for highlighting the nets.
8. Select the *Cycle* option to use the next color for highlighting subsequent objects.
9. Click *Apply*.

After configuring the options in the *Highlight Options* form, you can probe nets, shapes, pins, and instances using either pre- or post-selection mode:

- Pre-Selection Probing
- Post-Selection Probing

Pre-Selection Probing

1. In the Highlight Options Form, select *Flightlines.*
2. To highlight pins on the probed/selected nets, select *Pins.*
3. In the cellview, select one, a few, or all objects irrespective of their connectivity.
4. Enter the following SKILL function in the CIW:

   ```
   geEnterAddNetProbe()
   ```

   Objects with the same connectivity are probed with the same color.
5. Repeat steps 3 and 4 to probe more objects, if required.

Tip

Use the command `geDeleteAllProbe` in the CIW to remove the probes.

© 2020 All Rights Reserved.

In the design below, different objects have different connectivities, as indicated:

The following example illustrates the scenarios when the probe command is executed after either one, two, or all of the objects are edited.

© 2020
All Rights Reserved.

- When a single object is edited.

© 2020
All Rights Reserved.

- When multiple objects are edited.

© 2020 All Rights Reserved.

- When all the objects are edited.

All objects edited

Objects on the same net are probed using the same color.

Post-Selection Probing

1. In the Highlight Options Form, select *Flightlines.*
2. To highlight pins on the probed/selected nets, select *Pins.*
3. Enter the following SKILL function in the CIW:

   ```
   geEnterAddNetProbe()
   ```

4. In the cellview, click a single object.

   Other objects with the same connectivity as the selected object are probed.

© 2020 All Rights Reserved.

Note: In the post-selection mode, objects are probed only when you single-click an object. Objects are not probed if you select multiple objects or use the area selection method.

5. You can continue to select other objects, one at a time, with a single click. Objects with the same connectivity are probed with the same color. Earlier probed objects are retained.

6. Press Esc to cancel the probe command.

© 2020
All Rights Reserved.

Specifying True Color Highlights

You can use the environment variables, `trueColorProbe` and `probeHighlight` to specify the true color highlight. These variables should be set before opening the Virtuoso session.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

12

Microwave Commands

This chapter contains these topics:

- About Microwave Layouts
- Creating Transmission Lines
- Creating Bends
- Creating Tapers
- Transmission Line Bend Examples

© 2020 All Rights Reserved.

About Microwave Layouts

Microwave layouts consist of placed component instances such as resistors, capacitors, inductors, and transistors, and the interconnects drawn to connect the components.

You can connect components in a microwave layout with regular layout geometries such as polygons, rectangles, and paths, as well as microwave specific geometries: transmission lines, bends, and tapers.

You use the microwave commands to create transmission lines, bends, and tapers to connect your components. The microwave commands appear in the layout editor *Create* menu.

Editing Transmission Lines

You can edit microwave objects with the layout editor commands similar to the way you edit paths. You use these commands to

- Change the angle of paths
- Stretch sections of paths
- Reshape a path

Creating Transmission Lines

The *Trl* command lets you create a transmission line.

To display the Transmission Line form,

➤ Choose *Create – Microwave – Trl.*

Transmission lines (trl's) show connections in microwave designs. You create trl's the same way you create paths.

To create a transmission line:

1. Choose *Create – Microwave – Trl*.

   The Transmission Line Form appears.

2. Set *Bend Style* to *bend*.
3. Set *Bend Factor* to `1`.
4. Set the width to `2`.

© 2020 All Rights Reserved.

5. Click to create each point of the trl.

6. To complete the trl, double-click the last point.

Creating Bends

The Bend command lets you create a two-segment transmission line with two different width settings. Bends connect objects of different sizes, using a right-angle segment instead of a straight line. You create bends similar to the way you create paths.

To create a bend:

1. Choose *Create – Microwave – Bend*.

   The Transmission Line Bend Form appears.

2. Type the beginning and ending widths of the bend.

© 2020 All Rights Reserved.

3. Click to enter each point of the bend.

The length of the bend appears as you enter points.

Click on the first, second, and third points of the bend.

This is a completed bend.

Creating Advanced Transmission Lines

The *Bend Style* option lets you customize the corners of your transmission lines into standard bends, chamfers, and radial corners.

© 2020 All Rights Reserved.

1. In the Transmission Line Bend Form or in Transmission Line Form, choose *Bend Style* and type the corresponding factor to use.

Bend Style: **Bend**
Bend Factor: **1**

Bend Style: **Chamfer**
Chamfer Factor: **.2**

Bend Style: **Radial**
Radius Factor: **1**

Bend Style: **Bend**
Bend Factor: **3**

Bend Style: **Chamfer**
Chamfer Factor: **.5**

Bend Style: **Radial**
Radius Factor: **.5**

2. Click each of the three points of the bend.

Creating Tapers

The Taper command lets you create a one-segment transmission line that connects two transmission lines of different width settings.

You create tapers similarly to the way you create paths.

To create a taper:

1. Choose *Create – Microwave – Taper.*

   The Transmission Line Taper Form appears.

© 2020 All Rights Reserved.

2. Type in the beginning and ending widths of the taper.

3. Click to create each point of the taper.

This is a completed taper.

Transmission Line Bend Examples

Standard Bend Examples

This illustration shows a standard bend. All angles in a standard bend are 90 degrees.

Bend Factor is the allowed ratio of the width (W_{MAX}) of the bend for a given angle to the width (W_0) of a 90-degree bend. If the width (W) of a bend is greater than W_{MAX}, the bend is

© 2020 All Rights Reserved.

chamfered so that the width is decreased to W_{MAX}. The effect of *Bend Factor* is shown in the following illustrations.

90-Degree Bend with Width W_0

Specified Value of Bend Factor Defines W_{MAX}

$$\text{Bend Factor} = \frac{W_{MAX}}{W_0}$$

For $W > W_{MAX}$, bend chamfered to decrease W to W_{MAX}

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

Chamfer Examples

The types of chamfer bends and the effect of *Chamfer Factor* are shown in the following illustrations. The appearance of the corner is determined by the value of the chamfer factor, d_1/d_2.

Example 1

chamfer factor = d1 / d2 = .7

Example 2

chamfer factor = d1 / d2 = .5

Example 3

chamfer factor = d1 / d2 = .3

© 2020 All Rights Reserved.

Example 4

chamfer factor = 0.0

© 2020 All Rights Reserved.

Radial Examples

A radial bend has a rounded corner determined by the *Radius Factor.* The *Resolution* is the number of segments in a 180-degree bend.

r = (Radius Factor) * (width)

Bends must be properly formed. For example, segments adjacent to a bend must not be too short with respect to the bend. To prevent this, the layout editor does not allow a bend on adjacent segments to extend past the midpoint of the shortest adjacent segment.

In other words, the bend and the segment must coincide before or at the midpoint of the segment so that segments approaching the midpoint from opposite directions match at the

© 2020 All Rights Reserved.

midpoint. These segments must intersect the midpoint at 90 degrees. If necessary, the layout editor alters the bend to satisfy these conditions.

© 2020 All Rights Reserved.

13

Handling Unexpected Results

This chapter contains these topics:

- Design Access Problems
- Problems with Editing and Entering Points
- Mouse and Cursor Behavior
- Selection Tips
- Window Display Tips
- Problems with Stretching Parameterized Cells

© 2020 All Rights Reserved.

Design Access Problems

You might encounter the following situations while working in the Virtuoso® Layout Suite L layout editor.

Cannot Find a Library

If you do not see the name of the library you want in the *Library Name* field of the *Open File* form, one of the following might be true:

- The library path in the `cds.lib` file is incorrect.
- The library is not in the `cds.lib` file.

Cannot Open a Cellview

If you cannot open a cellview in a library, you might not have read access to the cellview files or the library might be empty. In these situations, the Open File form displays `<None>` in the *Cell Name* and *View Name* fields.

To gain read access to the cellviews in a library,

➤ Do one of the following:

- ❑ Change the access permissions using the *Tools – Library Manager – Edit – Access Permissions* form.
- ❑ Use the UNIX command `chmod` to change the access permissions in the UNIX directory containing the library.

Cannot Write to a Cellview

If you have write access to a library but cannot open a cellview to edit, one of the following might be true:

- You do not have write access for the cellview file
- Another user is editing the cellview and has locked it

If the owner of the cellview set privileges to read only, you see a message like this:

```
The cellView could not be opened for edit. Do you want to open it for read?
```

To gain write access to the cellviews in a library,

© 2020 All Rights Reserved.

➤ Do one of the following:

- ❑ Change the access permissions using the *Tools – Library Manager – Edit – Access Permissions* form.
- ❑ Use the UNIX command `chmod` to change the access permissions in the UNIX directory containing the library.

Cell Instances Are Missing

A cellview often contains instances of cells from other design libraries. If you open a cellview that contains instances of cells from a library that the layout editor cannot find, the following happens:

- When you try to open the cellview, you see a warning dialog box listing cells that the layout editor cannot find

© 2020 All Rights Reserved.

- When you close the dialog box, the cellview opens, but each area containing a missing cell displays a flashing box with an X

To include the missing cells,

➤ Add the path to the library containing the cell masters to the `cds.lib` file.

Loading an ASCII Template File Returns a CIW Warning Message

```
Warning in CIW: MPP template name is replacing an existing MPP template
```

When you load an ASCII MPP template file, the system checks the names of the MPP templates in the ASCII file against the names of MPP templates that already exist in the technology file in virtual memory. If there is a name conflict, the system displays a warning in the CIW saying that an MPP template in the ASCII file is replacing an existing MPP template.

The existing template is overwritten in virtual memory. If you do not want to overwrite the original MPP template in your binary technology library on disk, then do not save changes to

© 2020 All Rights Reserved.

the technology file. You can also avoid overwriting existing templates by changing the duplicate names in your ASCII file.

If you want to change the names of the MPP templates in your ASCII file right away, or you want to be able to save other changes to your technology library on disk without overwriting the original template, you need to:

1. Exit the layout software without saving technology file changes.
2. Edit your ASCII file to change template names.
3. Restart the layout software.
4. Reload your ASCII MPP template file.

Problems with Editing and Entering Points

Mistakes Entering Points

If you enter a point incorrectly, you can delete it.

To delete the last point entered,

1. Press `Backspace`.

   The last point you entered is deleted.
2. Click to enter a new point.

Click to enter a point. | Press `Backspace`. | The pointer remains in the same place, but the point is deleted. | Move the pointer and click to enter a new point.

If you finish creating an object and then find it is incorrect, remove the object by undoing the last command (choose *Edit – Undo*).

Entered Points on the Wrong Layer

You create objects on layers that represent the layers of your physical design. The Layer Selection Window controls which layer the object will be created on.

© 2020 All Rights Reserved.

➤ If you notice you are creating objects on the wrong layer, click the layer you want in the *Layers* assistant.

The current entry layer changes to the one you chose.

If you finish creating an object and notice it is on the wrong layer, change the layer for that object by doing either of the following:

- To move the object to a new layer, use the Move command.
- To change the object's layer property, use the Properties command.

Cannot Move Objects or Enter Points As Required

You might need to change the snap mode if you see the following behavior:

- You cannot enter points to create or recreate objects
- You cannot move or stretch an object

To change this behavior, change the snap mode by doing the following:

1. While the command is active open the pop-up options form by pressing `F3`.
2. Choose the snap mode you want.

To change the create and edit snap mode defaults,

1. Choose *Options – Display*.

   The Display Options Form appears.
2. Choose a new value for the *Create* or *Edit Snap Modes*.
3. Click *OK*.

Object Moves Instead of Stretching

If an object moves when you expected it to stretch, you probably selected the whole object rather than its edge or corner. To stretch an object, you must select only the edge or corner you want to stretch.

If you want to select edges or corners (vertexes), you must first set partial selection on. Partial selection lets you select edges or corners with the selection box.

To toggle partial selection on or off,

© 2020 All Rights Reserved.

➤ Press `F4`.

When partial selection is on, the Status Banner or the Status Toolbar show a `P` in front of the number of selected objects.

```
(P) Select: 1
```

If you select whole objects along with edges...

...the whole objects move.

The stretched and moved objects

If you select only partial objects (edges or corners)...

...all objects stretch.

The stretched objects

Object Changes Shape When Resizing

The *Size* command and the *Grow By* option for the *Layer Generation* command each let you resize an object by stretching its corners. Sometimes enlarging or reducing the object causes it to change shape.

© 2020 All Rights Reserved.

For example, if you use *Size* to enlarge the following polygon by 1 unit, each vertex stretches in both the X and Y direction by 1 unit, and the notch in the polygon disappears. The resulting shape is stored as a rectangle, rather than a polygon.

Original shape

After using *Size* to enlarge the shape by 1 unit

To cancel the changes you made by resizing a shape,

➤ Immediately choose *Edit – Undo*.

Cannot Paste an Object

If you use *Yank* and *Paste* to cut through the hierarchy and place parts of an instance into a different library, the new library must define all the layers you yanked. If it does not, you see the message

```
Shapes on invalid or invisible layers not pasted.
```

The *Yank* command lets you copy an area of the cellview, cutting through layers of hierarchy. You can select just part of an instance to copy with *Yank*, then use *Paste* to place the parts of the instance you copied into any cellview.

Pcell Parameters Do Not Appear

The parameters for parameterized cells (Pcells) appear in the Create Instance form as you place an instance of the Pcell.

When you type the name of a cell into the Create Instance form, you must tell the system you are done before the Pcell parameters will appear.

➤ If Pcell parameters do not appear in the Create Instance Form, press `Tab` to tell the system you are done typing the cell name.

© 2020 All Rights Reserved.

If the form is at the bottom of your screen, the parameter section can scroll off the screen.

➤ If the Create Instance form is at the bottom of your screen, move the form up to see the parameters.

If you use Component Description Format (CDF) C-level function parameters for your library or for any cells in the library, and your Pcell was compiled in version 4.2.1 or earlier, the CDF parameters suppress the Pcell parameter display.

➤ To recompile Pcells created in version 4.2.1 or earlier, choose *Pcell – Compile – To Pcell.*

Cannot Edit the Properties of Selected Instances

When multiple instances are selected and the common button is on, some fields may be grayed out and uneditable. This is because any instance parameters which have Component Description Format (CDF) SKILL function callbacks cannot be edited, unless all of the instances are the same component (in the case of Pcells the same super master) and have identical parameter values. Instances of different components without identical parameter values must be edited individually to ensure consistent data.

Mouse and Cursor Behavior

Right Mouse Button Does not Work

By default, the right mouse button works as follows:

- To display the pop-up context menu, click the right mouse button once
- To zoom in, press and hold the right mouse button and create a box
- To zoom out, press the `Shift` key and hold the right mouse button and create a box

If the right mouse button does not do any of these tasks, it is probably set to create strokes. A stroke is a pre-programmed figure you can create to start a command.

➤ To cancel the stroke-creation capability, you must exit and restart the Cadence® software.

If strokes are still on when you restart, then the commands to load strokes are included in your `.cdsinit` file.

To remove the stroke commands from your `.cdsinit` file,

© 2020 All Rights Reserved.

1. Use a text editor such as `vi` to open your `.cdsinit` file (usually located in your current or working directory).
2. Look for these lines in your `.cdsinit` file:

```
load(prependInstallPath( "etc/sted/stroke.il"))
load(prependInstallPath( "etc/sted/defstrokes.il"))
hiLoadStrokeFile("def.strokes" "Layout")
```

3. Type a semicolon (`;`) in front of each line to comment it out.
4. Save the edited `.cdsinit` file.
5. Exit and restart the Cadence software.

Pointer Changes Shape

The mouse pointer changes shape to show you how you can use it to select, move, or stretch objects. If you see the pointer change shape, this is not an error.

Cursor Sticks to Objects

If the mouse cursor seems to jump to or stick to objects in your cellview, the gravity setting is most likely on.

Gravity causes the mouse cursor to automatically snap to objects and grid points in the cellview. You typically use gravity when you want to snap the cursor to specific types of objects as you create them.

To turn gravity off,

➤ Move the mouse into a layout window and press `g`.

If this does not work, your system administrator changed the default key bindings. Use the Layout Editor Options Form to set gravity.

Cursor Snaps to the Wrong Object

If you want to use gravity but the cursor snaps to the wrong object, you might need to change the gravity settings. Use the Layout Editor Options Form to change the gravity settings.

© 2020 All Rights Reserved.

Cursor Does not Snap to a Point

If gravity is set off and the cursor does not snap to the grid point you want, the snap spacing might be set too wide. The X and Y snap spacing settings control the number of units at which the cursor snaps to the grid. Use the Display Options Form to change the snap spacing settings.

Selection Tips

Selectable Layers

All layers are selectable if they are included and set to selectable and valid in the `techDisplays` subsection in the technology file, whether or not they appear in the *Layers* assistant. If they do appear in the *Layers* assistant, you can turn on or off their selectability or validity in the Palette Assistant.

How to Make a Layer Selectable

In the Palette Assistant, you can turn off selection for one or more layers. When you do, a layer that is not selectable appears with its name shaded in the *Layers* assistant.

To make a layer selectable,

➤ Click right on the layer name in the *Layers* assistant.

The only objects you can select in a cellview are on selectable layers in the *Layers* assistant.

Note: To add a layer to the *Layers* assistant, choose *Edit – Set Valid Layers* in the *Layers* assistant. If you are using the leLswLayers section of the Technology File, only the layers listed in that file appear in the Show Selected Set Form.

© 2020 All Rights Reserved.

Selection During Edit In Place

If you are editing a cell instance in place, you can select and edit only those objects within the instance. Even though you can see objects around the instance, you cannot edit them.

- To check to see if you are editing in place, look at the cellview banner to see what instance you are editing.

  If the name in the banner does not match the name of the current cellview, then you are editing a cell in place.

- To return to editing the current cellview, choose *Edit – Hierarchy – Return*.

How to Select Objects in a Dense Design

If you click objects in a dense design and the layout editor does not select the object you want, try any of the following:

- If objects share the edge you chose, click again in the same place.

  The layout editor selects another of the objects. (The layout editor toggles between only two possible objects.)

- If possible, move the cursor closer to another edge of the object and click.

- If possible, zoom in on the edge you want to select.

  Use the *View – Save/Restore – Previous View* command to zoom back out again after selecting the object.

- Use the Palette Assistant to turn off selectability for:

  - Layers that overlap the object
  - Pins or instances (if a pin or instance overlaps the object)

© 2020 All Rights Reserved.

Instance Selectability

If you cannot select an instance, check to see if *Inst* in the Palette Assistant is turned off.

➤ If *Inst* is turned off, click to turn it on.

Problems Selecting Copied Objects

If you move or copy an object into a cellview in a different library and then cannot select the object,

1. Click in the new cellview.
2. Look for the name of the layer for the object in the *Layers* assistant.
   - ❑ If the layer name is unselectable (shaded), click right to make it selectable.
   - ❑ If the layer name does not appear in the *Layers* assistant, the layer either is not defined for this library or has been removed from the *Layers* assistant.

If the layer is not defined for the library, cancel the copy by doing one of the following:

➤ Choose *Edit – Undo* in the original cellview.

➤ Choose *File – Discard Edits* in the new cellview to delete all edits you made since you last saved the cellview.

If the layer has been removed from the *Layers* assistant, replace it by choosing *Edit – Set Valid Layers*.

Mouse Buttons for Zooming and Selecting

If you create a selection rectangle and the image in the cellview is enlarged or reduced, you drew the rectangle using the right mouse button instead of the left mouse button. Dragging right starts *Zoom In* and *Zoom Out* commands.

- Creating a rectangle with the right mouse button enlarges the cellview image.
- Creating a rectangle by pressing `Shift` with the right mouse button reduces the cellview image.

If you inadvertently zoom in or out instead of selecting objects,

1. Choose *Previous* from the *View – Save/Restore – Previous View* menu to re-display the cellview as it appeared before you zoomed in or out.

© 2020 All Rights Reserved.

2. Click and drag to create a rectangle to select objects.

How to Select Invisible Objects

If a NA outlines an area of your cellview that appears blank, you probably turned off the visibility of all layers that appear in an instance and selected that invisible instance.

For example, if you clicked *NV* in the *Layers* assistant to set all layers invisible, but *Instances* in the *Objects* assistant is still on, you can select the invisible instances.

To select invisible objects,

➤ Do one of the following:

- ❑ Set *Instances* off in the *Objects* assistant so that instances are not selectable.
- ❑ Click *AV* in the *Layers* assistant to make all layers visible.

Window Display Tips

How to Make a Design Layer Visible

If you cannot see one of your design layers, it is probably set to be invisible. A layer that is invisible appears with its layer color shaded in the *Layers* assistant.

To make a layer visible,

1. Click middle on a shaded layer.
2. Choose *View – Redraw* to see the results of your changes.

How to See Instance Contents

If you see only an outline of each instance instead of the details of objects inside the instance, change the display level settings.

To turn on instance detail,

➤ Press `Shift-f`.

`Shift-f` is the bindkey equivalent of using the Display Options form to set the display levels to 0 through 32.

© 2020 All Rights Reserved.

Displaying Instances You Are Editing

If you are editing a cell in place, you can set the window to display only the instance you are editing.

To see the rest of the instances,

1. Choose *Options – Display*.

   The Display Options Form appears.

2. Set *EIP Surround* on.

3. Click *OK*.

How to See What the Search Command Finds

If you use the command to search for or select objects and your cellview is zoomed in, the command might find an object that is not visible in the current window.

To zoom the cellview so the objects searched appear in the window:

➤ In the Find/Replace Form, set *Zoom To Figure* on.

Now each time you click *Previous* or *Next*, the cellview is redrawn so that the object is centered in the window.

Problems with Stretching Parameterized Cells

Pcell Stretch Handles Are Not Visible

If you cannot see handles on stretchable Pcells, the option might be turned off or the stretch handle layer might not be visible.

You can check whether Pcell stretch handles are set to display with the layout editor Display Options form or with the graphic and layout editor environment variable `displayStretchHandles`. The default is to display stretch handles. You cannot stretch a handle unless it is displayed.

If the *Stretch Handles* option is turned on, and you still cannot see stretch handles on your Pcell, the layer containing them might not be visible. See Making the Stretch Handle Layer Visible.

© 2020 All Rights Reserved.

Turning On Stretch Handle Display in the Options Form

To turn on *Stretch Handles* in the layout editor Display Options form, do the following:

1. Choose *Options – Display*.
2. In the Display Options form, if necessary, turn on *Stretch Handles* and click *OK*.

Turning on Stretch Handle Display in the CIW

To turn on `displayStretchHandles` environment variable, do the following:

1. Type the following in the CIW:

   ```
   window = hiGetCurrentWindow()
   ```

2. To display stretch handles type the following in the CIW:

   ```
   window~>displayStretchHandles = t
   ```

Making the Stretch Handle Layer Visible

The default layer for stretch handles is `y0 drawing`. To make that layer visible, do the following:

1. In the *Layers* assistant, choose *Edit Valid Layers.*
2. In the Show Selected Set Form, click *y0 dg* and click *OK*.
3. In the *Layers* assistant, click *AV* to make all layers visible.

   To see the results of your changes, you need to redraw the window.

4. Choose *View – Redraw.*

Note: If you want to use a different layer for stretch handles, you can change the layer with the `stretchHandlesLayer` environment variable.

Difficulty Selecting Pcell Stretch Handles

If you have trouble selecting stretch handles, use area selection rather than point selection or try turning off the layout editor *Gravity On* option. To turn off gravity, do the following:

1. Choose *Options – Editor.*
2. In the Layout Editor Options form, click to turn off the *Gravity On* option.
3. Try again to select a stretch handle.

© 2020 All Rights Reserved.

14

Dialog Box Messages

This chapter contains these topics:

- About Messages
- List of Error Messages

© 2020 All Rights Reserved.

About Messages

The Virtuoso® Layout Suite L layout editor displays error and other messages in dialog boxes. A dialog box is a Cadence® window that displays a warning or other information about the current command.

There are four types of messages:

Error messages display a international *No* symbol and indicate the layout editor cannot complete a command you have started.

Warning messages display an exclamation point and warn you of a result you might not have anticipated.

Informational messages display an *i* symbol and provide information about a command's progress.

Questions display a profile of a head and ask you to make a choice.

Closing Dialog Boxes

Most information, error, and warning boxes have one button, *Close*.

You must click *Close* to close the dialog box before you can continue editing.

Using Question Dialog Boxes

Question dialog boxes let you choose, for example, between changing data or leaving it unchanged.

- To change data, click *Yes*.
- To leave data unchanged, click *No*.
- To see the list of error and warning messages, click *Help*.

© 2020 All Rights Reserved.

List of Error Messages

The following list shows the error and warning messages you might see in the CIW or in a dialog box while using the layout editor.

The messages are listed alphabetically. Several messages begin with variables (a cellview name you typed, or the number of shapes you were editing). Those messages are listed first.

Error Message	Explanation and Solution
"cellname viewname" already exists in library *"name"*. Do you want to overwrite it?	You tried to save this cellview with a name that exists. ■ Click *Yes* to write over the existing file. ■ Click *No* to cancel the *Save As* command.
"cellname viewname" is being edited in a window. Do you want to overwrite it?	You tried to create a cellview with the same library, cell, and view name as one that is open for editing. ■ Click *Yes* to write over the existing file. ■ Click *No* to cancel the *Create File* command.
"cellname" is not a valid cell name.	The cell name field in the form is blank. Click *Browse* in the form to use the Library Browser and select the correct cell name. To create a new cell, choose *File – New – Cellview* in the Command Interpreter Window (CIW).
"libraryname cellname viewname filename" already exists. Do you want to overwrite it?	You tried to create a cellview with a name that exists. ■ Click *Yes* to write over the existing file. ■ Click *No* to cancel the *Create File* command.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
"*Number*" shapes not modified because they would become malformed.	When editing a group of shapes, you tried to delete points or edges that would cause the shapes to become invalid. For example, a path must have at least two points, and a polygon must have at least three points.
"*Number*" shapes not modified because they would be illegal.	You stretched a shape to a zero area, which would delete it. You cannot use *Stretch* to delete shapes.
"Shapenames" shapes not modified because they would be self-intersecting.	You stretched the shape so it overlaps itself.
"Shapenames" shapes were not modified because they would become malformed.	You tried to delete part of an object. To delete an object, select the entire object.
A Template: "*name*" already exists, would you like to replace it?	You have created a multipart path template with the same name as one which already exists. ■ Click *Yes* to overwrite the existing template with your newly created one. ■ Click *No* to cancel the Save Multipart Path Template command.
Apply the changes to the object?	You clicked *Next* in the Edit Properties form after making changes to a device in a chain, but before clicking *Apply*. You must click *Apply* before continuing to edit the chain.
Apply the command first to find figures that meet the current search criteria.	You clicked *Add Select*, *Select All*, *Replace*, or *Replace All* in the Search form before clicking *Apply*. You must click *Apply* to search for objects first.
Cannot create a pin because the path for the pin is non-orthogonal.	The *autopin* option for *Pin* only works on path ends parallel to the X or Y axis.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Cannot create a recursive instance placement.	You tried to use *Create – Instance* to place an instance of the master cell you are editing in this cellview into this cellview, or you tried to use *Make Cell* to create and place a new cell with the same name as the current cellview.
P&R boundary already exists in this cellView, do you want to replace it?	You tried to create a P&R boundary in this cellview when one already exists. ■ Click *Yes* to write over the existing boundary. ■ Click *No* to cancel the command.
Cannot create view file "*name*."	The *Show Selected Set*, *Summary*, and *Tree* commands create and open a temporary file in `/tmp` that contains the data displayed by these commands. You must have write and read access to `/tmp` for these commands to work. See your system administrator or become root and use `chmod` to set the `/tmp` directory permissions.
Cannot descend into a techfile device.	You tried to descend into a device that is defined in the technology file. To edit this device, you must edit its definition in the technology file.
Cannot descend into instance "*name*" because it is unbound.	The library for the instance you chose with *Descend* ■ Has been deleted ■ Is not in your library path; add it to your `cds.lib` file
Cannot edit in place a parameterized cell.	You chose a parameterized cell (Pcell) to edit with *Edit In Place*. You cannot edit Pcell instances in place because a Pcell instance is not identical to the Pcell master. You can edit the Pcell parameters for the instance using the *Edit – Basic – Properties* command.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Cannot modify figure, because cellView not open with write permission.	You tried to use the *Replace* or *Replace All* buttons in the Search form to edit objects you found with *Search*, but the objects are in cell instances in the current cellview. To open an instance to edit, use *Edit In Place*.
Cannot open cellview "*cellname viewname*" in library "*libname*"	You do not have write permission to either the cellview or the library you typed in the Make Cell form. You need to change the permissions for the cells or libraries.
Cannot open file "*name*" for read. Cannot open file "*name*" for write.	The *Summary* and *Show Selected Set* commands create and open a temporary file in `/tmp` that contains the data displayed by these commands. You must have write and read access to `/tmp` for these commands to work. See your system administrator or become root and use `chmod` to set the `/tmp` directory permissions.
Cannot open library *"name"* for writing.	You do not have permission to write to the library name you typed in the Save As form. ■ Type a different library name. ■ Have the owner of the library change the permissions.
Cannot open file "*name*" for reading.	You tried to save the settings in the Display Options, Layout Editor Options, or Submit Plot form to an ASCII file, but you do not have write access to the directory or file. ■ Type a new filename or include a path to a directory to which you have write access. By default, the layout editor tries to save the file to your current directory.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Cannot select figure that is not in current cellView.	You set the *Search* command to search for objects inside instances in the current cellview, then tried to use *Search* to select an object found in an instance. Search cannot select objects inside instances. To open an instance and select an object inside it, use *Edit In Place*.
Cannot set view file "*name*."	The *Summary*, *Show Selected Set*, and *Tree* commands create and open a temporary file in `/tmp` that contains the data displayed by these commands. You must have write and read access to `/tmp` for these commands to work. See your system administrator or become root and use `chmod` to set the `/tmp` directory permissions.
Cannot write to file *"name"*.	The filename you typed in the Submit Plot form is incorrect for one of the following reasons: ■ The directory does not exist. ■ You do not have permission to write to the directory. ■ You do not have permission to write to the file.
Cell "*name*" does not exist in library.	The cell name you typed in the Create Instance form is incorrect. Click *Browse* in the form and use the Library Browser to select a different library or cell name.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Cellview "*cellname viewname"* already exists in library *"libname*." Do you want to overwrite it?	While using *Make Cell*, you typed the name of a cell and view that already exists in the library you chose. ■ Click *Yes* to overwrite the original cell with the new objects you have selected with *Make Cell*. You can choose this only if you have write access to the original cell. ■ Click *No* to avoid overwriting the cell. Type a new cell name in the Make Cell form.
Cellview "*cellname viewname*" does not exist in library.	The cell name you typed in the Create Instance form is incorrect. Click *Browse* in the form and use the Library Browser to choose a different library or cell name.
CellView is already read only.	You tried to set *File – Make Read Only* when the mode was already read only.
CellView is already editable.	You tried to set *File – Make Editable* when the mode was already edit.
Chopper polygon area cannot be zero.	You clicked and then double-clicked on the same point when creating the polygon to cut an object with *Chop*.
Chopper polygon is self-intersecting.	You drew a self-intersecting polygon to chop an object. This shape is invalid. It is a self-intersecting polygon.
Chopper rectangle area cannot be zero.	You clicked the same point twice or drew a vertical or horizontal line when creating the rectangle to cut an object with *Chop*.
Circle not created because it would have zero area.	You clicked the same point twice when creating a circle with *Circle*.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Via "*name*" does not exist in library "*name*" using "*contact name*."	You used `envSetVal("contactName" contactType)` to set the default type of via to place with *Via*, but `contactType` (the name of a via to place) is not defined for this library.
Could not save cellview *"libraryname cellname viewname reason"*.	You tried to save a cellview. The software could not for the reason stated.
Delete figure?	You tried to delete a selected object. ■ Click *Yes* to delete the figure. ■ Click *No* to cancel the *Delete* command.
Deleting the layer purpose pair "*layer* (*purpose*)" (or layer purpose pairs) might cause errors in other technology file classes that reference it (or them). Delete "*layer* (*purpose*)" (or the layer purpose pairs) anyway?	You selected one or more layer-purpose pairs in the Layer Purpose Pair Editor form and clicked on the *Delete* button. ■ Click *OK* to delete the selected layer-purpose pair or pairs. ■ Click *Cancel* to cancel the *Delete* command.
Delta X is less than width of master's bBox.	The value you typed for the *Delta X* field in the Create Instance form would cause the instances to overlap.
Delta Y is less than height of master's bBox.	The value you typed for the *Delta Y* field in the Create Instance form would cause the instances to overlap.
Descending into a parameterized cell. Is this ok?	You chose an instance of a parameterized cell (Pcell) to edit with *Descend*. Because a Pcell master does not always resemble the instance, this message appears to remind you that you are about to open a Pcell master. ■ Click *OK* to descend into the master view of the Pcell. ■ Click *Cancel* to stop *Descend*.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Directory does not exist. Create directory "*name*"?	You saved to a directory that does not exist. ■ Click *Yes* to create the directory. ■ Click *No* to not create the directory.
Do you want to save your changes?	You tried to set *File – Make Read Only* without writing your edits to disk. ■ Click *No* to discard your edits. ■ Click *Yes* to save your edits.
Donut not created because inner radius equals outer radius.	The distance between the first and third point you clicked equals the distance between the first and second point.
Donut not created because of zero inner radius.	The third point you clicked is the same as the first point.
Ellipse not created because it would have zero area.	You clicked the same point twice when creating the bounding box of an ellipse.
Error found while reading "*file line*."	There is a syntax error on the line in the file you tried to load using the *Load From* button in the Display Options, Layout Editor Options, or the *Load* button in the Submit Plot form. See the *Virtuoso Layout Suite SKILL Reference* for details about Cadence SKILL language syntax.
Figures cannot be modified, because you do not have write permission for this cellView.	You tried to use the *Replace* or *Replace All* buttons in the Search form to edit objects you found with Search, but the current cellview is open in read-only mode. Use *File – Make Editable* to open the current cellview in edit mode.
Figures not in current cellView will not be selected.	You set the *Search* command to search for objects inside instances in this cellview, then tried to use *Search* to select objects it found in the instances. *Search* cannot select objects inside instances. To open the instances and select objects inside them, use *Edit In Place*.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
File *"name"* already exists. Do you want to overwrite it?	You tried to save this cellview with a name that exists. ■ Click *Yes* to write over the existing file. ■ Click *No* to cancel the *Save* command.
Find failed. *"reason"*	The *Verify – Markers – Find* command could not find any markers for the reason stated.
First or last segment of created path has length less than or equal to half the path width.	This usually happens when you complete a path and one of the path segments is smaller than half of the path width. You also can see this message when you reshape a path and the result is one of the path segments becomes smaller than half of the path width.
First or last segment of modified path has length less than or equal to half the path width. Use "Cancel" to interrupt the command.	Path modification was incorrect. This usually happens when you stretch one or more paths and the stretch causes one or more of the path segments to be smaller than half of the path width. ■ Click *OK* to close the dialog box. If you have more than one incorrect path segment, a dialog box opens for each one. After you close one dialog box, another one opens for the next incorrect path segment. You can click OK until all of the dialog boxes are closed. ■ Click *Cancel* to close the dialog box and interrupt the command. Nothing more will be stretched, and the warning process stops.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
hiToggleEnterForm(), normally bound to the F3 key, will show or hide the form associated with an active enterfunction if that enterfunction has an associated form. Do you wish to see this dialog again when no enterfunction is active or no form exists to toggle?	You pressed the `F3` key expecting to see the option form for the command running, but an option form does not exist for the command. ■ Click *Yes* and the box appears every time you press `F3` for commands that do not have option forms. ■ Click *No* and the box does not appear again during the editing session.
Instance name *"name"* contains a syntax error.	The instance name you typed in the Create Instance form is not allowed. The instance name syntax, a form of BNF (Backus Naur Format) notation, is `instName ::= baseName [ < number [ : number ] > ]`
Instance base name "*base name*" is not unique in this block. Instance member name "*member name*" is not unique in this block. Instance name "*instance name*" is not unique in this block.	The base, member, or instance name you typed in the Create Instance form is already used in this design cellview. See *"Instance name 'name' contains a syntax error"* for the instance name syntax.
Invalid window: *"windowID"*.	The window ID is incorrect.
Layout editing capability is not enabled.	You do not have the Cadence license for reading and viewing or changing and saving layout views
Layout must be opened with write permission.	You cannot use the commands on the *Create* or *Edit* menus if the layout cellview is opened in read-only mode. Use *File – Make Editable* to open the cellview in edit mode.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
leHiCreateMPP: width not specified and no default minWidth rule found for layer *"layer name"* Layout default pathWidth is being used	You have used the *Create – Multipart Path* command on a selected layer that does not have the minimum width defined in the technology file. The default path width is the last width used if you created a path during the same editing session. If you did not create a path earlier, the system applies the system-defined default path width.
Library "*library name*" does not exist.	The library name you typed in the Create Instance or Make Cell form is incorrect. ■ Click *Browse* in the form and use the Library Browser to select the correct library. ■ Correct your library path.
Modifying "object" throughout hierarchy. Is this okay?	You tried to replace a value for an object you found with *Search*, and the object is inside an instance in the current cellview. Any edits you make to the instance in this cellview also affect the master cell for the instance. ■ Click *Yes* to replace the object through the hierarchy. ■ Click *No* to cancel replacing the object.
Mosaic/Donut/Dot is not supported in the Rotate command. De-select it.	You tried to use the *Rotate* command on either a mosaic, donut, or dot pin. To rotate these objects, use the *Move* command.
No cellviews were refreshed.	No masters of cell instances in the current cellview have been edited and saved since you opened the current cellview, so the *Refresh* command did not update any of the instances.
No vias defined in library "*name.*"	You tried to place vias with *Via* or search for via with *Search*, but no vias are defined in the technology file for this library.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
No current window exists.	You tried to use a command that needs a graphic editor window.
No figures found that meet the current search criteria.	*Search* could not find any objects in this cellview that match the criteria you set in the Search form.
No figures selected.	You tried to use the *Make Cell* command without selecting the objects you want to make into a new cell.
No instances selected to be flattened.	The objects you chose to flatten are not cell instances.
No markers match the search criteria.	No markers were found as specified. Check your search criteria.
No modifications have been made.	You chose *Discard Edits* to delete any edits you made since the last time you saved the current cellview, but you have not edited this cellview since the last *Save*.
No more figures.	While using *Search*, you clicked *Previous* when there is no previous object in the group of objects *Search* found or you clicked *Next* when there is no succeeding object in the group of objects *Search* found.
No objects selected.	You chose the *Show Selected Set* command without first selecting a group of objects.
No shapes merged.	You did not select two or more overlapping shapes on the same layer to merge with *Merge*. Continue clicking on shapes to select the objects you want to merge.
No shapes selected.	You tried to use the *Layer Generation* command without selecting any objects on the layers you want to edit.
No shapes selected to be sized.	You chose the *Size* command without first selecting the objects you want to resize.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
No shapes selected were modified.	You tried to modify a corner, but there were no polygon or rectangle corners selected.
No views saved for this window.	You tried to restore a saved image for this window, but there are no saved images. Use the *View – Save/Restore – Save View* command to save an image.
Not a layout window.	You typed the SKILL version of a layout editor command in the CIW, but your current design window is not a layout view. Try clicking in the layout window first. If there is no layout view created for this cell, choose *File – New – Cellview* in the CIW to create a layout view (view name layout).
Nothing to paste.	You tried to use *Paste* without first using *Yank* to copy the objects you want to paste.
One or more of your templates already exist in techfile. Replace them all?	You have created one or more multipart path templates with the same name as one or more which already exist in your technology file. ■ Click *Yes* to overwrite the existing templates with your newly created ones. ■ Click *No* to cancel the Save Multipart Path Template command.
Only shapes can be sized.	You chose an instance or mosaic to resize with *Size*, but *Size* can only edit shapes. You can change the size of a selected instance or mosaic by using *Properties* and editing the *Magnification* field. **Note:** Magnification is not supported in OpenAccess.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Open read-only cells for edit to replace *"cellname"* ?	The *Search* and *Replace* command found something in a cell that is opened in read-only mode. ■ Click *Yes* to open the cell for editing. ■ Click *No* to skip this replacement.
Shape is already on net "*name.*"	The shape you tried to add to a certain net is already on a net. Shapes can be added to only one net.
Path not created because it would be self-intersecting.	You tried to create a self-intersecting path with *Path* or *Stretch*. Paths are self-intersecting if any edge of the path overlaps another edge of the path. A self-intersecting path
Path width is greater than maximum width.	You typed a width value in the Path form that is greater than the `maxWidth` constraint defined for the layer in the technology file.
Path width is less than minimum width.	You typed a width value in the Path form that is less than the `minWidth` constraint defined for the layer in the technology file.
Pin not created because it would be self-intersecting.	Polygon pins follow the same rules as polygons. They cannot be created as self-intersecting shapes.
Enter file name to save the display information change.	You said you wanted to save your changes before exiting. ■ Type a filename.
Point at the reference point for rotation.	Click the point you want to use as a pivot point for the rotation.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Warning in CIW Polygon would be self intersecting, point rejected	You tried to create a self-intersecting polygon with *Create Polygon* or *Stretch*. When possible, the *Create Polygon* command beeps and recreates the polygon so it is not self-intersecting; otherwise, the command returns the dialog box message. This shape is invalid. It is a self-intersecting polygon.
Warning in CIW Polygon would be self intersecting, point rejected	You drew a self-intersecting polygon to chop an object. This shape is invalid. It is a self-intersecting polygon.
Discard edits?	You chose *Discard Edits* to discard all the edits you made since the last time you saved this cellview. ■ Click *Yes* to discard all edits you made since the last *Save*. ■ Click *No* to cancel the *Discard Edits* command and keep your edits.
Rectangle must intersect shape.	The rectangle you drew with *Reshape* does not intersect the object you selected to edit.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Rectangle must intersect shape at no more than two places.	The rectangle you drew with *Reshape* intersects the object you selected to edit in more than two places. ■ Close the dialog box, and create a new rectangle. The reshaped rectangle intersects at more than two sides.
Rectangle not created because it would have zero area.	You clicked on the same point twice or drew a vertical or horizontal line when creating a rectangle.
Reference point is not active.	You pressed the – (minus) key on the keypad to make the reference point inactive and then tried to press `Shift-R11` (the `5` key on the keypad) to move the cursor back to the reference point. Check the status banner; if the *dX* and *dY* fields on the status banner show no value, the reference point is inactive. These fields normally show the delta distance along the X and Y axes between the current point and the reference point. ■ Press the + (plus) key on the keypad. The system prompts you to enter a reference point.
The Selector Parameter name "*`name`*" matches a parameter of the child Pcells.	You chose a selector parameter name that matches one of the child Pcell parameters. Change the selector parameter name to one that does not match any of the child Pcells.
Send Plot Only to File option must be set for \%s\ plot.	The plotter specified in the Plot Options form requires that the plot be sent to a file. ■ Select a different plotter or type a filename for the plot.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Send Plot to File requires an output file name.	You tried to plot to a file but did not specify a filename. ■ Type the filename in the Plot Options form.
Shape not created because it would be self-intersecting.	You tried to create a self-intersecting shape with *Reshape* or *Split.* This shape is invalid. It is a self-intersecting polygon.
Shape not modified because it would become malformed.	You tried to delete a point or edge that would cause the shape to be invalid. For example, a path must have at least two points, and a polygon must have at least three points.
Shape width must be greater than zero.	You typed zero or a negative number for the *Width* value in the Path form. ■ Type a positive number in the *Width* field.
Some read-only cells could not be opened to replace *"cellname"*. Continue?	The software could not open read-only cells for editing.
Specify the library name.	The *library* name field in the form is blank. ■ Type a library name.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
Terminal name "*name*" already exists with I/O type "*old I/O type*." Do you want to change the I/O type to "*new I/O type*?"	You are trying to place a pin with a terminal name that matches the terminal name for one or more pins already placed in this cellview. The I/O (Input/Output) type for the other pin does not match the I/O type for the pin you are now placing. ■ Click *Yes* to change the I/O type for this pin and all other pins with the same terminal name. ■ Click *No* to keep the I/O type for this pin the same as the I/O type for all other pins with the same terminal name.
Terminal name "*name*" contains a syntax error.	The terminal name you typed in the Create Pin form is not allowed. The terminal name syntax, a form of BNF (Backus Naur Format) notation, is `termName ::= baseName [ < number [ : number ] > ]`
The cellview for the command is no longer available.	The command received a bad cellview ID.
The cellView could not be opened for edit. Do you want to open it for read?	You tried to open a read-only cellview in edit mode using the *File – Open* command. ■ Click *Yes* to open the cellview in read-only mode. ■ Click *No* to cancel the *File – Open* command. To open the cellview in edit mode, you must change the permissions in the directory.
The current cellView is open with read-only permission.	You tried to save the current settings in the Display Options form to this cellview, but you do not have write permission to the cellview. ■ Use the *File – Make Editable* command to reopen the cellview in edit mode.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
The design could not be reopened because the data on disk has been updated by others. Read in new data?	You tried to change the edit/read-only mode and the software found that some of the cellviews in the window do not have the current data. ■ Click *Yes* to update the cellviews. ■ Click *No* to continue with current data.
The following lists inconsistencies among techfiles used in this design: Top level library: *"techfilename"* Lower lever library: *"techfilename"* *"problem"* *"list"*	The cellview you are opening has conflicts between the top-level technology file and the lower-level technology file. ■ Edit the technology files listed to correct the listed problems.
The following masters are used in this cellview, but cannot be found in your search path: *"path"*	The masters could not be found in the path in your `cds.lib` file. ■ Edit the `cds.lib` file to include the missing master cellviews.
The range of the rotate angle value is [-360..360].	The value you typed in the Rotate form is not within the range. ■ Type a rotation angle between -360 and 360.
The rectangle area cannot be zero.	You clicked the same point twice or drew a vertical or horizontal line when creating a rectangle with *Reshape* or *Yank* or when creating a parameterized rectangle for a Pcell instance.
This cellview has been modified. Do you want to save your changes? Library: *"name"* Cell: *"name"* View: *"name"*	You tried to close a cellview window without saving your changes. ■ Click *Yes* to save the changes and close the cellview window. ■ Click *No* to not save the changes and close the cellview window. ■ Click *Cancel* to not save the changes and leave the window open.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
This command is only available for layout views.	You typed the SKILL version of a layout editor command in the CIW, but your current design window is not a layout view. ■ If there is no layout view created for this cell, choose *File – New – Cellview* in the CIW to create a layout view (view name layout).
Unable to convert ellipse to polygon. The number of points would exceed the 2047 limit. Try a numPoints value less than 2048.	The value of *Conic Sides* in the Layout Editor Options form is more than 2047. Change the number to 2047 or less.
View "*name*" does not exist in library "*library name*."	The view name you typed in the Make Cell form is incorrect. ■ Click *Browse* in the form to use the Library Browser and select the correct view name. Then choose *File – New – Cellview* in the CIW.
This is the first marker.	When viewing the first marker, you clicked the *Previous* button in the Find Markers form.
This is the last marker.	When viewing the last marker, you clicked the *Next* button in the Find Markers form.
Yank polygon area cannot be zero.	You clicked and then double-clicked on the same point when creating a polygon around objects you want to yank.
Yank polygon is self-intersecting.	You drew a self-intersecting polygon around the objects you want to yank. This shape is invalid. It is a self-intersecting polygon.
Yank rectangle area cannot be zero.	You clicked the same point twice or drew a vertical or horizontal line when creating a rectangle around objects you want to yank.

© 2020 All Rights Reserved.

Error Message	Explanation and Solution
You cannot save *“cellname viewname”* in library *“libraryname”*, because it is opened in “read only” mode.	You do not have permission to write to the library name you typed in the Save As form. ■ Type a different library name. ■ Have the owner of the library change the permissions.
You don’t have permission to write file *“filename”*.	You do not have permission to write to the filename you typed in the Save As form. ■ Type a different filename. ■ Have the owner of the file change the permissions.
You must specify a valid cell name.	You left the *Cell Name* field in the Create New File form blank. ■ Type a cell name.
You must specify a valid view name.	You left the *View Name* field in the Create New File form blank. ■ Type a view name.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

15

FinFET Support in Layout Editor

This chapter describes enhancements made to the layout editor to support FinFET devices. These enhancements include mechanisms to let you display the snap pattern grid in the layout canvas and automatically snap design objects to that grid.

This chapter contains the following topics:

- Snapping Objects to Local Snap Pattern Shapes
  - Snapping Objects to Snap Patterns
  - Placing Objects Inside the Fin Boundary
  - Controlling the Display of Snap Patterns
  - Turning On Snap Pattern Snapping
  - Snapping Objects
  - Snapping Objects and Shapes as Groups
  - Track Group Support
  - Commands Supporting Snap Pattern Snapping
- Snapping Objects to Global Snap Pattern Grid
  - Controlling the Display of Global Grid
  - Anchoring the Global Grid
- Controlling the Display of Snap Patterns in Palette
- Modes for Object Placement
  - Selecting Placement Mode
  - Abstract Mode Placement
  - Manual Mode Placement Override
- Create Custom Placement Area

© 2020 All Rights Reserved.

- Snapping for singleTrackCenter Snap Mode

© 2020 All Rights Reserved.

Snapping Objects to Local Snap Pattern Shapes

Virtuoso has been enhanced in a way that layout engineers working on FinFET processes improve their productivity. A new snap pattern has been defined to capture layers that have a grid-like nature and new constraints have been defined to capture a new class of width and spacing rules for FinFET processes. For more information, see snapPatternDefs.

The layout editor has been enhanced with visualization of the fin grids made inherent in FinFET technology, as well as automated snapping of shapes to the fin grid.

Snapping Objects to Snap Patterns

The high-level steps that a layout designer needs to follow for snapping objects to snap patterns are listed below:

- Create the snap pattern definition for the fin boundary in the ASCII technology file.
- Specify the fin pitch (`step`), fin width (`trackWidth`), direction (`stepDirection`), distance of the first fin center line from the lower edge (`offset`), and type (`type`) of fin - local or global.
- Specify the name of the layer, typically the *Active* layer, on which objects will snap to the fin grid (`snappingLayers`) and the required fin enclosures from the center line (`enclosures`). There can be multiple snapping layers (`layer`) in the snap pattern definition. Each layer can optionally have multiple purposes (`purposes`) and enclosure values (`enclosures`).
- Draw the fin boundary in the layout editor.
- Move a Pcell (including modgen Pcells), a via, an *Active* shape, or an instance over the fin boundary. These objects are snapped automatically to the fin grid by the layout editor.

Placing Objects Inside the Fin Boundary

The layout editor provides rapid placement of shapes, Pcells, instances, and vias in a fin boundary using a shape snapping algorithm. This ensures a correct-by-construction layout on the fin grid. Automatic alignment is achieved by an alignment engine, which contains the alignment semantics. The layout editor commands that are responsible for creating and moving Pcells, instances, and vias in a layout make use of the alignment engine to ensure placement correctness. The alignment engine accepts a set of shapes, and based on their overlaps with fin boundaries in the design, places them appropriately. The automatic placement requirements and their results are classified as following:

© 2020 All Rights Reserved.

- When a snap pattern shape or a Pcell, instance, via or fig group containing a snap pattern shape (the free figure) overlaps another snap pattern shape or a Pcell, instance, via, fig group containing a snap pattern shape, (the fixed figures), on the same LPP, then the alignment engine tries to align the snap pattern tracks and in the process determines the transformation that is applied to the free figure, such that the closest tracks of the overlapping pair are aligned.

Snap pattern - Snap pattern snapping: The alignment engine aligns the tracks.

The above figure shows the result of snap pattern shape pair alignment. No modification of the Pcell code is necessary to enable snapping of Pcell instances to the fin grid, since the snapping is done based on the existing fin boundary shape in the Pcell.

- A snap pattern definition specifies a list of LPPs. Shapes on these LPPs should snap to the snap pattern grid. When a shape on such a layer, or a Pcell, instance, via, fig group containing a shape overlaps a matching snap pattern shape then the engine computes

© 2020 All Rights Reserved.

the transformation that would be required to place the shape as defined by the applicable track enclosure.

Shape - Snap pattern snapping: The alignment engine aligns the snap shape in a via to the grid, based on the Track Enclosure

The above figure shows the result of shape to snap pattern alignment for a via that contains this shape. No change to the via is needed to enable snapping of substrate contacts to the fin grid because the snapping is done based on the existing shape on the *Active* layer in the via master.

Note: When you select or drag a snap pattern shape, these shapes behave in the same way as any other shape in the layout editor. You can also change the visibility and selectability of the snap pattern shapes from the Palette assistant.

Controlling the Display of Snap Patterns

The layout editor has been enhanced to visualize the fin grid or the full fin shapes inside fin boundaries. The snap pattern display lets you switch between full fin detail display, which is used to debug fin-alignment related DRC violations, and high-level fin boundary display which is used to focus on wiring and placement without being distracted by fin shapes.

You can display the boundary, the tracks, or both of them for snap patterns.

1. Choose *Options – Display*.

© 2020 All Rights Reserved.

The Display Options Form opens.

2. Select one of the following from the *Snap Pattern Display* list:

- *Full*: Displays the snap pattern (SP) boundary, tracks, fins, and width spacing patterns periods.
- *Tracks*: Displays the snap pattern boundary and tracks.
- *Boundary*: Displays only the snap pattern boundary.
- *Periods*: Displays only the width spacing pattern periods.
- *Periods and Tracks*: Displays width spacing pattern periods and tracks.

Full

Tracks

Boundary

3. Click *OK*.

© 2020
All Rights Reserved.

Tip

You can use the new leCycleSnapPatternDisplay SKILL function to cycle through the different snap pattern display types. You set the default value using the new snapPatternDrawingStyle environment variable.

Turning On Snap Pattern Snapping

To turn on *Snap Pattern Snapping* in the Layout Editor Options Form, do the following:

1. Choose *Options – Editor*.
2. In the Layout Editor Options Form, click to turn on the *Snap Pattern Snapping* option and click *OK*.

You can also set the `snapPatternSnapping` variable in your `.cdsenv` file, to turn the *Snap Pattern Snapping* option on at startup.

```
layout snapPatternSnapping boolean t
```

Video

You can view the video, *Snap Pattern Enabling*.

Snapping Objects

This section describes how objects are snapped to snap pattern shapes.

Snap Pattern Shape to Snap Pattern Shape Snapping

Two snap pattern shapes drawn on the same LPP snap to each other. Assume a FinFET process where snap pattern shapes are used to draw fin boundaries. If two Pcell instances containing fin boundaries are placed close to each other, such that their fin boundaries

© 2020 All Rights Reserved.

overlap, then Virtuoso will automatically align the fins of these snap patterns, as shown in the figure below.

When an instance containing multiple snap pattern shapes overlaps a snap pattern shape, then the instance is snapped based on only one of the snap pattern shapes. If the instance partially overlaps the snap pattern shape, one of the overlapping snap patterns is used for snapping.

P&R Boundary Snapping

When you create a P&R boundary with the *Create Method* as *From Selected Set* and the constraint allowedPRBoundaryDimensions is set in the technology file, the P&R boundary will be a multiple of the value specified in the `allowedPRBoundaryDimensions` constraint.

Instance Snapping

When an instance overlaps a snap pattern shape and the master does not contain a snap pattern shape on the same LPP, then the instance is snapped based on its shapes (assuming it contains a shape on an LPP that snaps to the snap pattern shape).

For example, consider a Pcell instance that does not contain a snap pattern, but it contains a shape on the *Active* layer. If this Pcell instance is moved over a snap pattern shape that

© 2020 All Rights Reserved.

applies to the *Active* layer, then the instance is snapped to the grid based on the shape on *Active* layer using the snap pattern definitions `enclosures` attribute.

Pcell master without snap pattern shape

Pcell instance snapped to the grid after the `Move` command

If there are multiple matching shapes, the snapping is done based on the first LPP union bbox that is found, where the relevant edge (lower for vertical snapping, left for horizontal snapping) overlaps the snap pattern shape.

If the snap pattern shape applies to shapes on multiple LPPs and the instance master contains shapes on several such LPPs, the snapping algorithm does snapping based on any one of the LPPs.

If the instance overlaps multiple snap pattern shapes that are *Active*, then the algorithm does snapping based on just one snap pattern shape.

© 2020 All Rights Reserved.

Snapping Instances in Modgens

A modgen comprises multiple Pcell instances in the form of a complex, structured array. The layout editor automatically snaps the Pcells in modgens to the snap pattern. The entire modgen block is, therefore, snapped to the top-level snap pattern shape.

Shape Snapping

Shapes that overlap a snap pattern shape after a copy or move snap to the grid.

For example, consider a snap pattern shape that applies to the *Active* layer and it is moved over a snap pattern shape. The shape is snapped to the grid based on the snap pattern definitions `enclosures` attribute.

Single Shape

Shape snapped to the grid after the `Move` command

A shape may overlap multiple snap pattern shapes and more than one of those may apply to it. In this case, the snapping algorithm does snapping based on one snap pattern shape.

© 2020 All Rights Reserved.

Note: If a shape overlaps multiple snap pattern shapes, the snapping is done only if all the snap pattern shapes are aligned.

When you are creating a rectangle and the rectOnSnapPatternGrid environment variable is enabled, a DRC correct rectangle is created.

Via Snapping

Custom layout designs have a stdVia, cdsViaDevice, cdsGenVia, and customVia. Except for customVias, none of the vias insert a snap pattern shape into the via subMaster. Therefore, vias are snapped based on the shapes they contain, instead of a snap pattern.

Also, when a via in a via stack is snapped to a snap pattern shape, the entire via stack is snapped. A via stack is snapped as a group.

Via and via stacks that overlap a matching snap pattern shape after a copy, move, or create operation are snapped to the grid.

For example, consider a FinFET process where snap pattern shapes are used for fin boundaries. When a substrate contact containing an *Active* shape is placed on a fin boundary, it is automatically aligned to the fin grid based on the `enclosures` of the snap pattern shape. The figure below contains a via with a plate layer on "*Active/drawing*". When

© 2020 All Rights Reserved.

the via is moved over a snap pattern shape that applies to all *Active* shapes, the via is automatically snapped to the fin grid based on the `enclosures` of the snap pattern.

Via with Active implant layer

Via is snapped to the snap pattern

When a `snapPatternDef` does not have any `snapPurposes`, it applies to all shapes on the `snapLayer`.

If the snap pattern applies to shapes on multiple LPPs and the via master contains shapes on several such LPPs, the snapping algorithm does snapping based on one of the LPPs.

If the via overlaps multiple snap pattern shapes that are *Active*, then the snapping algorithm does snapping based on one snap pattern shape.

Note: Snap pattern shapes on the same LPP should snap to each other. Snapping should be performed after a copy, move, or stretch operation or creation of an instance or via.

Note: You can use the environment variables snapPatternSnappingDepth and snapPatternSnappingDepthMethod to specify the snap depth and the method to be used to define the snap depth for snap pattern snapping, respectively.

© 2020 All Rights Reserved.

Snapping Objects and Shapes as Groups

The following objects in a cellview are snapped as groups:

- Members of a fig group.
- A chain of abutted MOS devices. If the chain is moved in such a way that snapPattern snapping is triggered after the move, all the devices on it are snapped together. This ensures that the abutment chain remains aligned.

  Note: For *Move* and *Copy* commands, if the *Chain Mode* is set to *All* then even if one device is selected, all the devices are snapped as a group.
- Members of a selected set, only if the selected set has a shape.

  Note: For *Move* and *Copy* commands, if the selected set has a shape, then the entire selected set is snapped as a group. If the selected set has less than 25 figures, the representative figure is the nearest snappable shape in the hierarchy inside the selected set. However, if the selected set has more than 25 figures, the representative figure is the nearest top-level figure in the selected set; the heirarchy analysis is not done.
- Vias in a via stack.

© 2020 All Rights Reserved.

Track Group Support

The layout editor has been enhanced to support track groups available in `snapPatternDefs`.

When a `snapPatternDef` has track groups, tracks are combined in groups. Within each track group, the existing `step` is used as the track space. The `period` of the `snapPatternDef` is computed from the group spacings and the number of tracks in each group.

Snap pattern shapes of the `snapPatternDef` snap to their period lines.

The shape snapping for track groups is supported both for global and local snap pattern definitions.

For snap pattern to shape snapping, the shape is snapped by having the lower or left edge of the shape enclose a track by the `trackEnclosure` value. The shape is snapped to the nearest track. To move a snapped shape from one track group to another, move the pointer by at least half the track group spacing.

© 2020 All Rights Reserved.

The shape snapping to snap patterns with track groups is depicted in the figure below.
f

© 2020
All Rights Reserved.

For snap pattern to snap pattern snapping, the computed period is used. The lower-left edge of the moving snap pattern is snapped to the closest period line of the target snap pattern. The track groups in snap pattern to shape snapping are depicted in the figure below.

Snap Pattern to Snap Pattern Alignment

Snapping of Instances in Detail Mode

Special behavior is needed when snapping instances containing snap patterns with track groups. Non-uniform snap patterns have a lower precedence than the uniform snap patterns. The snapping behavior is described below:

- If an instance has both uniform and non-uniform snap patterns, only the uniform snap pattern is used for snapping.
- If an instance contains only non-uniform snap patterns, snapping is done to period lines.
- If there are no non-uniform snap patterns, there is no change in snapping behavior.

© 2020 All Rights Reserved.

Important

Instance snapping based on snap patterns will usually be more accurate than shape-based snapping. As a result, if a snap pattern shape is found, shape snapping will be disabled for that direction. In this case, shape snapping will not be performed even if snapping is not possible for the snap pattern shape.

© 2020 All Rights Reserved.

Snapping Objects to Global Snap Pattern Grid

The layout editor has been enhanced to display a uniform fin grid over the entire design window. This has been enabled by adding a global snap pattern grid to the design. You can create a global snap pattern grid by specifying the value `global` in the `type` constraint of the snap pattern definition, as shown below.

```
snapPatternDefs(
        (GFG            ("CellBoundary" "global")
        'offset          0.024
        'type            "global"
        'step            0.048
        'stepDirection   "vertical"
        ...
```

The type constraint can have the value `global` or `local`. For more information, see snapPatternDefs.

A local snap pattern definition is drawn using a snap pattern shape in the layout. A global snap pattern definition can be activated through either a constraint (`snapGridVertical` or `snapGridHorizontal`), or by drawing a corresponding snap pattern shape.

Controlling the Display of Global Grid

You can also enable global grid display using the `snapGridVertical` and `snapGridHorizontal` constraints. The constraint value is the name of the global `snapPatternDef` and the direction the same as is specified in that `snapPatternDef`.

In the example below, the constraint value is `GFG`, which is the name of the global, vertical `snapPatternDef` shown on the previous page.

```
spacings(
    snapGridVertical("GFG")
  );spacings
```

Anchoring the Global Grid

The global grid is anchored in the following order of priority:

- Shape on global snapPatternDef LPP
- Snap boundary
- PR boundary
- Origin axis

© 2020 All Rights Reserved.

You can also enable or disable the global grid anchoring using the following environment variable:

```
graphic snapPatternGlobalGridAnchoring boolean nil
```

When the global grid anchoring is disabled, the global grid is anchored to the origin axis.

A few examples of global grid anchoring are shown below:

- Anchoring by shape on global snapPatternDef LPP: The vertical grid is anchored to the lower edge of the drawn shape. In the example below, an offset of 24nm is shown.

© 2020
All Rights Reserved.

- Anchoring by PR boundary: If no shape is drawn on global grid LPP, then Virtuoso looks for a PR boundary. If it is found, then the global grid is anchored to the lower edge of the PR boundary.

By default, the PR boundary snaps to the least common multiple of the grid values in the X and Y directions. You can use the environment variables, boundarySnapXGrids and boundarySnapYGrids, to override the default behavior and snap the PR boundary to the specified grids in X and Y directions, respectively.

When the creation method for PR boundary is *Rectangle*, both the edges on the X and Y axis are snapped. However, when the creation method for PR boundary is *Polygon*, all points are snapped. When the creation method for PR boundary is *From Selected Set* and the *Rectilinear* option is not selected, the bounding box is snapped.

During creation, only the outer edges of the rectilinear PR boundary snaps to the grids. During stretch, all the selected edges, inner or outer, of the rectilinear PR boundary snap to the grid.

For editing command, the PR boundary snaps with the snap boundary as reference, if `dbGetGlobalGridOffsetReferenceType(cv)` is specified as `boundary`. If the

© 2020 All Rights Reserved.

`drdEditSmartSnapAllowedWidthSnap` environment variable is specified and the placement grid is present in the design, these settings are prioritized for snapping.

Note: The PR boundary snaps to the global grid irrespective of global grid offset reference being set to boundary or origin.

- Anchoring by Origin Axis: If a PR boundary, snap boundary, or a drawn shape on LPP is not found, then the global grid is anchored to the origin axis.

Global snapping is done similar to local snapping. For more information, see Snapping Objects.

Note: When you move an object in a design, if the object overlaps the local snap pattern shape, then the object snaps to the local snap pattern shape. However, if no local snap pattern shape is found, then the object snaps to the global snap pattern grid.

Commands Supporting Snap Pattern Snapping

Snap pattern snapping is supported in Create Rectangle, Create Instance, Create Via, Create Wire, Copy, Move, Stretch, Flip, Rotate, Yank and Paste, and Repeat Copy commands.

© 2020 All Rights Reserved.

You can enable the environment variable, snapResizeWireObjects, to snap, resize, and color all pathsegs and vias in the selection set during the Move and Copy commands. The pathsegs are snapped to the track by maintaining the relative spacing between them. The connectivity of pathsegs and vias is also maintained.

© 2020 All Rights Reserved.

Controlling the Display of Snap Patterns in Palette

You can control the visibility of snap patterns (local and global grids) through the *Grids* panel of the Palette assistant. The grid drawing style is controlled through the *Snap Pattern Display* field in the Display Options Form. For more information on the *Snap Pattern Display* field, see Controlling the Display of Snap Patterns.

Note: Starting with ICADV12.1 ISR10, the layerset file references snap pattern entries by the snap pattern Def name. In ICADV12.1 ISR9 and earlier releases, snap pattern entries were referenced by LPP name, instead of the snap pattern Def name. As a result, older layerset files that contain LPPs corresponding to these grids are not read correctly in ICADV12.1 ISR10 and later releases.

The selections that you make in the *Layers*, *Objects*, and *Grids* panels of the Palette assistant control the visibility of the shape boundary and grid on the canvas. When the grid visibility is on, the shape boundary and grid is displayed irrespective of the LPP and shape visibility settings.

© 2020 All Rights Reserved.

The following table lists the options selected in the Palette assistant and the results displayed on the canvas.

LPP Visibility (Layers Panel)	Shape Visibility (Object Panel)	Grid Visibility (Grids Panel)	Display on Canvas
t	t	t	Shape boundary + Grid
t	nil	t	Shape boundary + Grid
nil	nil	t	Shape boundary + Grid
nil	t	t	Shape boundary + Grid
t	t	nil	Shape boundary
t	nil	nil	Nothing
nil	nil	nil	Nothing
nil	t	nil	Nothing

© 2020 All Rights Reserved.

Modes for Object Placement

In a FinFET based chip, all instances should be aligned to a fin grid, either to several independent local grids or to a global grid.

For the placement of Pcells and small instances, placement is done by searching the instance master for fin grids. If no fin grids are found, snapping shapes are searched. This functionality is referred to as Detail Mode Placement. For more information, see Instance Snapping. Detail Mode Placement is suitable only for small blocks.

For the placement of large blocks, it is not feasible to search the instance master for fin grids or snapping shapes. Instead, it is recommended that you use Abstract Mode Placement. In Abstract Mode Placement, an instance snaps to a global snap pattern grid using its snapBoundary or PRBoundary. If a snapBoundary or PRBoundary is not found, then the instance's origin axis is used as the reference point for placement.

Abstract Mode Placement allows fast placement of large layout blocks. For more information, see Abstract Mode Placement.

There may be a situation when both Detail Mode and Abstract Mode Placements are not appropriate. One such example is when the design has a third-party IP block laid out in a different methodology. For such cases, you can use Manual Mode Placement Override. For more information, see Manual Mode Placement Override.

Selecting Placement Mode

You can switch between Detail and Abstract modes of instance placement using the snapPatternSnappingMode environment variable.

To enable Detail Mode Placement, set the `snapPatternSnappingMode` environment variable as follows:

```
envSetVal("layout" "snapPatternSnappingMode" 'cyclic "detail")
```

To enable Abstract Mode Placement, set the `snapPatternSnappingMode` environment variable as follows:

```
envSetVal("layout" "snapPatternSnappingMode" 'cyclic "abstract")
```

Abstract Mode Placement

When Abstract Mode Placement is enabled and an instance is placed in the design window, Virtuoso does not search for the master for individual shapes. Instead, an instance snaps to

© 2020 All Rights Reserved.

a global snap pattern grid using its snapBoundary or PRBoundary. If no global grid is enabled, then there will be no instance snapping.

The following figure depicts PRBoundary-based snapping to the global grid.

The placement offset is taken as the offset defined in the snap pattern definition of the global snap pattern. If the master contains a snapBoundary, it is used for snapping. If no snapBoundary is found, then the PRBoundary is used for placement. Otherwise, the instance's origin axis is used as the reference point for placement.

© 2020 All Rights Reserved.

The following figure shows the high-level flow diagram of instance snapping:

Manual Mode Placement Override

The Manual Mode Placement override is used in cases where a third-party IP block is used in a design. This mode enables you to manually specify the placement mode or offset on a per instance or master basis. This is done by setting the `leSnapPatternSnapping` property on either the instance or the master. If the `leSnapPatternSnapping` property is set on both the instance and its master, the instance definition overrides the master definition.

The `leSnapPatternSnapping` string property can have the following values:

- off: No automatic snapping
- detail: Force Detail Mode Placement
- abstract: Force Abstract Mode Placement
- 1 number: Force Abstract Mode Placement with manual offset for either one or both directions along which snapping is performed
- 2 numbers: Force Abstract Mode Placement with separate manual offsets

© 2020 All Rights Reserved.

- 1st number is the vertical offset (for example, fin grid)
- 2nd number is the horizontal offset (for example, poly grid)

© 2020 All Rights Reserved.

Create Custom Placement Area

Use this form to create a custom placement area.

Important

When using the form to manually create a custom placement area in Layout XL or higher tiers, the custom placement area is automatically snapped to an appropriate grid depending on the state of the `snapCpaToPlacementGrid` environment variable.

- *Scenario 1: The* `snapCpaToPlacementGrid` *environment variable is set to* `nil`.

  When the `snapCpaToPlacementGrid` environment variable is set to `nil`, which is the default state of the environment variable, the custom placement area is snapped to the X Snap and Y Snap spacing, provided the following conditions are fulfilled:

  a. X Snap and Y Snap spacing are multiples of the manufacturing grid, and

  b. Both, X Snap and Y Snap spacing are equal to or greater than the manufacturing grid.

  If either of the snapping conditions is not met, the custom placement area is snapped to the manufacturing grid.

- *Scenario 2: The* `snapCpaToPlacementGrid` *environment variable is set to* `t`.

  When the `snapCpaToPlacementGrid` environment variable is set to `t`, the custom placement area is snapped to the placement grid. If the placement grid is unavailable, the conditions for snapping to the X/Y spacing are verified. If both the conditions are fulfilled, the placement area is snapped to the X-snap and Y-snap spacing. Else, the placement area is snapped to the manufacturing grid.

Snapping for singleTrackCenter Snap Mode

The sections below describe the snapping behavior for layers with `singleTrackCenter` snapping mode. The snapping modes are specified in the `snappingMode` argument of the `snapPatternDefs` as describe in the snapPatternDefs section.

Important

`singleTrackCenter` snapping is performed only for global grids.

When the `singleTrackCenter` attribute is specified for a layer, a shape is snapped to global grid only if it can be centered on a single track while meeting enclosure rule on both

© 2020 All Rights Reserved.

sides. This means that a shape would snap to the global grid only if its dimension is twice the `trackEnclosure` value. In this case, combined snapping layer BBox is not considered. If the device contains any shape meeting the dimension rule, the shape is snapped even if the combined layer BBox does not meet the enclosure on both sides.

Enhancements to the Create Rectangle Command

In the Create Rectangle command, both the points are snapped to the grid by applying negative enclosure for the lower edge and positive enclosure for the upper edge of the rectangle. While drawing rectangles on LPPs with `singleTrackCenter` attribute, the points are snapped with respect to the same track, this restricting the dimension to twice the enclosure value.

In addition, you can now disable snapping while drawing rectangles using a special mode enabled using the `Shift` key.

Creating a Rectangle of Single Track Height

To create a rectangle of single track height, start the Create Rectangle command without pressing the `Shift` key. The first point of the rectangle is snapped according to the enclosure value. For vertical grids, the height of the rectangle is fixed after the first point is entered and the mouse can only be moved in the X-direction. Similarly, for horizontal grids, the width of the rectangle is fixed after the first point is entered, and the mouse can only be moved in the Y-direction.

A rectangle of single track height

© 2020 All Rights Reserved.

Creating a Rectangle of Arbitrary Height with First Point Snapped

To create a rectangle of arbitrary height with the first point snapped, start the Create Rectangle command. The first point is snapped according to the enclosure value. Press the `Shift` key during mouse drag to disable snapping for the second point. A rectangle of arbitrary height is created.

A rectangle of arbitrary height

Note: If the `allowedWidthRanges` constraint is specified for the LPP, the height of the rectangle is a multiple of the discrete width specified. If this constraint is not specified, a rectangle of arbitrary height is created.

Creating a Rectangle of Arbitrary Height without Snapping

To create a rectangle of arbitrary height without snapping, start the Create Rectangle command. Press the `Shift` key during mouse drag to disable snapping for the first point. The second point of the rectangle will also not be snapped. A rectangle of arbitrary height is created.

A rectangle of arbitrary height

© 2020 All Rights Reserved.

Note: If the `allowedWidthRanges` constraint is specified for the LPP, the height of the rectangle is a multiple of the discrete width specified. If this constraint is not specified, a rectangle of arbitrary height is created.

Snapping for Edit Commands

Shape Snapping

For edit commands, if a shape is on the snapping LPP with the `singleTrackCenter` snapping mode, snapping is performed only if the height or width (depending on the grid direction) of the shape is equal to twice the enclosure value. For other cases, snapping is disabled.

For shapes on a snapping LPP with the `singleTrackCenter` snapping mode, the value specified in the `checkAllSPGridsEnclosures` environment variable is ignored and the shape dimensions are verified in all cases.

Instance Snapping

The `singleTrackCenter` layer purpose pairs (LPPs) in the instance master have a higher precedence than `multiTrackCenter` LPPs. If there is a shape on the `singleTrackCenter` LPP in the master, the shapes on `multiTrackCenter` LPPs are not snapped. The `multiTrackCenter` LPPs are snapped only when there are no shapes on any of the `singleTrackCenter` LPPs. Snapping is not performed in a direction if shapes exist on the `singleTrackCenter` LPP, but no shape meets the dimension rule.

In case of `singleTrackCenter` LPPs, individual shapes inside the master are snapped instead of the combined bounding box which is used for `multiTrackCenter` LPPs. A shape on `singleTrackCenter` LPP is snapped only if dimension rule (twice the grid enclosure) is met. In case of multiple `singleTrackCenter` LPPs, all the `singleTrackCenter` LPPs are examined until a shape meeting the dimension rule is found. If all the shapes on the `singleTrackCenter` LPP violate the dimension rule, snapping is not performed in that direction.

Support for the exceptOverlap Attribute

When `exceptOverlap` attribute is specified for a snapping LPP, the shapes on this LPP do not snap if they completely lie inside the shapes on `exceptOverlap` LPP. In case of partial overlap, snapping does take place. This attribute can be specified for both `singleTrackCenter and multiTrackCenter LPPs.`

© 2020 All Rights Reserved.

For `singleTrackCenter` LPP, if a shape meeting dimension rule is found:

- If it does not completely overlap any shape on the `exceptOverlap` LPP, it is snapped to the grid.
- If it completely overlaps any `exceptOverlap` LPP shape, the shape is ignored for snapping

If all the shapes on `singleTrackCenter` LPP violate the dimension rule or completely overlap `exceptOverlap` LPP, snapping is not performed in this direction.

For `multiTrackCenter` LPP, snapping is performed only if the combined bounding box of shapes does not completely overlap any shape on the `exceptOverlap` LPP.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

16

Width Spacing Pattern Support in Layout Editing

This chapter includes the following topics:

- Overview
  - Controlling the Display of Width Spacing Patterns
- Snapping Behavior of Objects in Width Spacing Patterns
  - Pin Snapping
  - Region Shape Snapping
  - Region Group Snapping
  - Instance Snapping: Abstract Mode Placement
- Enhancements in Editing Commands
- Enhancements in Create Commands
  - Creating Pins
- Display Enhancements for WSP
  - Support for Alpha Blending
  - Support for Filtering Tracks
- Interactive Snapping
- Interactively Setting Color on a Selected Object

© 2020 All Rights Reserved.

Overview

Virtuoso has been enhanced to support snapPatternDefs called widthSpacingSnapPatternDefs. The width spacing pattern (WSP) definitions are stored in the technology database. A WSP is a sequence of width and spacing values that define the track patterns to which the shapes in a design are aligned. WSP is an advanced form of the snapPatternDef that can have multiple width and space combinations.

Note: `widthSpacingSnapPatternDef` are available both in the technology database and the design.

The figure below illustrates that each width spacing pattern is a sequence of width-space pairs. Patterns are specified as width and space pairs. Groups of width and space pairs can be repeated. The first track of the pattern is anchored to the period grid using an offset.

© 2020
All Rights Reserved.

Controlling the Display of Width Spacing Patterns

The layout editor has been enhanced to visualize the width spacing patterns. You can control the display of width spacing patterns through the Display Options form or Palette assistant.

Display Options Form

You can control the display of width spacing patterns using the *Snap Pattern Display* option in the Display Options form. In addition to boundary and tracks, you can now display periods for WSPs.

1. Choose *Options – Display*.

   The Display Options form opens.

2. Select one of the following from the *Snap Pattern Display* list:
   - *Full*: Displays the snap pattern (SP) boundary, tracks, fins, and width spacing patterns periods.
   - *Tracks*: Displays the snap pattern boundary and tracks.
   - *Boundary*: Displays only the snap pattern boundary.
   - *Periods*: Displays only the width spacing pattern periods.

© 2020 All Rights Reserved.

- *Periods and Tracks*: displays the width spacing patterns periods and snap pattern tracks.

Full

Tracks

Boundary

Periods

3. Click *OK*.

Tip

You can use the `leCycleSnapPatternDisplay` SKILL function to cycle through the different snap pattern display types. You set the default value using the new snapPatternDrawingStyle environment variable.

© 2020 All Rights Reserved.

Palette Assistant

You can control the visibility of snap patterns (Local Grids, Global Grids, and Width Spacing Grids) through the *Grids* panel of the Palette assistant.

The layerset file references snap pattern entries by the snap pattern Def name. For example, the local grid named *fbd46* and WSP grid named *M5WSP*, shown in the figure above, are listed as follows in the layerset file:

```
GridLayer Local_Grids;fb46 t NA

GridLayer Width_Spacing_Grids;M5WSP t NA
```

Grids Panel

The visibility settings of WSP grids are cellview specific. Therefore, if you have two different cellviews open and you modify the visibility setting of a WSP grid in one cellview, the visibility setting of this WSP grid is not affected in the other cellview.

© 2020 All Rights Reserved.

In the example shown below, you can see that the visibility settings of WSP grids in the two open cellviews are identical.

© 2020 All Rights Reserved.

If you now turn off the visibility of WSP grid *M4WSP* in cellview `even_odd`, you will find that the visibility setting of WSP grid *M4WSP* in cellview `inst_snapping` remains unchanged.

© 2020
All Rights Reserved.

However, while editing a cellview, if you run the *Edit In Place* command or descend into a different cellview, the visibility settings of WSP grids are refreshed to reflect the settings for the cellview being edited, as shown in the example below.

© 2020 All Rights Reserved.

Snapping Behavior of Objects in Width Spacing Patterns

Pin Snapping

A pin that has the snapping LPP that cannot be snapped to a snap pattern will be snapped to a width spacing pattern. A pin will either snap to a snap pattern or a width spacing pattern, but not both.

If a pin shape can be snapped to a snap pattern, there will be no change in behavior. Otherwise, it will try to snap to WSP. WSP snapping is only supported for rectangular and not polygon pins. The center of the pin is snapped to the center of the closest track.

For snapping a pin to a WSP region shape, the applied pattern is considered; and for a global WSP, the default active pattern is considered.

If the target track width is greater than the pin width, the pin width is updated to match the track width. However, if the track width is less than the pin width, the pin is updated based on the value of the `minWidth` constraint defined in the technology file.

Pin snapped to WSP

Note: A pin inside an instance matches a WSP track if its width and mask color match that of the WSP track.

Pin Snapping for Overlapping Regions

There regions can be multiple overlapping regions and the snapping behavior is described below:

- **Region R1 partially overlaps another region R2**

© 2020 All Rights Reserved.

The region R1 has `WSP1` as the active pattern and the region R2 has `WSP2` as the active pattern. The overlap area of the two regions does not have any pattern available and displays a blank area. Also, a pin cannot be snapped inside the overlapping area.

- **Two regions with the same size overlap each other**

  The two regions R1 and R2 have the same size. The region R1 has `WSP1` as the active pattern and the region R2 has `WSP2` as the active pattern. In this case, no snapping is possible inside the overlapping region and pins cannot be created inside this region.

- **Region R1 is fully enclosed in another region R2**

© 2020 All Rights Reserved.

- Region R1 has `2xWidth` as the active pattern and region R2 has `WSP1` as the active pattern. The tracks in the innermost region are effective for pin snapping. The tracks of R1 are used for snapping pins in the overlapping area and in the remaining area the tracks of R2 are used for snapping.

In WSP detail mode snapping, if your design has pins, they are used for snapping. However, if pins are not available in the design, rectangles, pathSegs, two-point paths, and rectangular polygons are used for snapping.

Note: If the snapRectToWSPGrid environment variable is set, rectangles snap to the WSP grid, similar to pins.

© 2020 All Rights Reserved.

Region Shape Snapping

A region is any shape drawn on a WSP LPP. The region's `widthSpacingSnapPatternDef` may or may not be defined as part of the global WSP grids. You can create a region shape on a non-global WSP grid with a period different than the period of the global grid.

WSP drawn on a non-global WSP grid

The snapping LPP of the region's `widthSpacingSnapPatternDef` is used to find the global WSP grid to which the region shape snaps. If there are multiple global grids to which the snapping layers of region `widthSpacingSnapPatternDef` can snap, the region is snapped to the first matching global grid.

For vertical WSPs, the region shape is snapped along the x-direction. For horizontal WSPs, the region shape is snapped along the y-direction.

Region snapping is supported only for rectangles and polygons. If the WSP LPP is also the snapping LPP of a `snapPatternDef`, the region shape is snapped only to the WSP grid.

Note: You can define the WSP grids in the technology file, or you can draw them in the design.

© 2020 All Rights Reserved.

Region Group Snapping

A region group is an `oaFigGroup` that contains only WSP region shapes or snap pattern shapes. In other words, the LPP of each shape in the `oaFigGroup` has a `snapPatternDef` or `widthSpacingSnapPatternDef`.

The reference region shapes along each direction are determined based on the region with the largest period of the snapping `widthSpacingSnapPatternDef`, and are snapped similar to Region Shape Snapping. If there is no region along a direction in the group, then snapping does not occur along that direction.

Note: If a region group contains both WSPs and SPs, WSP snapping takes precedence over SP.

Instance Snapping: Abstract Mode Placement

Th abstract mode placement allows fast placement of large layout blocks. The WSP default abstract mode placement is performed in the following cases:

- If the `leSnapPatternSnappingVerticalGrid` or `leSnapPatternSnappingHorizontalGrid` properties are not specified.
- If the snapPatternSnappingMode environment variable is not set to abstract.
- If the `leSnapPatternSnapping` property has not been specified.

In the above cases, the steps used to snap to the default WSP grid in either direction are as follows:

- The largest period is identified.
- For the matching period, the definition with the highest mask number snapping layer is identified.
- In case there are multiple definitions on a snapping layer with the largest period, sorting is done based on the definition names.
- If no WSP is found along a given direction, the SP global grid is used for abstract mode placement.

An advanced scenario when abstract mode placement is performed for WSPs is described below:

- If you specify the `leSnapPatternSnappingVerticalGrid` or `leSnapPatternSnappingHorizontalGrid` property.

© 2020 All Rights Reserved.

Note: The offset overrides can be specified for WSP abstract mode snapping using the `leSnapPatternSnapping` property only if one of the properties, `leSnapPatternSnappingVerticalGrid` or `leSnapPatternSnappingHorizontalGrid`, has been set. If you do not specify `leSnapPatternSnappingVerticalGrid` or `leSnapPatternSnappingHorizontalGrid` and only specify the offset overrides using the `leSnapPatternSnapping` property, the largest grid in the constraint in either direction is considered for snapping. It is assumed that you want to specify an offset override only when you override the grid.

In this case, the grid that you have specified is accepted without any further check. If `leSnapPatternSnapping` property has an offset override, it is applied based on the existing rules that apply to the `leSnapPatternSnapping` property. If only one direction grid has been specified, the other grid is determined based on the default algorithm.

Note: When `leSnapGridHorizontal` and `leSnapGridVertical` abstract override properties have been specified, the object closest to pointer is used as the reference object for snapping. However, in case of modgens, the object that has the abstract override properties specified is used as the reference object for snapping.

© 2020 All Rights Reserved.

Enhancements in Editing Commands

The `Move`, `Copy`, and `Stretch` commands have been enhanced to support snapping for WSP LPPs. If the selected set has multiple objects, the object closest to the reference point is snapped and all the objects in the selected set are moved by the same transform to maintain their relative spacing. The snapping object, closest to the pointer, in the selected set is the reference. In case there are multiple shapes closest to the pointer, the shape with the higher mask number takes precedence. If there is no shape closest to the pointer, the precedence that is followed is the closest instance, row, boundary, and then blockage.

As illustrated in the figure below, in case multiple objects are selected, this reference object is highlighted on the canvas.

The object is snapped depending on its type (pin, region, or instance). For more information, see Snapping Behavior of Objects in Width Spacing Patterns. The pointer is snapped to the snapping grid as you move the mouse. It gives a visual indication of how the objects will be placed after the editing command is complete.

For `Move` and `Copy` commands, the selected pathseg will be resized if no other object is selected and the Wire Snapping option, in the Layout Editor Options form, is set to *Snap Pattern*.

© 2020 All Rights Reserved.

For `Stretch` of partially selected objects, the selected edge is snapped to the snapping grid. The edges that are not selected are not modified and these edges could be off-grid.

For `Chop` and `Yank` commands, the WSP-specific information (like definition name) present on the region shape is preserved.

Note: For `Move` and `Copy` commands, reference pathseg and vias are resized interactively according to the track width.

© 2020 All Rights Reserved.

Enhancements in Create Commands

Creating Pins

A pin shape snaps to the snap patterns of the LPP on which the pin is being created or edited.

If the snap pattern is a `snapPatternDef`, the pin shape is snapped based on the width magnet parameters. If the snap pattern is `widthSpacingSnapPatternDef`, the pin shape is snapped to the default active pattern available in the region. When you create a pin shape on a pattern, the height of the pin shape is always equal to the track pattern width. By default, a rectangular pin is created where the height of the pin is equal to the track width. You can enable the square pin creation mode using the environment variable rectangularWSPPinAsSquare. In the square pin creation mode, the width of the pin is fixed by the track width. The length of the pin is also fixed by the track width. A square pin can be created on a WSP track with a single click. This mode is useful when you want to pre-fill net names in the Create Pin form. The square pins can be placed quickly, one pin per click. The figure below illustrates how a pin is snapped to a `snapPatternDef`.

© 2020
All Rights Reserved.

The figure below illustrates how a pin is snapped to a `widthSpacingSnapPatternDef`.

The precendence that is followed for pin shape snapping is described below:

- Local grid (`snapPattern` followed by `widthSpacingSnapPattern`) available bound to shape:
  - If the shape is bound to a WSP, the default active pattern applicable to corresponding WSSPD is applicable for pin shape snapping.
- Global grid (`snapPattern` followed by `widthSpacingSnapPattern`) available for pin LPP.
- If local or global grid is not found, the pin shape is snapped to the X or Ysnap spacing grids.

Note: The `minWidth` constraint defined in the technology file needs to be met for pin snapping. If the pin width is less than the `minWidth` constraint, the next track is searched.

Dynamic Display in Pin Creation and Editing Commands

While running the pin creation and editing commands, the applicable WSP track is automatically displayed in the design so that you can visualize the available tracks and their width. This is important since WSP is so track centric.

Note: Dynamic display is visible while running the pin creation and editing commands, only when the *Snap Pattern Snapping* option is selected in the Layout Editor Options form and the reference object is snappable to the WSP grid.

For the `Create Pin` command, the entry LPP is used in the dynamic display. In case you are editing an existing pin, the LPP of the pin is used in the dynamic display.

Note: When you are creating or editing a pin, the corresponding WSP grid is displayed even if the visibility of the grid is turned off from the *Grids* panel of the Palette assistant.

© 2020 All Rights Reserved.

Display Enhancements for WSP

Support for Alpha Blending

The alpha blending feature lets you control the transparency of tracks, FinFET/WSP shapes, regions, and Wire Editor tracks.

You can control the transparency at different levels for the track fill or outline using the options available in the Transparency Options form or the Alpha Blending Environment Variables.

1. Choose *Options – Transparency*.

   The Transparency Options Form opens.

2. Select the *Transparency Enabled* check box to enable the transparency options in the form.

© 2020 All Rights Reserved.

3. You can specify the outline and fill values for *Tracks*, *FinFET/WSP Shapes*, *Regions*, and *Wire Editor Tracks* using the sliders.

Note: You can use the *Linked* check box to change the *Outline* and *Fill* values for *Tracks*, *FinFET/WSP Shapes*, *Regions*, or *Wire Editor Tracks* by tuning only one slider at a time. If you want to change the *Outline* and *Fill* values of *Tracks*, *FinFET/WSP Shapes*, *Regions*, and *Wire Editor Tracks* together by tuning only a single slider, select the *All Linked* check box.

Alpha Blending Environment Variables

You can also use the following environment variables to control the transparency at different levels for the track fill or outline:

- alphaBlendingBoundaryFillValue
- alphaBlendingBoundaryHaloValue
- alphaBlendingBoundaryOutLineValue
- alphaBlendingFinFillValue
- alphaBlendingFinHaloValue
- alphaBlendingFinOutLineValue
- alphaBlendingTrackFillValue
- alphaBlendingTrackHaloValue
- alphaBlendingTrackOutLineValue
- useAlphaBlending

Support for Filtering Tracks

In Virtuoso, by default, all the tracks on a WSP snapping layer are active. You can activate only a subset of tracks. This is referred to as track filtering. Track filtering is based on the `wireType` for each WSP track. When a track is active, it is displayed and used for layout creation. When a track is inactive, it is not displayed and is, therefore, not used for layout creation.

When a track filter is active, only active tracks are displayed. The track filter specifies a list of `wireType` names. A WSP track with a specified `wireType` is not displayed if its `wireType` is not in the list of filter `wireTypes`. WSP tracks with no specified `wireType` are displayed only if the empty string is included in the list of active wire types.

© 2020 All Rights Reserved.

For more information about track filters, see Filtering Tracks by Wire Type.

Note: When a track filter is active, pin editing is supported only for active tracks. Also, during pin creation, the pointer snaps to active tracks.

Interactive Snapping

Layout editing has been enhanced to provide interactive snapping. This enables you to visualize how the object will be placed after snapping. During interactive snapping, the object which is snappable and is closest to the reference point is used as the reference object. The snapping transform is calculated for the reference object at every mouse movement and the pointer is adjusted to ensure that the reference object is snapped to the grid.

Interactive Snap Pattern Snapping: Create Rectangle

With the interactive snap pattern snapping enhancement for the Create Rectangle command, the first and the second points of the rectangle are snapped. The first point snaps to the upper enclosure and the second point snaps to the lower enclosure.

If discrete dimension is enabled, snapping does not update the second point. The dimension of the created rectangle is based on the value of the `allowedWidthRanges` constraint is specified in the technology file.

© 2020 All Rights Reserved.

If discrete dimension is disabled, the second point discretely snaps by the step size of the snapping grid. The figure below depicts the snapping of the rectangle with the discrete dimension option enabled and disabled.

Interactive Snap Pattern Snapping: Move and Copy Commands

When you move objects, snapping ensures that the bottom edge (vertical snapping) and left edge (horizontal snapping) of the reference object is snapped to the grid. The reference object moves by the step size of the snapping grid. The reference object snaps if the bottom edge of the shape is inside a snapping region even if the reference point goes outside it.

If the reference object overlaps multiple snapping regions, it will snap to the region under the bottom edge of the shape.

If the bottom edge of the reference object goes out of the snapping region then snapping is not performed even if some part of the shape overlaps the snapping region.

© 2020 All Rights Reserved.

The figure below depicts interactive snapping when moving objects.

Note: Interactive snapping for the Copy command works similar to the way it works for the Move command.

Interactive Snap Pattern Snapping: Stretch Command

When you stretch objects, snapping ensures that the bottom edge (vertical snapping) and left edge (horizontal snapping) of the reference object is snapped to the grid. The reference object stretches by the step size of the snapping grid.

If discrete dimension is enabled and it adjusts the selected edge, snapping does not modify the edge. So, if the shape is off-grid before the Stretch command, it will remain off grid after the Stretch command also. If the discrete dimension option is disabled, snapping ensures that

© 2020 All Rights Reserved.

the edges of the shape are on grid after the Stretch command. This is depicted in the figure below.

If you stretch a partially selected edge, the selected edge is snapped to the grid. In this case, the bottom edge snaps to the lower enclosure and the top edge snaps to the upper enclosure of the track.

When you stretch the top edge of the shape, it snaps to the grid till it is inside the snapping region. Snapping is not performed when the edge is out of the snapping region.

In additon to the above commands, the instance is snapped interactively during Create Instance command with every mouse drag.

© 2020 All Rights Reserved.

Interactively Setting Color on a Selected Object

Enhancements in the Property Editor

You can interactively set the color for the selected MPT shape objects and vias through the Property Editor and assistant. This eliminates the need of only using SKILL functions to change the color information for MPT shape objects and vias.

Setting Color on Shape Objects

For a selected MPT shape, you can modify the color and color state using the Property Editor and assistant.

Use the *Color* and *State* fields in the MPT Coloring section on the *Attribute* tab of the Edit Shape Properties form to modify the color information.

Property Editor determines the available colors through the #Masks (number of masks) defined in the technology database. The table below summarizes the display colors based on the association between the #Masks found in the technology database and MPT Coloring section.

#Masks	MPT Coloring Section	Available Colors
NoColor	Gray Out | Hidden	N/A

© 2020
All Rights Reserved.

#Masks	MPT Coloring Section	Available Colors
2	Enabled	grayColor | mask1Color | mask2Color
3	Enabled	grayColor | mask1Color | mask2Color | mask3Color

The values of the *State* field for the available colors are summarized in the table below:

Color	Color State
grayColor	Gray Out | Hidden
mask1Color	unlock | lock
mask2Color	unlock | lock
mask3Color	unlock | lock

The default value of the *Color* field is defined by the `setDefaultColorOnMPTLayers` environment variable. The table below summarizes the default values of the *Color* and *State* fields in the Property Editor, based on the value of the `setDefaultColorOnMPTLayers` environment variable.

setDefaultColorOnMPTLayers	Default Value: Color	Default Value: State
grayColor	grayColor	unlock
mask1Color	mask1Color	unlock
mask2Color	mask2Color	unlock
mask3Color	mask3Color	unlock

Setting Color on Vias

For a selected via, you can modify the color using the MPT Coloring section of the Edit Via Properties form or the Property Editor assistant. You can set the colors for Layer1, CutLayer, and Layer2.

The table below summarizes the MPT coloring fields that are available for the layers.

Via Layer	Available Colors
Layer1	grayColor | mask1Color | mask2Color | mask3Color

© 2020 All Rights Reserved.

Via Layer	Available Colors
CutLayer	grayColor | mask1Color | mask2Color | mask3Color
Layer2	grayColor | mask1Color | mask2Color | mask3Color

Similar to shapes, you can set the lock state for vias. When multiple vias are selected, which implies the Common check box is selected in the Edit Via Properties form, and the lock state is different for some of the selected vias, the AS IS option is displayed to reflect the lock state of the vias.

Video

You can view the enhancements related to MPT Coloring in the videos, *Coloring Utilities in Property Editor* and *Coloring Utilities Dynamic Selection Assistant*.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

A

Layout Editor Forms

- Add Shape to Net Form
- Align Form
- Area and Density Calculator Form
- Assign Net Form
- Background Cellview Form
- Choose Object for Label Creation Form
- Chop Form
- Clean Overlapping Contacts Form
- Copy Form
- Create Area Boundary Form
- Create Blockage Form
- Create Circle Form
- Create Cluster Boundary Form
- Create Custom Placement Area Form
- Create and Edit Clusters Form
- Create Encore Report Form
- Create Geometric Wire Form
- Create Group Form
- Create Guard Ring Form
  - Fluid Guard Ring Context-Sensitive Menu
- Create Instance Form

© 2020 All Rights Reserved.

- Create Label Form
- Create Measurement Form
  - Ruler Context-Sensitive Menu
- Create Multipart Path Form
- Create Path Form
- Create Pin Form
- Create Pins From Labels Form
- Create Polygon Form
- Create P&R Boundary Form
- Create Rectangle Form
- Create Row Form
- Create Slot Form
- Create Snap Boundary Form
- Create Via Form
- Create Via Options Form
- Custom Show Purpose Form
- Define Net/Pin Connections Form
- Delete Form
- Delete Form
- Delete All Markers Form
- Delete Configuration Form
- Delete Marker Form
- Descend Form
- Descend Options Form
- Discard Edits Form
- Display Options Form
- Dynamic Display Form

© 2020 All Rights Reserved.

- Dynamic Selection Assistant Options Form
  - Dynamic Selection Context-Sensitive Menu
- Edit Properties Form
  - Add Property Form
  - Modify Property Form
  - ROD Property Form
  - Edit Cellview Properties Form
  - Edit Cut Pattern Form
  - Edit Via Array Pattern Form
  - Edit Instance Properties Form for a Fluid Guard Ring
- Edit Slot Form
- Edit Synchronous Clone Form
- Enter Points Form
- Export Label Form
- Export Image Form
- Filter Layers Form
- Find Marker Form
- Find/Replace Form
- Flatten Form
- Flip Form
- Group Return
- Guard Ring Form
- Guard Ring Chop Options Form
- Guard Ring Create Tunnel Options Form
- Heal Options Form
- Halo Editor Form
- Highlight Options Form

© 2020 All Rights Reserved.

- Install Guard Ring Form
- Instance to Mosaic Form
- Layer Generation Form
- Layout Editor Options Form
- Load Multipart Path Template Form
- Magnifier Options Form
- Make Cell Form
- Make Readonly/Editable Form
- MarkNet Options Form
- Marker Text Window
- Modify Corner Form
- Move Form
- Pad Opening Info Form
- Palette Options Form
- Paste Form
- Print Hierarchy Tree Form
- Quick Align Form
  - Quick Align Context-Sensitive Menu
- Refresh Cellviews Form
- Remaster Instances Form
- Remove Shape From Net Form
- Repeat Copy Form
- Reshape Form
- Restore View Form
- Return To Level Form
- Rotate Form
- Save a Copy Form

© 2020 All Rights Reserved.

- Save All Mark Nets Form
- Save/Restore Selection Set
- Save Configuration Form
- Save Via Variant Form
- Save View Form
- Save Multipart Path Templates Form
- Select Area Shape Type Form
- Select Array Member Form
- Selection Options Form
- Selection Protection Highlight Options Form
- Set Area View Level Form
- Set Default Application Form
- Set Pin Label Text Style Form
- Show Selected Set Form
- Size Form
- Split Form
- Stretch Form
- Summary Window
- Tap Form
- Track Pattern Editor Form
- Transmission Line Bend Form
- Transmission Line Form
- Transmission Line Taper Form
- Transparency Options Form
- Tree Form
- Uniquify Form
- Unmark Net Form

© 2020 All Rights Reserved.

- Yank Form

© 2020 All Rights Reserved.

Add Shape to Net Form

Net Name lets you enter a net name when the *Auto* button is off.

Auto determines whether you enter a net name or get the net name automatically:

- *On* selects the net based on the pins overlapping the selected shapes. The overlapping pin and shape must have the same layer-purpose pair. Each selected shape is given the net of a pin that overlaps it, if the shape does not already have a net.
- *Off* lets you specify the net name. The selected shape does not have to be overlapping the specified pin.

Related Topics

Adding Shapes to a Net

leHiAddShapeToNet

© 2020 All Rights Reserved.

Align Form

The form comprises the following sections:

- Reference
- Align Using
- Spacing and Sorting

Reference

This section allows you to choose a reference point or edge of an object, shape or layer to which you want to align other objects.
Environment variable: alignReference

Point aligns objects, shapes, or layers to a point you select. Select the point by using the pointer or by typing the numerical X and Y values of the point in the CIW.

Object aligns objects, shapes, or layers to an object. Select the type of object from the cyclic field, then use the pointer to select the object. Layout editor supports instances, rows, cluster boundaries, groups, and pins. The *Align* command does not support area halos, layer halos, P&R boundaries, snap boundaries, area boundaries, and blockages. When you select the *Object* option, the *Use* field is activated.
Environment variable: alignRefObjType

Layer aligns objects or shapes to a layer edge of a reference object, shape or layer. Select the layer from the cyclic field, then use the pointer to select the reference object. The cyclic field displays layers defined in the technology file. When you select the *Layer* option, the *Use* field is activated. You can search for specific layer-purpose pair using the search filter in the *Layer* list box.

Note: If the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form, only the layers selected using the search filter in the *Layers* panel of the Palette are visible in the *Layer* list box.
Environment variable: alignLayer

Align Direction sets the alignment direction.
Environment variable: alignDirection

- **Horizontal** aligns objects in the X direction.
- **Vertical** aligns objects in the Y direction.

Use cyclic field sets the edges that are aligned to the objects, shapes or layers.

- If you select a *Point* to use as reference and select the *Horizontal* or *Vertical Align Direction*, you can align the point to an object or layer by using the options in the

© 2020 All Rights Reserved.

Align Using section of the form. The *Use* cyclic field in the *Reference* section is disabled when you select *Point*.

- If you select either the *Object* or *Layer* option to use for reference and *Horizontal* as the *Align Direction*, the valid values of *Use* are *Top Edge*, *Bottom Edge*, and *Center*. If you select the *Vertical Align Direction*, valid values are *Left Edge*, *Right Edge*, and *Center*. If the *Object* is an *Instance*, the *Use* valid value also includes *Origin* for both align directions. You can align the object to an object or layer by using the options in the Align Using section of the form. You can either select the objects and then set the alignment criteria (ion mode) or set the alignment criteria and then select the objects (post-selection mode).
  Environment variable: alignRefMethod

Set New Reference clears any existing reference point and prompts you to specify a new reference point to which the selected objects are to be aligned. Click the *Set New Reference* button, then click in the cellview window at the point where you want the objects to begin alignment.

Align Using

This section allows you to choose which point or edge of the objects will align to the reference point, object or layer selected through the Reference section of the form.
Environment variable: alignUsing

Object sets the object type of objects that can align to the point, object or layer selected in the Reference section. The valid objects include *Instance*, *Any Shape*, *Row*, *Cluster Boundary*, *Group*, and *Pins*. The default value is *Instance*. When *Pins* are selected in the *Reference* section, this object list is restricted to *Pins* only.
Environment variable: alignObjType

Layer sets the layer of the objects to align to the point, object or layer selected in the Reference section. The layers defined in the technology file display in the cyclic field. You can search for specific layer-purpose pair using the search filter in the *Layer* list box.

Note: If the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form, only the layers selected using the search filter in the *Layers* panel of the Palette are visible in the *Layer* list box.
Environment variable: alignLayer

Use cyclic field sets the object or layer edge to align with the point, object or layer edge selected in the Reference section of the form.
Environment variable: alignMethod

- If you select *Point* in the *Reference* section and *Object* in the *Align Using* section, the *Use* cyclic field choices are *Upper Left*, *Upper Center*, *Upper Right*, *Center*

© 2020 All Rights Reserved.

Left, *Center Center*, *Center Right*, *Lower Left*, *Lower Center*, *Lower Right* and *Origin*.

- If you select *Object* or *Layer* in the Reference section, *Horizontal* as the *Align Direction,* and *Object* in the *Align Using* section, the *Use* cyclic field choices are *Top Edge*, *Bottom Edge*, and *Center*. If you select *Vertical* as the *Align Direction*, the *Use* cyclic field choices are *Left Edge*, *Right Edge*, and *Center*. If the *Object* is set to *Instance* in the *Align Using* section, the *Use* valid value also includes *Origin*.

Spacing and Sorting

This section allows you to control the spacing and sorting functionality.

Apply Spacing check box aligns objects with respect to the reference object as per the spacing value specified in the text field, in the *Align Direction*. If you do not select this option, the *Align* command aligns objects and leaves the original spacing between objects intact.
Environment variable: alignApplySpacing

Apply Spacing field specifies the spacing value between the aligned objects in the *Align Direction*. The value must be a floating point number. The default value is `1.0`.
Environment variable: alignSpacingValue

Orthogonal Spacing check box aligns objects with respect to the reference object as per the spacing value specified in the text field, in the direction perpendicular to *Align Direction*. This option is available only when the *Apply Spacing* option is selected.

Orthogonal Spacing field specifies the spacing value between the aligning edges of objects, in the direction perpendicular to the *Align Direction*. The value must be a floating point number. The default value is `1.0`.

If the *Align Direction* is *Horizontal*, then the *Apply Spacing* field value is used to space objects along X-axis while *Orthogonal Spacing* field value is used to space objects along Y-axis. Similarly, if the *Align Direction* is *Vertical*, the *Apply Spacing* field value will space objects along Y-axis while *Orthogonal Spacing* field value will be used to space objects along X-axis. You can deselect the *Orthogonal Spacing* option if you want to space objects only in the *Align Direction.*

Between Objects specifies the distance (in user units) to maintain between adjacent edges of objects to be aligned. The first object overlaps the selected reference point. The next object aligns at a distance specified in the *Apply Spacing* field from the adjacent edge of the previous object. If you specify a negative number, the aligned objects overlap.

© 2020 All Rights Reserved.

In the following example, point R is the selected reference point.

The selected *Align Direction* is *Horizontal*, the *Use* cyclic field in the Align Using section is set to *Lower Left*, and the *Apply Spacing* value is specified as `2.0`. When you apply the *Align* command, the lower left point of the first rectangle, A, overlaps the reference point R. For aligning the next rectangle, B, the *Between Object* spacing distance of `2.0` is calculated from the outer edge of rectangle A to the lower left point of rectangle B, in the horizontal direction.

The remaining rectangles are similarly aligned in the horizontal direction with the *Between Objects* spacing of `2.0`.

If the *Reference* section is set to *Object* or *Layer* and the *Align Spacing* check box is selected, both *Between Objects* and *Pitches* options are disabled. However, the spacing of the aligned objects by default follows the *Between Objects* spacing behavior. Environment variable: alignSpacingType

© 2020 All Rights Reserved.

Pitches specifies the distance (in user units) to maintain across objects between the specified points (*Use* in the Align Using section). The first object overlaps the selected reference point. The next object aligns at a distance specified in the *Apply Spacing* field from the aligned point of the previous object. The *Pitches* option is available only if you select *Point* in the *Reference* section. The spacing distance is specified in the *Apply Spacing* field.

In the following example, point R is the selected reference point.

The selected *Align Direction* is *Horizontal*, the *Use* cyclic field in the Align Using section is set to *Center Center*, and the *Pitches Apply Spacing* value is specified as `3.0`. When you apply the *Align* command, the center point of the first rectangle, A, overlaps the reference point R.

The center point of the next rectangle, B, then aligns to the center point of rectangle A at a distance of `3.0`, in the horizontal direction.

© 2020 All Rights Reserved.

Similarly, center points of subsequent objects align to the center point of the previous object at a distance of `3.0`.

Environment variable: alignSpacingType

Sort Objects By allows you to control how the selected objects are sorted in relationship to the reference object. This option is valid only with preselected objects. Environment variable: alignSortMethod

Align Direction Order sorts the objects by the horizontal or vertical alignment direction selected in the Reference section of the form.

Horizontal Order sorts the objects aligned horizontally, retaining their original horizontal order.

Vertical Order sorts the objects aligned vertically, retaining their original vertical order.

Order Selected sorts the objects in the order they were selected. However, objects are placed above or below, or to the right or left of the reference point according to their original position in the cellview, irrespective of the order in which they are selected. To have all objects above the reference point, they must originate above the reference point.

Instance Name sorts the objects alphanumerically by instance name.

Net Name sorts the objects alphanumerically by net name.

Reverse Sort allows you to reverse the sorting of objects specified in *Sort Objects By*. Environment variable: alignReverseSort

Chain Mode enables you to align device chains to another point, object, or layer. Environment variable: chainMode

All aligns all the instances in the chain.

Selected aligns only the selected instances.

Selected Plus Left aligns the selected instance and all the instances to the left of the selected instance in the chain.

Selected Plus Right aligns the selected instance and all the instances to the right of the selected instance in the chain.

© 2020 All Rights Reserved.

Match layer/purpose with the reference pin lets you change the target pin layer purpose of the pin to the reference pin layer purpose.
Environment variable: alignMethod

Match size with the reference pin lets you want to change the target pin size to that of the reference pin.
Environment variable: alignMatchRefSize

Update pin label layer lets you update the layer of labels to that of the pin. This option can be used when the layers of the pin and label are different.
Environment variable: alignUpdateLabelLayer

Related Topics

Aligning Objects

leHiAlign

© 2020 All Rights Reserved.

Area and Density Calculator Form

Object lets you specify the objects for which you want to calculate the area and density. You can select *All* (all objects) or *Selected* (only selected objects will be considered for area calculation). If an instance, group, via, or guard ring is selected, the area of the shapes inside the selected instance is calculated.

Layer lets you select the layer for which you want to calculate the area and density. You can select the layer pursose pair from the drop-down list or you can specify a layer purpose pair. You can select a single layer or multiple layers for area calculation. The syntax that you should use to specify the layer is:

Single Layer:

```
Met1/drawing
```

Multiple Layers:

```
Met1/drawing1 Met1/drawing2 Met2/drawing1
```

All Purpose:

```
Met1
```

Use () to merge different layers:

```
(Met1 M1) (Met1/drawing1 Met1/drawing2)
```

Hierarchy Range sets how many levels of the hierarchy are to be considered for area and density calculation.

current cellView the current cellview is considered.

current to bottom the current level of hierarchy to the level specified in the *Display Level To* field in the Display Options Form (stop level) is considered. This option is selected by default in the drop down list.

current to stop level the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options Form (stop level) is considered,

current to user level the current level in the hierarchy to level specified by the you.

Region

Entire CellView considers the entire cellview for area and desity calculation. This option is selected by default.

Viewing Area considers the portion of the cellview that is displayed in the canvas at that instant for area and density calculation.

© 2020 All Rights Reserved.

A Selected Boundary considers a selected boundary for area and density calculation.

Draw a Rectangle Region on the Canvas or **Draw a Polygon Region on the Canvas** enables you to specify a rectangular or polygonal area for area and density calculation. Select the required option, and then click in the canvas to specify the vertices of the rectangle or polygon.

Related Topics

Using Area and Density Calculator

leComputeAreaDensity

© 2020 All Rights Reserved.

Assign Net Form

Net Name lets you select a net that you want to assign to the shapes.

Pop up selection window brings up a form in case of overlapping shapes. The form lists all the shapes with details of the assigned net name. You can use this form for both picking a net and assigning a net.

Propagate connectivity option propagates connectivity to physically connected shapes. This option follows Extractor settings specified for constraint group and *Extract connectivity to level*.
Environment variable: propagateAssignNet

Related Topics

Assigning and Unassigning Nets to Shapes

© 2020 All Rights Reserved.

Background Cellview Form

Library, **Cell**, and **View** specify the library, cell, and view names of the cell you want to open as the background cellview. You cannot open the top display cellview as the background cellview.

X and **Y** specify the coordinates to place the background cellview in the window.

Rotation specifies the angle at which the background cellview is placed in relation to the top display cellview.

Display specifies the luminosity at which the background cellview is displayed.

> **off** does not display the background cellview.
>
> **on** displays the background cellview.
>
> **dimmed** displays the background cellview at the luminosity determined by *Dim Intensity*.

Related Topics

Displaying a Background Cellview

© 2020 All Rights Reserved.

Choose Object for Label Creation Form

In the manual and auto label creation modes, if there are multiple overlapping shapes, this form prompts you to choose a layer when clicking on a coordinate with objects on overlapping layers. You can select one or more objects for label creation.

Label displays the label for the layer.

Note: This field is available only if the label creation mode is auto.

Layer displays the layer for the label.

Purpose displays the purpose for the label.

Object displays the object.

© 2020 All Rights Reserved.

Chop Form

Chop Shape controls the shape you want to use as the cutter. You can choose from *rectangle*, *polygon*, *line,* or *wire*. If *Chop Shape* is set to *line*, *Remove Chop* has no effect.

When you select the chop shape as rectangle, you select the *Specify Size* option to specify the size of the chop shape. You specify the width, height, and justification of the chop shape using the *Width*, *Height*, and *Justification* fields.

When you select the chop shape as circle, you select the *Specify Size* option to specify the size of the chop shape. You specify the radius and justification of the chop shape using the *Radius* and *Justification* fields.

When you select the chop shape as line, you select the *Specify Spacing* option to specify the size of the chop shape. You specify the type, width, and justification of the chop shape using the *Type*, *Width, a*nd *Justification* fields. You can select *User Spacing* or *Min Spacing* from the *Type* drop-down list. *User Spacing* lets you specify width of spacing and *Min Spacing* calculates the spacing based on the minimum spacing specified in the technology file. Also, for path, pathSeg, and rectangle, when the *Min Spacing* option is selected, the begin point of the line is on the lower orthogonal grid and the end point of the line is on the upper orthogonal grid.

Note: For all chop shapes, a pathSeg is not converted to a polygon.

Environment variable: chopShape, chopCircleJustify, chopCircleRadius, chopLineJustify, chopLineType, chopLineUserWidth, chopRectHeight, chopRectJustify, chopRectWidth, chopSpecifySize, chopSpecifySpacing

Snap Mode controls the shape of polygon or line segments. *Snap Mode* applies only when *Chop Shape* is set to *polygon* or *line*. You can choose from *anyAngle*, *diagonal*, *orthogonal*, *L90XFirst*, and *L90YFirst*. For more information about these choices, see Setting Up Snap Mode.

Note: When snap mode is *anyAngle* or *diagonal*, paths and pathSegs are chopped similar to the snap mode *orthogonal*.

Remove chop removes the part of the object enclosed by *Chop Shape*.

Chop array enables you to chop an array of instances (mosaic). This check box is selected by default. When this check box is selected and you define the chop area for an array, none of the points of the chop area (vertices of the rectangle and polygon and end points of a line)

© 2020 All Rights Reserved.

should be contained within the array. The vertices or end points should be either outside the array or on the boundary of the array. This is explained in the figures below.

© 2020
All Rights Reserved.

In the following chop areas, not all the points are outside the array. Therefore, these arrays of instances will remain unchanged even if the *Chop Array* check box is not selected.

Environment variable: chopArray

Snap

Pins and boundary to grid snaps the chopped object to a grid depending on the block type. If it is an *analog* block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a *digital* block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not snapped, even if they were touching the boundary before the chop.

Pins to boundary automatically snaps pins to the place and route boundary.

Soft Block Pins to automatically snaps soft block pins either to the *Modified Edge* or to the *Closest Edge* after the chop is complete.

© 2020 All Rights Reserved.

Related Topics

Chopping Objects

Editing Soft Blocks

Chopping an Object in Level-1 Editing Mode

leHiChop

© 2020 All Rights Reserved.

Clean Overlapping Contacts Form

For information about the Clean Overlapping Contacts form, see Clean Overlapping Contacts Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Copy Form

Keep Copying enables you to create multiple copies of an object. With this check box selected, a copy of the selected object is created every time you click in the design display area. You can customize this mode of copying by configuring the other settings in the form. The *Delta*, *Spacing*, and *Exact Overlap* check boxes are not available when you select this check box.

Snap Mode controls the direction in which you can move the copied object.

Change To Layer enables you to copy an object to another layer. You can search for the specific layer-purpose pair using the search filter in the *Change To Layer* list box.
Environment variable: copyToLayer

Note: If the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form, only the layers selected using the search filter in the *Layers* panel of the Palette are visible in the *Change To Layer* list box.

Note: The value selected in the *Change To Layer* field is reset after each copy operation.

Copy color specifies that the coloring information from source objects will be copied from source objects as-is to the corresponding destination objects when copying.

Synchronous copy creates a synchronized copy of the selected layout objects. This implies any edits made to the source object are also replicated in the copied object, and vice versa.

Note: The *Synchronous copy* field is reset after each copy operation.

For more information, see Generating Synchronous Copy.

Display draglines Controls whether draglines are displayed during the Copy command. By default, the draglines indicate connections from the pins of the object you are moving to the pins of the nearest objects. For the draglines to be displayed, both *Copy connectivity* and *Display Draglines* should be selected.

Copies

Type specifies the type of copy, *Line* or *Array*.

> **Line** lets you create a line of copied objects. Use the *Number* field to specify the number of copies to create.
>
> **Array** lets you create an array of copied objects. Use the *Rows* and *Columns* field to specify the dimensions of the array. The array of copied object can be placed only in the original cellview; it cannot be copied to another cellview.

© 2020 All Rights Reserved.

Number lets you specify the number of copies to be made.

Spacing

Mode lets you select the spacing mode.

> **Pitch** lets you specity the detla that applies between the objects in an array the distance specified in the *Delta X, Y* field. The distance is measured between the same-edge corners of the objects in the array.
>
> **Spacing** applies between the objects in an array the distance specified in the *Spacing X, Y* fields.
>
> **Absolute** enables you to copy an object at the location specified in the *X*, *Y* fields.

Apply copies the selected object based on the settings specified in the form.

Connections

Copy connectivity copies the connectivity information on path, pathseg, fluid guard ring, and via to the new object.
Environment variable: copyConnectivity

Note: When the *Display Draglines* option is also selected, draglines are displayed during the Copy command to indicate the connections from the pins of the object you are moving to the pins of the nearest objects.

Increment net name increments the bus bit on the copied object by one if the selected object has a net name with a bus bit. This option is available only when the *Copy connectivity* option is selected.
Environment variable: copyIncrementNetName

Chain Mode enables you to copy device chains.

> **All** copies all instances in a chain.
>
> **Selected** copies only the selected instances.
>
> **Selected Plus Left** copies the selected instance and all the instances to the left of it in the chain.
>
> **Selected Plus Right** copies the selected instance and all the instances to the right of it in the chain.

Snap

> **Pins and boundary to grid** snaps the pin edges to the manufacturing or the routing grid, depending on the selected block type:

© 2020 All Rights Reserved.

If the block type is custom, the pin edges are snapped to the manufacturing grid.

If the block type is digital, the pin edges are snapped to the routing grid.

Rotate turns the object counterclockwise by 90 degrees each time you click the button.

Sideways mirrors the object along the X-axis.

Upside Down mirrors the object along the Y-axis.

Related Topics

Copying Objects

Repeat Copy Form

Generating Synchronous Copy

leHiCopy

© 2020 All Rights Reserved.

Create Area Boundary Form

Name assigns a name unique within the cellview to the area boundary.

Create Method chooses which method is used to create the area boundary object.

> **From Selected Set** creates the area boundary using the rectangular bounding box of the selected set of objects.
>
> **Rectangle** creates the area boundary using two points.
>
> **Polygon** creates the area boundary using multiple points.

Apply creates the P&R boundary using the selected objects.

Snap Mode selects snap mode to use as you enter points when *Polygon* create method is selected.

Create as ROD Object creates the area boundary as a ROD object.

> **Name** assigns a unique name to the new area boundary in the current cellview if the *Create as ROD Object* check box is selected. This unique name consists of the prefix `boundary` followed by an integer. For example, the first area boundary is named `boundary0`, the second one `boundary1`, and so on. You can edit the default name, if required, but ensure that the new name is unique in the cellview.

Soft Fence creates the soft fence. Objects enclosed by the area boundary with a soft fence are compacted relative to each other as well as compacted as an entity relative to objects outside the boundary. Selecting this option adds the property `ctuBoundaryType` to the area boundary, with the property value `prSoftFence`.

Related Topics

Creating Area Boundary

leHiCreateAreaBoundary

© 2020 All Rights Reserved.

Create Blockage Form

Blockage Type selects the type of blockage to create. The choices are: placement, routing, pin, feedthru, or screen.
Environment variable: blockageType

Material selects the material on which the blockage is enforced. The choices are: all, other, nwell, pwell, nplus, pplus, ndiff, pdiff, poly, metal, diffusion, li, cu, or passivationCut. Not all materials are legal for all types of blockages. When Blockage Type is placement, the Material field is grayed out. Other denotes undefined materials.
Environment variable: blockageMaterial

Layer selects the layer associated with the selected material. The layers displayed are those defined in the technology file as having one of the above materials. The default material is `"All"`. The layers listed in this field with `"All"` material include those layers that belong to an LPP with the purpose name `"blockage"`. When Blockage Type is placement, the *Layer* field is grayed out.
Environment variable: `blockageLayer`

Color lets you select the color in which the blockage is created. The default value is *grayColor*. The options available in the *Color* field are based on the layer selected in the *Layer* field. If you select a layer in the *Layer* field that support *mask1Color* and *mask2Color*, the *Color* field will have the options, *grayColor*, *mask1Color*, *mask2Color*, *blackColor*, and *multiColor*. If you select multiple layers in the *Layer* field, the options in the *Color* field are a combination of all the colors supported by the selected layers.

Compute spacing using lets you select method used to compute spacing. You can select *Blockage geometry*, *Effective width*, or *Specify* a spacing value. The *Blockage geometry* option computes spacing using the blockage geometry specified. The *Effective width* option is a blockage attribute that you can assign to any blockage type, such as routing, pin, feedthru, defined by a layer or layers. You can also specify a spacing using the *Specify* option.

Note: When *Blockage Type* is *placement*, the *Compute spacing using* field is grayed out.

Environment variable: blockageSpacingType, blockageSpacingValue

Max Density sets the maximum percentage of the total design area that the metal in the specified layer may occupy. Max Density applies to blockage types placement and screen. Max Density does not apply to blockage types routing, pin, and feedthru.
Environment variable: blockageMaxDensity

Allow PGNet allows shapes with signal types power and ground to be placed within the blockage area.

© 2020 All Rights Reserved.

Environment variable: blockageAllowPGNet

Attach to Owner assigns one instance or one cluster as the blockage's owner, depending upon the type of blockage.
Environment variable: blockageAttachOwner

Keep Owner retains the owner information. When you create multiple blockages, the owner is automatically assigned to the additional blockages. Keep Owner remains grayed out until you select the cluster radio button in the Attach to Owner field.
Environment variable: blockageKeepOwner

Pushed Down indicates if the blockage has been pushed down from a higher-level block.

Environment variable: blockageIsPushedDown

Create as ROD Object creates the blockage as a ROD object.

> **Name** assigns a unique name to the new blockage in the current cellview if the *Create as ROD Object* check box is selected. This unique name consists of the prefix `blockage` followed by an integer. For example, the first blockage is named `blockage0`, the second `blockage1`, and so on. You can edit the default name, if required, but ensure that the new name is unique in the cellview.

Create Method

Environment variable: blockageCreateMethod

> **Manual** requires that you enter points to define the blockage.
>
> > **Snap Mode** selects mode to use when you enter points. Valid snap modes are anyAngle, diagonal, orthogonal, L90XFirst, L90YFirst.
>
> **Automatic** derives points from objects you select, then creates the blockage from these points.
>
> > **Derive From** sets the type of object or objects from which you can derive the points to create a blockage.Select one of the following choices from the cyclic Field Selected Set bBox, Instance Halo, Instance prBoundary, or Instance bBox, then select the corresponding object or objects. The software derives the points from the objects you select.
> > Environment variable: blockageDerivePoints
> >
> > **Apply** creates the blockage using the derived points.

© 2020 All Rights Reserved.

Related Topics

Creating Blockages

leHiCreateBlockage

© 2020 All Rights Reserved.

Create Circle Form

Shape Type enables you to select the shape of the circle that you want to create (full circle, half circle, or quarter circle).
Environment variable: circleShapeMode

Conic Sides sets the number of segments to use for creating the circle. The value specified in the Conic Sides field of the Layout Editor Options Form is displayed in this field.

Specify radius enables the *Radius* field.

Radius sets the radius of the circle.
Environment variable: circleRadius

Related Topics

Creating Circles

leHiCreateCircle

© 2020 All Rights Reserved.

Create Cluster Boundary Form

Cluster Name lists the clusters in the design.

Cluster Boundary Name assigns by default a unique name of `CB_integer`, where *`integer`* is incremented each time a cluster boundary is created within the cellview.

Points lists the points of the cluster boundary.

Overwrite Cluster Boundary lets you overwrite the existing cluster boundary. By default, this field is selected. If this field is not selected, the other fields on the form are not enables and you cannot edit the existing cluster boundary. If there is no cluster boundary, this field is not enabled.

Fit Member Instances Into Cluster Boundary fits the member instances into the cluster boundary.

Cross-Highlight Boundaries highlights in the layout window the cluster boundary associated with the cluster selected in the form.

Create Method

Manual creates a boundary when you enter points in the cellview window.
Environment variable: createBoundaryMethod

Rectangle creates the cluster boundary using two points.

Polygon creates the cluster boundary using multiple points.

Snap Mode selects which snap mode you use when selecting the Polygon create mode.

Automatic creates the boundary using the aspect ratio, origin, and target area.

Aspect Ratio (W/H) sets the aspect ratio. Aspect Ratio W/H is the width-to-length ratio of the cluster boundary.
Environment variable: clusterBoundaryAspectRatio

Origin X and **Origin Y** set the origin point.

Create as ROD Object creates the boundary as a ROD object.

Name lets you assign a name to the new boundary in the current layout cellview. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `boundary`. The prefix is followed by a number. For example, the first ROD boundary would be named `boundary0`, the second, `boundary1`, and so on.

© 2020 All Rights Reserved.

Area Information

Target Area is a float area value in user units.

Apply creates the blockage using the points derived from the aspect ratio, the x and y origin, and the target area.

Related Topics

Creating Cluster Boundary

leHiCreateClusterBoundary

© 2020 All Rights Reserved.

Create Custom Placement Area Form

Create Method creates an area in three different modes:
Environment variable: `customPlacementAreaCreateMethod`

Manual mode lets you create a custom placement area by creating a rectangle using two points with an area size greater than zero. In this mode, the coordinates are not necessarily on placement grid.

Assisted mode lets you use all the options except the Number of Placement Areas field. If the values are not compatible with the placement grid coordinates the values are adjusted to multiples of the placement grid.

Auto mode lets you use all the options. If the values are not compatible with the placement grid coordinates the values are adjusted to multiples of the placement grid.

Definition Section

Base Name assigns by default a unique name of CPA_integer, where the integer is incremented each time a custom placement area is created within the cellview. You can overwrite this name with a user assigned name, as long as the name is unique to the cellview.

Height sets the height of the placement area.
Environment variable: `customPlacementAreaHeight`

Width sets the width of the placement area.
Environment variable: `customPlacementAreaWidth`

Number of Placement Areas sets the number of custom placement areas to create.
Environment variable: `customPlacementAreaNumberOfAreas`

Channel Width sets the width between the placement areas you create.
Environment variable: `customPlacementAreaChannel`

Orientation sets the orientation of the placement areas.
Environment variable: `customPlacementFirstUnitOrient`

Flip Every Other Area lets you flip every other placement area. You can put grounds and power together when you flip the rows
Environment variable: `customPlacementAreaFlipUnit`

Flip First Area lets you flip the first custom placement area.
Environment variable: `customPlacementAreaFlipFirstUnit`

Abut boolean button lets you abut custom placement areas.
Environment variable: `customPlacementAreaAbutUnit`

© 2020 All Rights Reserved.

Areas lets you enter a positive integer that sets the number of custom placement areas that will abut together.
Environment variable: `customPlacementAreaAbutNumberOfUnits`

Abut First Area lets you abut the first placed custom placement area.
Environment variable: `customPlacementFirstUnitOrient`

Templates Section

Templates allow you to place rails and/or components automatically into the rows.

Type allows you to select *Rail*, *Allowed Component*, or *Both* in the *Pre-Defined* pane of the form. You move them to the *Selected* pane using the arrows. Once they are in the *Selected* pane, they are available to place in the cellview. You can then add the components individually selected in the *Selected* pane.

Related Links

Create Custom Placement Area

leHiCreatePlacementArea

© 2020 All Rights Reserved.

Create and Edit Clusters Form

The Create and Edit Clusters Form is divided into quadrants.

Lower Right Quadrant:

Use cyclic field determines which objects from the cellview are displayed in the Filter Instances and Clusters area. The choices are Clusters, Instances, and Instances in selected set.

Name is a filtering mechanism using a combination of a == or =! cyclic field and a filter field which accepts any regular expression.

Cluster Type when Clusters is selected in the Use cyclic field the second filtering mechanism becomes Cluster Type. The cyclic field choices are All, inclusive, exclusive and suggested.

Cell Type when Instances or Instances in selected set is selected in the cyclic field the second filtering mechanism becomes Cell Type. The cyclic field choices are All, none, block, blockRing, cover, coverBump, pad, padSpacer, padAreaIO, core, coreSpacer, coreAntenna, corner, softMacro, blockBlackBox, and via.

Filter By button displays the objects selected by the filtering process and displays them in the Filter Instances and Clusters section of the form.

Add to Cluster button moves instances or clusters you have selected in the Filter Instances and Clusters section of the form to the Current Cluster Editing Area of the form. The selected instances and clusters are added to the trunk, or if you select the branch in the Current Cluster Editing Area the instances and clusters are added to the correct branch.

Lower Left Quadrant:

Filter Instances and Clusters displays the names, types, library name, cell name, and view name of instances, clusters and sub clusters in the cellview.

Upper Left Quadrant:

Clusters cyclic field contains all of the current clusters defined in the cellview. This field is used to change the information displayed in the Current Cluster Editing Area.

Type cyclic field displays the types of clusters (inclusive, exclusive and suggested) that are assigned to the cluster displayed in the Clusters cyclic field. Choose the clusters from the Clusters cyclic field then use the Type cyclic field to change what type they are.

© 2020 All Rights Reserved.

Current Cluster Editing Area displays clusters and instances in tree table format, and allows you to edit the cluster and sub cluster.

Upper Right Quadrant:

Name creates a cluster of the name you enter in the field.

Type assigns a type to the newly created cluster. The options are inclusive, exclusive and suggested.

Create creates a new cluster, once you have entered the name and type in the Name and Type fields.

Create Sub Cluster creates a new sub cluster, once you have entered the name and type in the Name and Type fields.

Create Cluster Boundary opens the Create Cluster Boundary form, which allows you to enter a cluster boundary in the cellview. The Create Cluster Boundary form is seeded with the current cluster information in the cellview.

Remove From Cluster removes instances or sub clusters selected in the Current Cluster Area from the cluster.

Flatten removes instances or sub clusters you have selected in the Current Cluster Area from the currently displayed cluster by moving the selected elements to the parent one level up. If no parent exists the objects will be part of the design pool. You can also control whether to move associated blockages one level up or not. If you move the blockages to the design pool, the blockages will exist without any owner.

Delete removes instances or sub clusters you select in the Current Cluster Area from the currently displayed cluster.

Cross-Highlight Instances highlights instances in the cellview window that are included in the currently displayed cluster, when you select the check box.

Cross-Highlight Boundaries highlights cluster boundaries in the cellview window that are included in the currently displayed cluster, when you select the check box.

Close ends the command and closes the form.

Related Links

leHiCreateClusters

© 2020 All Rights Reserved.

Create Encore Report Form

Use the **Create Encore Report** form to customize the report format to meet your requirements.

Headers lets you customize the header information for your report.

- Select the elements to include from the left-hand pane and click >> to move them to the right-hand pane.
- *Move Up* and *Move Down* let you change the order of the elements in the right hand pane.
- The << button removes the selected elements from the right-hand pane. The available elements are:

Element	Description
Empty Line	Inserts a blank line
Separator	Inserts a line of asterisks
Title	Inserts the title “Pad Opening Locator”
Created	Inserts the date and time the report was generated
Libname	Inserts the name of the library
Cellname	Inserts the name of the cell
Pad Label Hierarchy	Shows the hierarchy level that was searched for existing pad labels. (This is the value set in the *Label/Name: Search Depth* field in the *Search Criteria* section of the Pad Opening Info Form.)
Num Pads Found	Shows the total number of pad shapes that match the search criteria
Num Labeled Pads	Shows the number of pad shapes that have labels
Num Unlabeled Pads	Shows the number of pad shapes that do not have labels

© 2020 All Rights Reserved.

Columns lets you specify what information is included for each pad shape in the report. The use model is the same as that described above for the **Headers** section. The available columns are:

Column	Description
Pad Opening Name	The label name (if any), otherwise NO_NAME.
X	The X coordinate of the center point of the pad shape.
Y	The Y coordinate of the center point of the pad shape.
Area	The area of the pad shape.
Width:Height	The width and height of the pad shape.

File Path lets you specify the location and file name for the generated report. Type the full path in the field provided, or use the *Browse* button to navigate to the required location on disk.

Filter lets you specify filters to limit the data shown in the report. Click **Set Filter** to open the Pad Opening Info: Filter Form where you can specify the filters to be applied. See Setting Filters for a Customized Report for more information.

Related Topics

Generating a Customized Report

© 2020 All Rights Reserved.

Create Geometric Wire Form

See Creating Geometric Wires for more information.

See Create Geometric Wire Form in the *Virtuoso Interactive and Assisted Routing User Guide* for description of the fields in this form.

© 2020 All Rights Reserved.

Create Group Form

Name the group name.

Orientation sets whether the group is rotated or mirrored. See **Orientation Valid Values**.

Related Topics

Creating Groups

leHiCreateGroup

© 2020 All Rights Reserved.

Create Guard Ring Form

For information about the Create Guard Ring form, see Create Guard Ring Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Fluid Guard Ring Context-Sensitive Menu

For information about the Fluid Guard Ring Context-Sensitive menu, see Fluid Guard Ring Context-Sensitive Menu in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Create Instance Form

Browse... lets you select the library, cell, and view names by clicking on them in the browser.

Library, **Cell**, and **View** set the library, cell, and view names of the master cell you want to place as an instance in this cellview.

Note: A thumbnail of the library, cell, and view that you have selected will be displayed on the Create Instance form. The *Library*, *Cell*, and *View* text fields, which can be auto-completed. When you type the characters, the drop-down list shows the names containing the matching characters at any position in the name (beginning, middle, or end of the name). Select from the drop-down list and specify the name in the text field. The drop-down list is retrieved from cds.lib. The fields are highlighted if there are no matching library or cell or view names for the characters mentioned in the field.

Names sets the name assigned to this instance. You can type any name unique to the cellview here or let the layout editor automatically assign instance names that begin with the letter `I`, followed by a number. You can enter multiple names (separated by a space) to place several instances of the same cell. If the design has EMH, you can specify a hierarchical path name for creating the instance at the desired location in the EMH.

Halo turns on or off the ability to define halo blockages for the instance.

Define Halo opens the Halo Editor Form.

Physical Only creates an instance in such a way such that it exists only in the block domain and not in the module domain. This option is enabled only for instances and is disabled for mosaics. Once you create an instance as a physical only instance, you cannot change its domain visibility. This attribute is used for filler (pad and core) and corner cells.
Environment variable: instPhysicalOnly

Mosaic

> **Create as Mosaic** lets you create an instance array when you select the check box.
> Environment variable: instMosaicCreation
>
> **Note:** When this option is selected, the *Physical Only* option is disabled.
>
> **Rows**, **Columns** set the number of rows and columns in an array of instances. These fields are enabled when the mosaic check box is selected.
>
> **Delta Y, X** set the spacing between rows and columns in an array of instances. The value of *Delta X* and *Delta Y* fields is calculated based on the value (*BBox*, *P&R Boundary*, or *Snap Boundary*) specified in the *Instance Editing Mode* field of the Layout Editor Options Form. These fields are enabled when the mosaic check box is selected.

© 2020 All Rights Reserved.

Parameters

If the instance has associated parameters, they are listed in the Parameters section.

The orientation buttons at the bottom of the form let you change the orientation of the selected instance.

Cluster adds the instance being created to an existing cluster in the design.

Cluster drop-down list specifies the name of the cluster to which the instance is added. Choose an existing cluster from the drop-down list or type the name of a new cluster into the field.

Type specifies the type of cluster; *inclusive*, *exclusive*, or *suggested*.

Rotate rotates an instance or array 90 degrees counterclockwise and clockwise each time you click the middle mouse button. Rotation of a mosaic is applied to the complete array.

Note: (ICADVM20.1 Virtuoso RF Solution Only) For SiP object types, the Rotate button is disabled. In addition, rotation using the middle mouse button during Create Instance is disabled for SiP object types. You can rotate SiP objects by changing the value of the angle parameter in the Parameters section of the Create Instance form.

Sideways and **Upside Down** mirror the instance. As you place the instance, press `Ctrl` and click the middle mouse button to mirror the instance *Sideways* or press `Shift` and click the middle mouse button to mirror the instance *Upside Down*.

At the first click, the instance mirrors along the X axis.

© 2020 All Rights Reserved.

At the second click, the instance mirrors along the Y axis.

Hide closes the form and lets you continue with the command.

Cancel closes the form without executing the command.

Defaults displays all the default CDF parameters values.

Related Topics

Creating Instances

leHiCreateInst

© 2020 All Rights Reserved.

Create Label Form

Mode sets the method for generating labels.

Manual derives label name from the text string specified in the *Label (Pattern)* field and places the label at the point of click. *Manual* is the default mode of creating labels. You can place multiple same or different labels on a single object. In the *Manual* mode, you can label any type of shape and object except the DB object types 'textDisplay' and 'label'. You can also add labels in empty spaces.

Auto automatically generates labels from the connectivity information on objects and places the label at the point of click. *Auto* also automatically generates labels for instances from the instance name or instance master cell and places the label at the center of the instance.

Two tabs become available in this mode, From Interconnect and From Instance. There is a check box above each tab. You can select one or both the check boxes to enable label creation from connectivity information or instances or both.

In the *Auto – From Interconnect* mode, if the shape to be labeled is a pin, the label name is derived from the pin's terminal name. For other shapes, the label name is derived from the net name associated with the shape. You can place multiple labels on a single object, but all the labels on an object will have the same label name, derived from the object's connectivity information.

Label (Pattern) specifies the text or indexed pattern to use for assigning label names. You specify the pattern inside the < and > symbols. These symbols are retained in the generated labels and represent bus bits. You can also use the [and] symbols to enclose the pattern, in which case, the symbols are not part of the labels generated.

The following table provides some examples of the patterns that can be specified.

Pattern Definition	**Example Pattern**	**Label Names Generated When *Expand* is on**
By using < >		
Simple naming for each label	J clk q c<1> c<2>	J clk q c<1> c<2>
Simple sequential naming of the type <n1:n2>	A<0:3>	A<0>, A<1>, A<2>, A<3>
Simple sequential naming of the type <<n1:n2>>	A<<0:3>>	A<0>, A<1>, A<2>, A<3>
Mixed sequential naming	A<2:3> clk B<1:0>	A<2>, A<3>, clk, B<1>, B<0>

© 2020 All Rights Reserved.

Pattern Definition	**Example Pattern**	**Label Names Generated When *Expand* is on**
Interleaving using '-'	A<1:4>-B<0:1>	A<1>, B<0>, A<2>, B<1>, A<3>, A<4>
	A<1:2>-B<1:5:2>-C<1:2>.....n	A<1> B<1> C<1> .... A<2> B<3> C<2>.....n
	A[[1:5:2]]-B<[0:1]>	A[1], B<0>, A[3], B<1>, A[5]
Increment label index by more than 1 by using the format <n1:n2:n+>, where n+ is the increment value	A<0:6:2>	A<0>, A<2>, A<4>, A<6>
Decrease label index by more than 1 by using the format <n1:n2:n->, where n- is the decrement value	B<7:0:2>	B<7>, B<5>, B<3>, B<1>
By using [ ]		
Simple sequential naming	A[0:2]	A0, A1, A2
	A[[0:2]]	A[0], A[1], A[2]
	A[1:2]B[1:2]	A1B1, A1B2, A2B1, A2B2
	A[1:2]C	A1C, A2C
	A[1:5:2]C	A1C, A3C, A5C
Interleaving using '-'	A[1:2]-B[1:2].....n	A1 B1 A2 B2.....n
Bus label	A[1:2] B[1:2]	A1B1 A1B2 A2B1 A2B2
By using both <> and [ ]		
Simple sequential naming		
Example 1	A[<0:2>]	A[0], A[1], A[2]
Example 2	A<[0:2]>	A<0>, A<1>, A<2>

If you use an escape character (\) preceding a sequential pattern, the expansion is overridden.

You can include special characters in the label name. Spaces in a pattern are used to distinguish the different label names in the sequence. After you have placed all the labels of a sequence and if *Keep Label* is selected, the labels in the sequence are re-generated and are available to be placed. If, however, *Keep Label* is off, you are prompted to specify the label text again in the *Label (Pattern)* field.

© 2020 All Rights Reserved.

Note: With *Keep Label* selected, the labels are re-generated only if the *Create Label* command is in repeat mode. See Repeating Commands.

The *Label (Pattern)* field is active only in the *Manual* mode. In the *Auto* mode, the label names are generated automatically from connectivity information of shapes or from instances. Therefore, the *Label (Pattern)* field is not available in the *Auto* mode.

The form comprises the following sections:

- Label Layer/Purpose
- Label Options
- Scan Line
- Auto Step

These sections and various options in these sections are available based on the mode you select to create labels.

Label Layer/Purpose

This section determines the layer and purpose for the label.

Use current entry layer places the label on the active layer displayed in the Layers Panel. This option is selected by default.

Use same layer as shape, select purpose places the label on the same layer on which the shape is drawn. You select the purpose from list. A warning is displayed in the CIW if the combination of the shape layer and the selected purpose is not valid. You cannot use this option to add labels to empty spaces in the cellview. This option does not support labeling of instances, vias, groups, and mosaics.

Auto (Using stampLabelLayers section from technology file) derives the layer and purpose on which to create the label from the stampLabelLayers section of the technology file. If the `stampLabelLayers` section does not exist in the technology file, a warning is displayed in the CIW. You cannot use this option to add labels to empty spaces in the cellview. With this option selected, you cannot label instances, vias, groups, and mosaics.

© 2020 All Rights Reserved.

Consider the following sample of `stampLabelLayers` section from a technology file.

```
;*******************************
; LAYER RULES
;*******************************
layerRules(

 functions(
 ;( layer        function                [maskNumber])
 ;( -----        --------                ------------)
  ( METAL1       "metal"                 1           )
  ( VIAS1        "cut"                   2           )
  ( METAL2       "metal"                 3           )
  ( VIAS2        "cut"                   4           )
  ( METAL3       "metal"                 5           )
  ( VIAS3        "cut"                   6           )
  ( METAL4       "metal"                 7           )
  ( VIAS4        "cut"                   8           )
  ( METAL5       "metal"                 9           )
  ( VIAS5        "cut"                   10          )
  ( METAL6       "metal"                 11          )
  ( CONT         "CUT"                   12          )
  ( POLY1        "poly"                  13          )
  ( POLY2        "poly"                  14          )
  ( NMOAT        "ndiff"                 15          )
  ( PMOAT        "pdiff"                 16          )
 ) ;functions
 stampLabelLayers(
  ( ("METAL2" "drawing") METAL1 METAL3 )            //rule id1
  ( text METAL2 METAL5 )                            //rule id2
  ( text ("METAL2" "drawing") METAL4 METAL5 )       //rule id3
  ( ("text" "drawing") ("METAL3" "drawing") METAL6 )//rule id4
  ( ("Poly" "drawing") METAL1 METAL5 )              //rule id5
  ( pin VIAS1 VIAS2 CONT )                          //rule id6
 )

) ;layerRules
```

Each statement in the `stampLabelLayers` section defines a rule for creating a label on a particular layer-purpose. The first layer/purpose entry of a rule indicates the layer-purpose on which the label is created for a shape that exists on layer-purposes defined by the remaining entries of a rule. If a purpose is not specified in the first entry, the valid list of purposes for the specified layer is considered for creating the label. For example, in `rule Id1`, the first entry `"METAL2" "drawing"` represents the layer-purpose on

© 2020 All Rights Reserved.

which the label will be created. The remaining entries `METAL1 METAL3` represent the layers on which the shape to be labeled should exist.

If a layer or layer-purpose pair of the shape occurs in multiple rules, the first encountered rule for that layer or layer-purpose is used for deciding the layer-purpose for the label. For example, if a shape exists on layer `METAL5`, `rule Id2` will be used even though `rule Id2`, `rule Id3`, and `rule Id5` are all `METAL5` rules. Similarly, for a shape on `METAL3`, `rule Id1` will be used. If you attempt to label a shape whose layer or layer-purpose pair are not part of any rule, an error message will be issued in the CIW. In the above sample `stampLabelLayers` section, there are rules for the layers `METAL1`, `METAL2`, `METAL3`, `METAL4`, `METAL5`, `METAL6`, `VIAS1`, `VIAS2`, and `CONT`. You cannot create a label for a shape that exists on layer `POLY` because no rule for the `POLY` layer has been defined in the `stampLabelLayers` section.

In the sample section above, `rule Id1` specifies that if the shape to be labeled is on `METAL1`-any purpose or `METAL3`-any purpose, the label will be created on `METAL2` purpose `drawing`. Similarly, `rule Id2` specifies that if a shape is on `METAL2`-any purpose or `METAL5`-any purpose, the label will be created on `text` layer and on a valid purpose defined for the `text` layer in technology file. `rule Id4` states that the label will be created on `text` layer purpose `drawing` only if the shape exists on `METAL3` purpose `drawing` or `METAL6`-any purpose. If the shape does exist on `METAL3`, `rule Id1` will be used for determining layer-purpose for label.

The following table explains all the rules of the sample `stampLabelLayers` section:

	If shape is on...		**Label will be created on...**	
Rule	**Layer**	**Purpose**	**Layer**	**Purpose**
`rule Id1`	`METAL1` `METAL3`	Any Any	`METAL2`	`drawing`
`rule Id2`	`METAL2` `METAL5`	Any Any	`text`	Valid purpose*

`rule Id3`	`METAL2`
	`METAL4`
	`METAL5`
		`drawing`
		Any
		Any
			`text`	Valid purpose*

`rule Id4`	`METAL3` `METAL6`	`drawing` Any	`text`	`drawing`

© 2020 All Rights Reserved.

`rule Id5`	`METAL1`	Any	`Poly`	`drawing`
	`METAL5`	Any		
`rule Id6`	`VIAS1`	Any	`pin`	Valid purpose*
	`VIAS2`	Any		
	`CONT`	Any		

where, Valid purpose is any valid purpose defined in the technology file for that layer.*

Select layer places the label on the layer purpose selected from the drop-down list. You can search for specific layer-purpose pair using the search filter in the *Select layer* list box.

Note: The layer selection using the search filter is available only if the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form.

Label Options

Font sets the font type of the label text. Choices include *eurostyle*, *gothic*, *math*, *roman*, *script*, *stick*, *fixed*, *swedish*, and *milSpec*. The default font is *stick*.

Height sets the height of the label in user units (usually microns). The default value is `1`.

Justification sets the location of the label origin relative to the position of the label text. The origin appears as a small square on the label when you place or select it. Choices include *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *centerRight*, and *upperRight*. The default is *centerCenter*.

Drafting prevents a label from being rotated by more than 90 degrees. This check box is on by default.

Overbar determines how text strings containing underscore characters are displayed. If you select this check box, any underscore character in the text string or the specified pattern displays as an overbar. This check box is off by default.

Keep Label retains the label text or pattern specified in the *Label (Pattern)* field after you have placed all the generated labels. If this check box is off, labels are consumed from the *Label (Pattern)* field as you place them and the field becomes empty after you have placed all the generated labels. You are then prompted to specify a label text in the *Label (Pattern)* field. This check box is on by default in the *Manual* mode. This check box is not available in the *Auto* mode because in this mode, labels are generated from the connectivity information or from instances.

© 2020 All Rights Reserved.

Attach attaches the label to the object you click in the cellview. The object to which you attach the label becomes the parent of the label. With the *Attach* check box selected, the label is automatically attached to the clicked shape. However, if you place the label in empty space, you are prompted to click a shape to which the label is to be attached. The *Attach* check box is on by default in the *Manual* mode. This check box is not available in the *Auto* mode because in this mode, labels are always attached to the shape.

Expand generates each label name of the sequence based on the pattern specified in the *Label (Pattern)* field. Consider the pattern `A<0:2> GND VDD`. If the *Expand* check box is selected, the generated labels will be `A<0>`, `A<1>`, `A<2>`, `GND`, and `VDD`. These labels will be assigned one by one to the shapes clicked. However, if the *Expand* check box is not selected, the single label `A<0:2> GND VDD` will be generated and the same label will be assigned to any shape you click (provided *Keep Label* is also selected). The *Expand* check box is on by default in the *Manual* mode. This check box is not available in the *Auto* mode because in this mode, labels are generated from the connectivity information or from instances.

Note: You can override the sequence expansion even if the *Expand* check box is selected by preceding the sequence pattern with the backspace character (\).

Scan Line

Label Selected Objects enables only the ed or post-selected objects to be labeled when the scan line intersects multiple objects.

Auto Zoom Out automatically zooms out the design display area and pan the design if you extend the scan line beyond the visible design area. The design returns to the current zoom level after you release the middle mouse button.

Auto Step

This section is available only in the *Manual* mode. These options enable you to create labels by specifying the distance between consecutive labels in the X and Y directions.

Step enables you to specify the distance between consecutive labels by using *Custom* or *Input X&Y* method.

Custom enables you to indicate the distance between labels and the direction of placing the labels by clicking in the design display area.

Input X&Y enables you to specify the distance between labels along the X and Y axes.

© 2020 All Rights Reserved.

X and **Y** enables you to specify the distance between the labels in the X and Y directions. These fields are grayed out if *Custom* is selected from the *Step* list.

The following tabs are available in the *Auto* mode.

- From Interconnect
- From Instance

From Interconnect

In this *Auto* mode, the form comprises the following sections:

- Label Layer/Purpose: All the options are available in this mode.
- Label Options: *Font*, *Height*, *Justification*, *Drafting*, *Create As Label*, and *Overbar* are available in this mode.

  Create As Label determines whether the object is created as a label or text display. If this option is selected, a label is created, else a text display is created.

  Note: When a shape with a textDisplay is removed from a net, the textDisplay is automatically removed as well. If the shape is added back to a net, the textDisplay is also automatically recreated.

  Environment variable: createAsLabel
- Scan Line: Both the options are available in this mode.

From Instance

In this *Auto* mode, the form comprises the following sections:

- Label Layer/Purpose: *Use current entry layer* and *Select layer* options are available in this mode.
- Label Options: *Font*, *Height*, *Justification*, *Drafting*, *Overbar*, and *Create As Label* check boxes are available in this mode.

  Show Name of enables you to specify whether to use the *instance* name or the instance *master* cell name for generating the labels.
- Scan Line: Both the options are available in this mode.

Snap Mode locks the direction in which you can draw the scan line. You can set it to *anyAngle*, *diagonal*, *orthogonal* (default), *horizontal*, or *vertical*. The default selection is

© 2020 All Rights Reserved.

orthogonal. The setting in this list is applicable while using the Scan Line and Auto Step (*Custom*) methods for generating labels.

Rotate turns the label 90 degrees counterclockwise.

Sideways mirrors the label along the X axis. Drafting must be off for this button to work.

Upside Down mirrors the label along the Y axis. Drafting must be off for this button to work.

Related Topics

Creating Labels

leHiCreateLabel

© 2020 All Rights Reserved.

Create Measurement Form

Type lets you select the display type of the ruler.
Environment variable: rulerDisplayType

Ruler displays the ruler in the design window.

Distance displays the distance between two points on the canvas.

Both displays both the ruler and the distance between two points on the canvas**.**

Segment Mode lets you select the type of the ruler created by the measurement.

Single lets you create a single-segment ruler. This is the measurement between points on the canvas.

Multiple lets you create a multiple-segment ruler. In this mode, you can view the measurement between consecutive points specified by the user.

Auto creates a multiple-segment ruler. In this mode, points are automatically calculated. There are two ways in which the ruler is created in the auto mode. The first way is an auto ruler is created between two specified points on the canvas. The second way is when you press the `Shift` key in the auto mode. In this case, an orthogonal ruler is created with all edges visible on the canvas in vertical or horizontal direction. In this case, the direction of the ruler is based on the edge closest to the pointer.

Note: The *Mid Point of Distance*, *Dynamic Measurement*, and *Edge Measurement* options are not available when the segment mode is auto.

Environment variable: rulerSegmentModeType

Mid Point of Distance lets you select the ruler point as a mid-point of the two points that you specify.

Snap Mode controls how the ruler snaps to the grid.

anyAngle creates the ruler segment at any angle to the X or Y axis.

diagonal creates the ruler parallel to the X or Y axis or at a 45-degree angle to the axes.

orthogonal creates the ruler parallel to the X or Y axis.

L90XFirst creates orthogonal two-segment rulers and creates the first segment in the X direction.

L90YFirst creates orthogonal two-segment rulers and creates the first segment in the Y direction.

© 2020 All Rights Reserved.

L90YFirst creates orthogonal two-segment rulers and creates the first segment in the Y direction.

Snap Target lets you select the snap target.

None does not snap to the target.

Edges snaps the ruler to only the edges.

Points snaps the ruler to vertices, midpoint of edges, nexus, end points, and origin point of objects.

Edges and Points snaps the ruler to edges, centerlines, vertices, nexus, and end points of objects.

Dynamic Measurement enables you to dynamically display measurement between the pointer and the closest edges.

If you move the pointer between two shapes, the distance between the two edges is displayed, as shown in the figure below.

© 2020 All Rights Reserved.

If you move the pointer on an edge of a shape, the edge is highlighted in yellow. Also, the distances between the highlighted edge and the two closest edges are displayed.

Video

For a video demonstration of the *Dynamic Measurement feature*, see *How to use Dynamic Measurement feature to measure distance* and *Dymanic Measurement in Create/Edit commands*.

Layer Scope lets you select the layers to be considered for dynamic measurement. You can select *All*, *Active*, *Edge*, *Palette*, or *Selectable* layers.

If you select *All*, all the layers are considered for dynamic measurement, as shown in the figure below.

M1and M2 considered for Dynamic Measurement

© 2020 All Rights Reserved.

If you select *Active*, only M1 is considered for dynamic measurement because M1 is the active layer. M2 is ignored.

If you select *Edge*, M1 is considered for dynamic measurement because the pointer is on an edge of M1, even though it is not the active layer.

© 2020 All Rights Reserved.

If you select *Palette*, all the layers on the Palette are considered for dynamic measurement (M1 to M11/slot). This option is available only if the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form.

If you select *Selectable*, only the selectable layers on the Palette are considered for dynamic measurement. In the figure below, M2 is not considered for dynamic measurement because it is not selectable.

Note: If you select the *Highlight Aligned Edges On* option on the Dynamic Display form, the edges of the shapes aligned to the pointer are highlighted, as shown in the figure below.

© 2020
All Rights Reserved.

Edge Measurement lets you enable the edge measurement mode for rulers. Edge measurement mode allows you to click the edge of an object and a measurement is drawn with the two vertices as the two ends. The first click lets you select the edge and the second click lets you place the measurement on the canvas.
Environment variable: rulerEdgeMode

Note: The Edge Management check box is disabled when the *Snap Mode* is none or when the *Segment Mode* is *Multiple* or *Auto*.

Savable Measurement saves the persistent rulers in a cellview when you save the cellview. It is on by default. If this option is off, all the rulers existing in the cellview are deleted when you save the cellview.
Environment variable: saveRulers

Display Options let you define the ruler and snap target visualization settings.

Font Size lets you select the font size of the ruler labels. You can choose from *Small*, *Medium*, or *Large*.
Environment variable: rulerFontSize

Color lets you select the font color of the ruler labels.

© 2020 All Rights Reserved.

Background Color lets you select the background color of the ruler labels.
Environment variable: rulerLabelBackgroundColor

Bold displays the ruler labels in bold face.
Environment variable: rulerFontBold

Snap Target Width enables you to specify the width of the snap target highlight. The default is `0`.
Environment variable: targetWidth

Snap Target Color enables you to select the color of the snap target highlight. The default is `yellow`.
Environment variable: targetColor

Snap Target Line Style enables you to select the type of snap target highlight. The default is `thickline`.
Environment variable: targetLineStyle

Related Topics

Using Rulers

leHiCreateMeasurement

© 2020 All Rights Reserved.

Ruler Context-Sensitive Menu

Right-clicking in the design display area when the *Create Measurement* command is active displays the *Ruler* context-sensitive menu. The context-sensitive menu offers the following options:

- Smart snapping off
- Snap to Edges
- Snap to Points
- Snap to Edges and Points
- Snap to Mid Point of distance between 2 targets
- Edge measurement
- Label bold font
- Label background color
- Options

Smart snapping off

Identifies and snaps ruler to objects in the vicinity of the pointer, based on the settings in the Gravity Controls section of the Layout Editor Options Form. The settings in the *Gravity Controls* section are applied only if *Gravity On* is selected. If *Gravity On* is not selected, ruler snapping to objects is disabled. In that case, you define the start and end points of a ruler by clicking at exact points in the design display area.

Snap to Edges

The *Snap to Edges* option is selected by default. This option highlights the nearest edge within the aperture distance from the pointer. Edges are highlighted depending on the option (*All*, *Edge, Centerline*) selected in the *Edges* section of the *Snapping Options* tab of the *Create Measurement* form. Up to five target edges in proximity of the pointer can be

© 2020 All Rights Reserved.

highlighted in succession by pressing the `Spacebar` key. Click to snap the pointer to the highlighted edge.

Snap to Points

This option highlights the points if they are within the aperture distance from the pointer. Points are highlighted depending on the option (*All*, *Vertex*, *Nexus*, *Edge Midpoint*) selected in the *Points* section of the *Snapping Options* tab of the *Create Measurement* form. Up to five target points in proximity of the pointer can be highlighted in succession by pressing the `Spacebar` key. Click to snap the pointer to the highlighted point.

Snap to Edges and Points

This option highlights all the types of edges and points if they are within the aperture distance from the pointer. The edges and points that are highlighted include:

- Point Types: *All*, *Vertex*, *Nexus*, *Edge Midpoint*
- Edge Types: *All*, *Edge, Centerline*

Up to eight targets in proximity of the pointer can be highlighted in succession by pressing the `Spacebar` key. Click to snap the pointer to the highlighted target. If a point and an edge are both potential targets, then the point is highlighted before edges.

Snap to Mid Point of distance between 2 targets

This option highlights the nearest edge within the aperture distance from the pointer. After you click (twice) to snap the pointer to two such edges, the pointer snaps to the mid point of the distance between the points snapped at the two edges. To identify the target edges, you can press the `Spacebar` key to highlight in succession up to five target edges in proximity of the pointer.

© 2020 All Rights Reserved.

Edge measurement

This option lets you enable the edge measurement mode for rulers. Edge measurement mode allows you to click an edge of an object and a measurement is drawn with the two vertices as the two ends. The first click lets you select the edge and the second click lets you place the measurement on the canvas.

Label bold font

This option enables you to display the ruler label in bold face.

Label background color

This option enables you to select the background color of the ruler label.

Options

Opens the Create Slot Form to configure global ruler settings.

© 2020 All Rights Reserved.

Create Multipart Path Form

Template Name shows the name of the currently selected template, or shows *New* if no template is selected.

MPP Template lets you select a template for a multipart path (MPP) or choose to create a new MPP.
Environment variable: mppTemplate

Name lets you assign a name for a new MPP in the current layout window. The name must be unique in the cellview. If you leave this field blank, the system assigns a unique name consisting of the prefix `path`, followed by a number. For example, for the first path in the cellview for which you do not specify a name, the system assigns the name `path0`, if unique; for the second, `path1`, and so on.

Technology Library is a cyclic field that lets you choose any templates that are defined in the “effective” technology library. Once you have chosen a technology library only the multipart path templates in the chosen library are displayed.

Choppable indicates whether or not the master path can be chopped. This field is valid only for a path that has at least one subpart. When a master path is choppable, all its subparts must be choppable. When a master path is not choppable, each of its subparts can be choppable or not. The system ignores this field for single-part paths. The default is choppable.

Width specifies the width of the master path. It must be a positive integer or floating-point number. If you do not specify the width, the system uses the `minWidth` rule for the master path layer from the technology file. If the `minWidth` rule is not defined in the technology file, and

- This is the first time you have used this form in your current editing session and you do not specify a width, the system supplies a default and displays an warning saying that the default width is being used.
- You have used this form earlier in your current editing session, the previous value for this field is displayed and the system displays a warning.

End Type defines how the ends of the master path are created. The default value is *flush*. If you want to specify a beginning or ending extension, you must choose *variable*. Subparts inherit the end type of the master path.

truncate creates ends that end at the first and last points of the path.

extend creates ends that extend beyond the first and last points of the path by one half the path width.

© 2020 All Rights Reserved.

round creates octagonal-shaped ends that extend beyond the first and last points of the path by one half the path width.

variable creates path ends that extend beyond or retract from first and last points of the path by the value you enter in the *Begin Extension* and *End Extension* fields.

Offset specifies the distance by which the master path is offset from the points you enter (the point list), in relation to the direction of the master path. The direction of the master path is determined by the order in which you enter the first and last point of each segment. The *Offset* value must be a signed integer or floating-point number; the default is `0`.

A positive value defines the master path to the left of the point list; a negative value defines the master path to the right of the point list. For a detailed description about offsetting the master path, see "About Offsetting the Master Path" in the *Virtuoso Relative Object Design User Guide*.

Begin Extension specifies the distance by which the starting edge of the master path extends beyond its first point. The value must be a zero or a positive integer or floating-point number; the default is `0`. This field is ignored unless *End Type* is *variable*.

Justification defines the part of the master path (the centerline, left edge, or right edge) that is offset from the points you enter (the point list). *Justification* is relative to the direction of the master path. The direction of the master path is determined by the order in which you enter the first and last point of each segment. The default is *center*. Together, the *Offset* and *Justification* fields determine the location of the master path in relation to the point list.

center specifies that the centerline of the master path is offset from the point list by the value you enter for the *Offset* field.

left specifies that the left edge of the master path is offset from the point list by the value you enter for the *Offset* field.

right specifies that the right edge of the master path is offset from the point list by the value you enter for the *Offset* field.

End Extension specifies the distance by which the ending edge of the master path extends beyond its last point. The value must be a zero or a positive integer or floating-point number; the default is `0`. This field is ignored unless *End Type* is *variable*.

Connectivity determines whether to associate the master path with a net and whether to make it into a pin. The default is *None*. When you choose *Net*, the Create Multipart Path form expands to show the net connectivity fields. When you choose *Pin*, the form expands to show both the net and pin connectivity fields.

None indicates that you do not want to define connectivity.

© 2020 All Rights Reserved.

Net indicates that you want to associate the master path with a specific net. When the *Connectivity* is set to *Net*, the Connectivity Section adds to the form where you are required to specify the net name.

Pin indicates that you want the master path to be a pin. When the *Connectivity* is set to *Net*, the Connectivity Section adds to the form where you are required to specify the net name.

Load Template opens the Load Multipart Path Template form to let you enter the path and file name for an ASCII file containing one or more MPP template definitions. When you click *OK*, the system adds the templates in the file to the MPP templates currently in virtual memory and adds the template names to the list of template names in the *MPP Template* field on the Create Multipart Path form. If the filename you entered does not exist, the system displays a dialog box.

Save Template opens the Save Multipart Path Templates form to let you save any one template named in the *Template List* field or save all templates named in the *Template List* field either to an ASCII file in your local hierarchy, or, if you have write permission, to your technology library. In both cases, the system saves the template in the temporary technology file in virtual memory. When you save to an ASCII file, and a file with the same name already exists, the existing file will be overwritten; see the following message in the CIW:

```
Template templateName is replacing an exisitng template by the same name.
```

Subpart opens the ROD Subpart form to let you define, change, or delete subparts for the associated master path. For a description of the ROD Subpart form, see the ROD Subpart Form.

Connectivity Section

When the *Connectivity* field is set to *Net*, the following field is displayed.

Net Name specifies the net with which you want to associate the master path or subpart. This field is required. If the net does not exist, the system creates it, using the name you type in this field. MPP templates do not contain net names, so when want to associate the MPP with a net, display the form and enter a name.

When the *Connectivity* field is set to *Pin*, the net and pin connectivity fields are displayed in the Connectivity section.

Net Name specifies the net with which you want to associate the master path or subpart. If the net does not exist, the system creates it, using the name you type in this field. MPP templates do not contain net names, so when you want to associate an MPP with a net, display the form and type a net name. When you choose *Pin*, you must type a net name

© 2020 All Rights Reserved.

before you create the MPP in the layout cellview window; if you do not, the system beeps when you click to begin creating the MPP.

I/O Type assigns a property used by routers to identify the direction of the signal into or out of the terminal. The default is *inputOutput*. If the terminal specified in the *Net Name* field already exists, it must have the same direction type as defined in the *I/O Type* field. If the direction type is not the same, the system displays an error in the CIW and no MPP is created.

input allows a signal to come in only.

output allows a signal to go out only.

inputOutput allows signals to go in and out simultaneously (bidirectional).

switch allows a signal to go either in or out, but not both simultaneously.

jumper allows a signal to pass through the net.

Access Direction defines the access directions for the pin. The default is *Top*, *Bottom*, *Left*, and *Right*. When you choose *Any*, the *Top*, *Bottom*, *Left*, and *Right* options remain on.

Display Terminal Name defines whether to create a text-display object using the net name and associate it with the pin you are creating. The default is to not create a text-display object.

Note: The system uses the values of four fields to calculate the position of the pin name: *Reference Handle*, *Offset X*, *Offset Y*, and the *Justification* field on the Pin Name Display form.

Display Terminal Name Option opens the *Set Pin Label Text Style* form to let you set values for how the pin name is shown, such as the font, height, layer, rotation, and justification. For a description of the *Set Pin Label Text Style* form, see Setting Pin Label Text Style Options.

Reference Handle specifies the name of the handle on the master path or subpart to which you want to align the pin name. In this context, a handle is a point associated with the object you are defining (the master path or a subpart), such as the starting point or ending point of the object, or a point on the bounding box of the object. The default is *centerCenter*. For a detailed description of handles, see "Accessing Handles on ROD Objects" in the *Virtuoso Relative Object Design User Guide*.

Offset X is a signed floating-point number specifying the distance of the origin of the pin name along the X axis from the point specified by *Reference Handle*. The default is `0`.

Offset Y is a signed floating-point number specifying the distance of the origin of the pin name along the Y axis from the point specified by *Reference Handle*. The default is `0`.

© 2020 All Rights Reserved.

Related Topics

Creating Multipart Paths

leHiCreateMPP

© 2020 All Rights Reserved.

ROD Subpart Form

Scroll Window lets you see a list of the offset, enclosure, or subrectangle subpaths that already exist for the MPP you are creating, depending on which of the following you select: *Offset Subpath*, *Enclosure Subpath*, or *Subrectangle*. There is one line of data for each subpath of the type selected. Each data element in a line corresponds to a field in the lower part of the ROD subpart form. When you click an existing subpart in the scroll window, the system fills the fields in the rest of the ROD Subpart form with the values for that subpart. You can add a new subpart, delete an existing subpart, or change the values for an existing subpart.

Add lets you create a new subpart from the values currently in the ROD Subpart form.

Delete lets you delete the highlighted subpart.

Edit lets you change the values for an existing subpart. Highlight the subpart you want to change, change the values in the ROD Subpart form, and then click the *Edit* button. The system replaces the field values of the highlighted subpart with the values currently in the ROD Subpart form.

Offset Subpath lets you see one line of data for each existing offset subpath in the Scroll window, and lets you create a new offset subpath. To define an offset subpath, turn on this button, and the ROD Subpart form shows the fields for offset subpaths. For a description of the offset subpath fields, see Offset Subpath Fields. For a detailed description of offset subpaths, see "About Offset Subpaths" in the *Virtuoso Relative Object Design User Guide*.

Enclosure Subpath lets you create a new enclosure subpath. To define an enclosure subpath, turn on this button, and the ROD Subpart form shows the fields for enclosure subpaths. For a description of the enclosure subpath fields, see Enclosure Subpath Fields. For a detailed description of enclosure subpaths, see "About Enclosure Subpaths" in the *Virtuoso Relative Object Design User Guide*.

Subrectangle lets you create a new set of subrectangles. To define a set of subrectangles, turn on this button, and the ROD Subpart form shows the fields for subrectangles. For a description of the subrectangle fields, see Subrectangle Fields. For a detailed description of subrectangles, see "About Sets of Subrectangles" in the *Virtuoso Relative Object Design User Guide*.

© 2020 All Rights Reserved.

Offset Subpath Fields

Scroll Window at the top of the form lets you see a list of the offset subpaths that already exist for the current MPP. There is one line of data for each offset subpath. Each data element in a line corresponds to a field in the lower part of the ROD subpart form. The data for an offset subpath is displayed in the following sequence:

Order of Element	Corresponding Form Field
1	Layer
2	Width
3	Choppable
4	Separation
5	Justification
6	Begin Offset
7	End Offset
8	List of connectivity elements, if any

When a subpart has connectivity, the connectivity data is listed in the sequence shown below. When Display Pin Name is equal to `t`, the following elements correspond to fields on the Pin Name Display form: *Height*, *Layer*, *Justification*, *Font*, *Text Options*, and *Orientation*.

Order of Element	Corresponding Form Field
1	Net Name
2	I/O Type
3	Connectivity = Pin
4	Access Direction
5	Display Pin Name
6	Height
7	Layer
8	Justification
9	Font
10	Text Options

© 2020 All Rights Reserved.

Order of Element	Corresponding Form Field
11	Orientation
12	Reference Handle
13	Offset X:Offset Y

Layer lets you choose the layer-purpose pair for the subpart. The default purpose is `drawing`. Choosing a layer-purpose pair for a subpart does not change the current layer in the *Layers* assistant.

If the layer-purpose pair you choose is not valid in the *Layers* assistant, the system displays a dialog box asking if you want to add the layer-purpose pair to the *Layers* assistant. If you agree, the system adds the layer-purpose pair, but does not change the current layer-purpose pair. If do not want to add the layer-purpose pair to the *Layers* assistant, choose a layer-purpose pair that is already in the *Layers* assistant or the system will use the current layer-purpose pair.

Note: The layer selection using the search filter is available only if the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form.

Choppable indicates whether or not the subpart can be chopped. The default is choppable. When the master path is choppable, all of its subparts must be choppable. When the master path is not choppable, each of its subparts can be choppable or not.

Begin Offset specifies the offset of the starting edge of the subpath from the starting edge of the master path.The value must be a signed integer or floating-point number. A positive number extends the beginning of the subpath beyond the beginning of the master path; a negative number retracts the beginning of the subpath from the beginning of the master path. The default is *End Offset*, if specified; otherwise `0`.

End Offset specifies the offset of the ending edge of the subpath from the ending edge of the master path.The value must be a signed integer or floating-point number. A positive number extends the end of the subpath beyond the end of the master path; a negative number retracts the end of the subpath from the end of the master path. The default is *Begin Offset*, if specified; otherwise `0`.

Width specifies the width of the offset subpath. The value must be a positive integer or floating-point number. If you do not specify the width, the system uses the `minWidth` rule for the offset subpath layer from the technology file. If the `minWidth` rule is not defined in the technology file, the system uses the width of the master path. If the `minWidth` rule is not defined in the technology file, and

© 2020 All Rights Reserved.

- This is the first time you have used this form in your current editing session and you do not specify a width, the system supplies a default and displays an warning saying that the default width is being used.
- You have used this form earlier in your current editing session, the previous value for this field is displayed and the system displays a warning.

Justification specifies from which part of the master path to separate the subpath, in relation to the direction of the master path. When the value is *left*, the right edge of the subpath is separated from the left edge of the master path. When the value is *right*, the left edge of the subpath is separated from the right edge of the master path. When the value is *center*, the centerline of the subpath is separated from the centerline of the master path. The default is *center*.

Separation specifies the distance between the centerline or an edge of the subpath and the centerline or an edge of the master path, depending on the value of the *Justification* field. The value must be a signed integer or floating-point number; the default is `0`.

Connectivity determines whether to associate the subpart with a net and whether to make it into a pin. The default is *None*. When you choose *Net*, the ROD Subpart form expands to show the net connectivity fields. When you choose *Pin*, the form expands to show both the net and pin connectivity fields.

None specifies that you do not want to define connectivity.

Net specifies that you want to associate the subpart with a specific net. Type a value in the *Net Name* field.

Pin specifies that you want the subpart to be a pin. To define a pin, you must also specify a value for the *Net Name* field.

For a description of the subpart connectivity fields, see Connectivity Section.

Enclosure Subpath Fields

Scroll Window at the top of the form lets you see a list of the enclosure subpaths that already exist for the current MPP. There is one line of data for each enclosure subpath. Each data element in the line corresponds to a field in the lower part of the ROD subpart form.
Enclosure specifies the enclosure of the subpath in relation to the master path. To calculate the width of an enclosure subpath, the system **subtracts** two times the enclosure value from the width of the master path:

```
Width of Enclosure Subpath = Width of Master Path - (2 * Enclosure Value)
```

The value must be a signed integer or floating-point number. If you do not specify this field, the system uses the `minExtensionDistance` rule from the technology file for the master

© 2020 All Rights Reserved.

path layer to the subpath layer. If the `minExtensionDistance` rule is not defined in the technology file, the system sets the value to zero.

Connectivity is the same as for offset subpaths.

Subrectangle Fields

Scroll Window at the top of the form lets you see a list of the sets of subrectangles that already exist for the current MPP. There is one line of data for each set of subrectangles. Each data element in the line corresponds to a field in the lower part of the ROD subpart form.

Begin Offset specifies the offset of the starting edge of the first subrectangle from the starting edge of the master path.The value must be a signed integer or floating-point number. A positive number extends the beginning of the subrectangles beyond the beginning of the master path; a negative number retracts the beginning of the subrectangles from the beginning of the master path. The default is *End Offset*, if specified; otherwise `0`.

End Offset specifies the offset of the ending edge of the last subrectangle from the ending edge of the master path.The value must be a signed integer or floating-point number. A positive number extends the end of the subrectangles beyond the end of the master path; a negative number retracts the end of the subrectangles from the end of the master path. The default is *Begin Offset*, if specified; otherwise `0`.

Begin Segment Offset specifies the offset of the first rectangle in a segment from the edge of the master path.The value must be a positive integer or floating-point number. Does not apply to the first segment of a subrectangle subpath; the position of the first rectangle in the first segment is determined by the value of the *Begin Offset* option.

If the segment begins before the full distance specified by *Begin Segment Offset*, then it is not applied, and a subrectangle could be placed at the corner of the subrectangle path. The

© 2020 All Rights Reserved.

system will not create a spacing error; if the first subrectangle in a segment would be too close to the last subrectangle in the previous segment (if there would be less space than you specified for the *Space* option), the system discards the last subrectangle in the previous segment. The default is `0`.

End Segment Offset specifies the offset of the last rectangle in a segment from the edge of the master path.The value must be a positive integer or floating-point number. Does not apply to the last segment of a subrectangle subpath; the position of the last rectangle in the last segment is determined by the value of the *End Offset* option.

If the segment ends before the full distance specified by *End Segment Offset*, then it is not applied, and a subrectangle could be placed at the corner of the subrectangle path. The system will not create a spacing error; if the first subrectangle in a segment would be too close to the last subrectangle in the previous segment (if there would be less space than you specified for the *Space* option), the system discards the last subrectangle in the previous segment. The default is `0`.

© 2020 All Rights Reserved.

Width specifies the width of the rectangle(s). The width is parallel to the width of the master path. The value must be a positive integer or floating-point number.

If not specified, the system uses the value of the *Length* field; if neither is specified, the system uses the `minWidth` rule for the subrectangle layer from the technology file. If the `minWidth` rule is not defined in the technology file, the system uses the width of the master path.

Length specifies the length of the subrectangle(s). The length is parallel to the master path centerline.

If not specified, the system uses the `minWidth` rule for the subrectangle layer from the technology file. If the `minWidth` rule is not defined in the technology file, you must specify a length; otherwise, the length remains zero, the system displays a warning in the CIW, and no subrectangles are created.

© 2020 All Rights Reserved.

Space specifies the distance between the edges of adjoining rectangles. The value must be a positive integer or floating-point number. If not specified, the system uses the `minSpacing` rule for the subrectangle layer from the technology file. If the `minSpacing` rule is not defined in the technology file, you must enter a number or the system displays an error in the CIW.

Gap specifies the method the system uses to place subrectangles within each segment. The default is *distribute*. The system always uses the value of the *Space* field for minimum space between subrectangles and calculates the maximum number of rectangles that fit in the segment. When the value of *Gap* is *distribute*, the system distributes the space around subrectangles as evenly as possible, in multiples of the grid space specified by `mfgGridResolution`. Any remaining space is left after the last subrectangle in the segment.

© 2020 All Rights Reserved.

When the value of *Gap* is *minimum*, the system places subrectangles *Space* apart until there is no space for another rectangle, then leaves the excess space after the last subrectangle in the segment.

Justification specifies from which part of the master path to separate the subrectangles, in relation to the direction of the master path. The default is *center*. When the value is *left*, the right edge of the subrectangles is separated from the left edge of the master path. When the value is *right*, the left edge of the subrectangles is separated from the right edge of the master path. When the value is *center*, the centerline of the subrectangles is separated from the centerline of the master path.

Separation specifies the distance between the centerline or an edge of the subrectangles and the centerline or an edge of the master path, depending on the value of the *Justification* field. The default is `0`.

Diagonal specifies whether to create subrectangles on diagonal portions of the path. Setting this option to `t` works only when diagonal path segments are 45 degrees and all subrectangles on the path are square.

Related Topics

Adding a New Subpart

© 2020 All Rights Reserved.

Create Path Form

The form comprises the following tabs:

- Path Options
- Slot Options

Path Options

Width enables you to specify the path width. The path width that you specify must be greater than the minimum width (`minWidth`) defined in the technology file for the layer on which you are creating the path. If you do not specify the path width or if you specify a value less than the minimum width of the layer, the path width is set to the minimum width defined for that layer. You can click the *Defaults* button to set the default minimum width for the path anytime during the path creation process.

Fixed Width, is selected, uses for all segments of the path the width that you specify in the *Width* field, even when you change layers.

If the *Fixed Width* check box is not selected, the path width is reset to the minimum width (`minWidth`) defined in the technology file for the layer when you change layers. If you click the *Defaults* button, it deselects the *Fixed Width* check box and resets the *Width* field to the minimum width (`minWidth`) defined in the technology file for the current layer.

Justification determines the edge of the path that you create: left, center, or right. If you set a path offset, *Justification* determines the edge of the path that is offset from the points you enter.

Offset offsets the edge that you use to create the path from the entered points. Select the path edge to offset from the *Justification* list.

End Type controls how the path ends are created.

> **truncate:** The path ends and path points end at the same points.
>
> **extend:** The path ends extend from the path points by one half the path width.
>
> **round:** The path ends extend from the path points by one half the path width, symmetrically in four points at each end.
>
> **variable:** The path ends extend from the path points by the value that you specify in the *Begin Extension* and *End Extension* fields.

© 2020 All Rights Reserved.

Begin Extension and **End Extension** enable you to specify a beginning and an ending extension when *End Type* is set to *variable*.

Net Name assigns the path to a specific net.

Create as ROD Object creates the path as a ROD object.

> **Name** assigns a unique name to the new path in the current cellview if the *Create as ROD Object* check box is selected. This unique name consists of the prefix `path` followed by an integer. For example, the first ROD path is named `path0`, the second `path1`, and so on. You can edit the default name, if required, but ensure that the new name is unique in the cellview.

Note: When the *Create as ROD Object* option is selected, the path with slots is created as a ROD object with subrectangles as subparts. When the *Diagonal* option is selected in the ROD Subpart form, it specifies whether to create subrectangles on diagonal portions of the path. This option works only when diagonal path segments are 45 degrees and all subrectangles on the path are square.

Enable Metal Slotting makes available the Slot Options tab. On this tab, you specify the settings for creating a path with slots.
Environment variable: slotEnablePath

Snap Mode controls how path segments snap to the grid. Following are some *Snap Mode* examples:

To switch between the *L90XFirst* and *L90YFirst* snap modes while creating a path, click the middle mouse button.

Acute Angle enables you to create paths that enclose acute angles. This check box is available only when *Snap Mode* is set to *anyAngle*. When this check box is not selected, you can create paths with angles ranging from 90 degrees through 180 degrees.

Slot Options

See the Create Slot Form for a description of fields and options. For more information about creating slots in an existing path, see Creating Slots.

© 2020 All Rights Reserved.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The fields in the *Slot Options* tab are not supported for curved shapes.

Related Topics

Creating Paths

leHiCreatePath

© 2020 All Rights Reserved.

Create Pin Form

Mode

Manual enables you to create and place pins manually.

Auto enables the automatic pin placement mode. See the Auto Mode Options.

Environment variable: pinMode

Manual Mode Options

Connectivity

Strong lets you create multiple pins as strongly connected. You can add multiple pin figures to the selected pin, to define all of them as strongly connected. This option is selected by default.

Weak lets you create multiple pins as weakly connected.

Environment variable: defCreatePinConnModel

Must Connect All Pins lets you apply the `LEF58_MUSTJOINALLPORTS` property to all newly created pin terminals in the layout, which implies that the router must connect all pins of the selected terminal.

Environment variable: mustConnectAllPins

Terminal Names assigns a name to this pin. Terminal names are used to identify the net to which this pin connects. You can create multiple pins by specifying a space separated list of terminal names, one for each pin.

Physical Only creates a terminal in such a way so that it exists only in the block domain and not in the module domain. Once you create a terminal as a physical only pin, you cannot change its domain visibility. This attribute applies only to terminals and not to pins. If the terminal name specified in the *Terminal Names* field already exists in the design (either in block or module domain or in both), switching the *Physical Only* option on or off will not be effective. This is because no new terminal will be created, and the option will not change the existing terminal's attribute.

Keep First Name specifies that multiple pins or pin figures can be created with the same terminal name. When selected, this option does not remove the first name from the *Terminal Names* field. In the case of strongly connected pins, this option adds pin figures to the same pin shape on the same terminal. In the case of weakly connected pins, this option creates multiple pins on the same terminal.

© 2020 All Rights Reserved.

X Pitch sets the horizontal distance, in user units, between the center points of the pins. This field is editable when the terminal name represents a bus. For example, `data <0:7>`.

Y Pitch sets the vertical distance, in user units, between the center points of the pins. This field is editable when the terminal name represents a bus.

Create Label attaches a label, showing the terminal name, to the pin.

> **manual** lets you place the label manually on the pin.
>
> **auto** places the label automatically at the center of the pin. For polygonal pins, the label is placed at the boundary of the longest edge of the pin.

Environment variable: pinLabelPlacementMode

Note: The label is displayed for the pin only if the *Pin Names* option in the Display Options form is enabled.

Options opens the *Set Pin Label Text Style* form, which lets you set the font, height, layer name and purpose, justification, and orientation of the pin name.

Create as ROD Object creates the pin as a ROD object.

> **Name** lets you assign a name to the new shape pin in the current layout cellview. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `rect`, `dot`, or `polygon`, depending on the type of shape pin you are creating. The prefix is followed by a number. For example, the first ROD rectangle pin would be named `rect0`, the second, `rect1`, and so on.

Pin Figure specifies the shape of a manually created pin.

> **rectangle** lets you create a rectangular pin.
>
> **dot** lets you create a dot pin. You drag a rectangle to define the extent of the dot. Click twice on the same point for a zero-area dot pin.
>
> **polygon** lets you create a polygon pin.
>
> **circle** lets you create a circular pin.

Schematic Aware specifies whether pin creation must be schematic aware. When the option is selected, the tool uses the I/O type of terminal and signal type of net present in the corresponding schematic for the new pins. If the option is not selected or a corresponding schematic reference is not found, the values specified in the Create Pin form are used.

I/O Type assigns a property used by routers to identify the direction of the signal into or out of this cellview. The signal can be input, output, inputOutput (bidirectional), switch (carries

© 2020 All Rights Reserved.

data either in or out, but not simultaneously), or jumper (passes data through this cellview), unused, tristate.

Snap Mode limits how the pointer snaps when you create a polygon pin. This option is disabled when you are creating strongly connected pins.

anyAngle creates lines at any angle.

diagonal creates lines parallel to the X or Y axis or at a 45-degree angle to the axes.

orthogonal creates lines parallel to the X or Y axis. This is the default setting.

L90XFirst creates orthogonal two-segment lines and creates the first line in the X direction.

L90YFirst creates orthogonal two-segment lines and creates the first line in the Y direction.

Access Direction assigns a property used to identify the part of the pin to which routers can connect routing.

Signal Type lets you set the signal type of the net that is used for creating the pin. The options available for signal type are *signal*, *ground*, *power*, *clock*, *analog*, *tieOff*, *tieHi*, *tieLo*, *scan*, *reset*, *optical*, *singleModeOptical*, or *multiModeOptical*. The default is *signal*.

Auto Mode Options

When mode is set to auto, a pin is placed at the end of the wire or the path nearest to the point you clicked. The pin width matches the shape and width of the wire or path and it is placed on the same layer as the shape. This feature works only on wire or path ends parallel to the X or Y axis. The access direction is automatically set in this mode. You can create auto pin on rectangle, polygon, path, pathseg, and ellipse.

Note: X and Y Pitch, Pin Shape, and Access Direction options are not available in auto mode.

The following additional fields are displayed in this mode:

Pin Layer/Purpose lets you choose LPP to be used for auto pin creation. You can select the *Use current entry layer* option if you want to use the entry layer for auto pin creation. You can use the *Use same layer as shape, select purpose* to use the layer of the selected shape and select a purpose from the list of purposes.
Environment variables: pinLPP

Pin Size enables you to create a pin of type rectangle and of width and height you specify at the point of click or where the scan line crosses the shape. When you use this option, there

© 2020 All Rights Reserved.

can be multiple overlapping shapes. The LPP of the chosen shape will be of the shape with dynamic highlight and it will be shown in drag mode. In case of scan line, instead of creating a rectangular pin at the either ends of the path and pathseg, a rectangular pin of width and height where the scan line crosses the shape will be created.
Environment variables: autoModePinSize**,** autoPinRectHeight**,** autoPinRectWidth

Justification enables you to set the placement of the rectangular pin. You can choose *from lowerLeft*, *upperLeft*, *centerLeft*, *lowerCenter*, *centerCenter*, *upperCenter*, *lowerRight*, *upperRight*, and *centerRight*. The default value is *centerCenter*.

Snap to Wire End lets you snap the pin created on pathseg and rectangle to the edge. If this option is not selected, the pin is created at the point of click on the pathseg. The pin width is aligned to the pathseg width or the shorter edge of the rectangle.

Create Pin Options: Virtuoso Photonics Solution (ICADVM20.1 Photonics Only)

Note: You will need the `Virtuoso_Photonics_Option` license to use the functionality discussed in this section. For information on obtaining the required license, contact your local Cadence representative.

The following options are displayed when the layer function of the entry layer is waveguide or the pin is created on a net whose signal type is optical (with the signal type option as *optical*, *singleModeOptical*, or *multiModeOptical*). These options are available only in the manual pin creation mode.

Optical pins, which are an essential part of a photonics solution, are created by defining the signal types as optical. The optical signal type further contributes in creating the optical ports and waveguides that can be identified by assigning a distinctive color and line style.

In photonics, a waveguide is used to guide light as optical modes and it is curved. A photonic port is represented as a circular pinFig.

The three properties associated with every photonic port are described below:

Photonic Width enables you to specify the width of the facet of the waveguide that connects to this pin.
Environment variables: photonicPinWidth

Photonic Angle enables you to specify the angle of the facet of the waveguide at the intersection of the pin.
Environment variables: photonicPinAngle

© 2020 All Rights Reserved.

Photonic Radius enables you to specify the radius of the curve of the waveguide center at the point it meets the facet.
Environment variables: photonicPinRadius

You can edit an optical pin using the options in the Edit Photonics Pin Properties form. For more information, see Edit Properties Options: Virtuoso Photonics Solution (ICADVM20.1 Photonics Only).

A facet is the edge of a waveguide at which the pin is placed and connected to another waveguide, as shown in the figure below.

Related Topics

Creating Pins

© 2020 All Rights Reserved.

Create Pins From Labels Form

Creation Options let you control the scope of pin generation, and which objects the system uses to create pins.

Objects Within

Cellview creates pins only at the top-level cellview.

Selected Instances creates pins within the currently selected instances.

Object Type specifies which objects are used to create pins.

Label creates pins from labels.

Text Display creates pins from text displays.

Both creates pins from both labels and text displays.

Objects

All creates pins for all the objects of the specified type.

Selected creates pins only for the currently selected objects of the specified type.

Attach Label to Pin attaches a label to the pin.

Define With

Layer shows layers in the *Pin Layer* field, and the default width and length in the *Width* and *Length* fields.

Text shows all pin names in the *Objects Text* field, their default layers in the *Pin Layer* field, and the default width and length in the *Width* and *Length* fields.

Objects Layer shows the layers on which text labels were found when *Define With* is set to *Layer* and the *Object Type* is set to *Labels* (or *Both*). There is a single entry for each layer of text labels selected. This field is not editable.

Pin Layer shows the available layers for pins to be created on. Choose the layer you want from the drop down list.You can search for specific layer-purpose pair using the search filter.

Note: If the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form, only the layers selected using the search filter in the *Layers* panel of the Palette are visible in the *Pin Layer* list box

Width sets the width of the pins to be created. The default width is selected in this order:

1. The net width set in the Constraint Manager (if the pins are created)

© 2020 All Rights Reserved.

2. The `minWidth` spacing rule for the layer if no constraint exists
3. The `pinWidth` environment variable, if it is set

Length sets the length of the pins to be created. By default, the length is the same as the default width.

Ignore specifies that no pin is to be created for that entry.

Related Topics

Creating Pins from Labels

leHiCreatePinsFromLabels

© 2020 All Rights Reserved.

Create Polygon Form

Snap Mode limits how the pointer snaps when you create a polygon.

anyAngle creates lines at any angle.

diagonal creates lines parallel to the X or Y axis or at a 45-degree angle to the axes.

orthogonal creates lines parallel to the X or Y axis.

L90XFirst creates orthogonal two-segment lines and creates the first line in the X direction.

L90YFirst creates orthogonal two-segment lines and creates the first line in the Y direction.

Add an arc segment lets you create an arc in the polygon

Note: (ICADVM20.1 Virtuoso RF Solution Only) If the *Add an arc segment* option is selected, the curved polygon and the arc segment are created using the LPP selected in the palette.

Net Name assigns the polygon to a specific net.

Probe Nets highlights nets while you are creating interconnect.

ROD

Create as ROD object creates the polygon as a ROD object.

Name assigns a unique name to the new polygon in the current cellview if the *Create as ROD Object* check box is selected. This unique name consists of the prefix `polygon` followed by an integer. For example, the first ROD polygon is named `polygon0`, the second `polygon1`, and so on. You can edit the default name, if required, but ensure that the new name is unique in the cellview.

Related Topics

Creating Polygons

Selecting a Layer Automatically when Creating a Path or Shape

leHiCreatePolygon

© 2020 All Rights Reserved.

Create P&R Boundary Form

Create Method chooses which method is used to create the P&R boundary object.
Environment variable: createBoundaryMethod

> **From Selected Set** creates the P&R boundary using the rectangular bounding box of the selected set of objects.
>
> **Rectangle** creates the P&R boundary using two points.
>
> **Polygon** creates the P&R boundary using multiple points.

Apply creates the P&R boundary using the selected objects.

Snap Mode selects snap mode to use as you enter points when Polygon create method is selected.

Create as ROD Object creates the boundary as a ROD object.

> **Name** lets you assign a name to the new boundary in the current layout cellview. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `boundary`. The prefix is followed by a number. For example, the first ROD boundary would be named `boundary0`, the second, `boundary1`, and so on.

Define Halo opens the Halo Editor Form.

Related Topics

Creating P&R Boundary

leHiCreatePRBoundary

© 2020 All Rights Reserved.

Create Rectangle Form

Net Name assigns the rectangle to a specific net.

Probe Nets highlights nets while you are creating an interconnect.

Size

Enable smart snapping enables smart snapping (identifies objects in vicinity of the pointer intelligently and attaches to them) when the rectangle is created.
Environment variable: rectSmartSnapEnable

Active layer only enables smart snapping of the rectangle on only the current active layer only.
Environment variable: rectSmartSnapOnActiveLayer

Specify size enables you to click at the desired location on the canvas and place the rectangle.

Width enables you to specify the width of the rectangle.
Environment variable: rectWidth

Height enables you to specify the height of the rectangle.
Environment variable: rectHeight

Justification determines the base creation point of rectangle that you create: Bottom left, Bottom right, Top left, Top right, Center.
Environment variable: rectBaseMode

ROD

Create as ROD object creates the rectangle as a ROD object.

Name enables you to specify the name of the rectangle.

Slotting

Enable makes available the slotting options. With these options, you can specify the settings for creating a rectangle with slots.
Environment variable: slotEnableRect

Note: (ICADVM20.1 Virtuoso RF Solution Only) The *Slotting* options are not supported for curved shapes.

© 2020 All Rights Reserved.

Layer enables you to specify an LPP where you want slots to be created in ROD objects. Because no real holes are created in ROD objects in the design, the *Slot Layer* information is used later by manufacturers to punch holes in the metal layer where the slot LPP occurs. You do not need to specify a slot LPP for polygons because real holes are created in the polygon metal layer in the design.

Specify width, area enables you to specify the width and area parameters for a slot. The length of the slot is automatically calculated.
Environment variables: slotWidth and slotSpecifyArea

Dimensions

Width: The spacing between slots is taken into consideration to derive the `minSpacing` value if a one-dimensional spacing table exists in the technology file. Both interslot spacing and the length of the slot are taken into consideration to derive the `minSpacing` value if a two-dimensional spacing table exists in the technology file.

Length: The spacing between rows is taken into consideration to derive the `minSpacing` value if a 1-D spacing table exists in the technology file. Both row spacing and the width of the slot are taken into consideration to derive the `minSpacing` value if a two-dimensional spacing table exists in the technology file.

Area: Displays the area of the slot.

Spacing

Column: Represents the width of the metal between slots. The metal spacing between the slots is governed by the `minWidth` and `maxWidth` rules.

Row: Represents the width of the metal between slots, which is governed by the `minWidth` and `maxWidth` rules.

Edge: Represents the width of the metal between a slot and the shape edge. Here too the metal spacing is governed by the `minWidth` and `maxWidth` rules.

For information about the process rules associated with these fields, see Calculating Dimensions Associated with Slots.

Related Topics

Creating Rectangles

leHiCreateRect

© 2020 All Rights Reserved.

Create Row Form

The *Create Row* form has following sections:

- Definition
- Templates
- Create Method

Definition

You use the *Definition* section of the form to input basic information for rows, including the site definition name, user specified or software generated names, and orientations.

Base Name allows you to enter a unique name for each row object you create.
Environment variable: rowBaseName

Row Orientation allows you to set the row orientation using radio buttons for Horizontal and Vertical.
Environment variable: rowFirstUnitOrient

Starting Site Orientation allows you to set the orientation of the site and the first row created.
Environment variable: rowUnitOrient

Options

Flip Every Other Row toggle button allows you to flip every other row in the site.
Environment variable: rowFlipUnit

Flip The First Row toggle button allows you to flip the first row in the site.
Environment variable: rowFlipFirstUnit

Abut toggle button allows you to abut a specified number of rows. Abut overrides Channel Width.
Environment variable: rowAbutUnit

rows allows you to specify how many rows you want to abut. The integer must be positive.
Environment variable: rowAbutNumberOfUnits

Abut The First Row toggle button allows you to abut the first and second rows.
Environment variable: rowAbutFirstUnit

© 2020 All Rights Reserved.

Channel Width allows you to specify a floating-point number that is the space between the rows you create. If you specify the Abut rows, the Channel Width is overridden. Environment variable: rowChannelWidth

Filter By Site Definition allows you to filter for core site definitions or pad site definitions. You can see both by retaining *ALL* in the cyclic field. If you select *ALL*, all site definitions in the technology library are displayed. The form opens with the last value retained in the field.

Site Definition Name displays all site definition names in the list box. When you have multiple site definitions use the arrows to scroll up and down through the list. You can select one site name at a time.

Technology Library is a read-only field displaying the library that you have selected in the *Site Definition Name* field.

Site Information is a read-only field displaying the type of site definition (core or pad) that is the current site in the *Site Definition Name* field. This information contains the type (core or pad), the width and height of the site definition, and three types of symmetry. This information is specified in the technology library site definitions. The symmetry can be *SymmetricInX*, *SymmetricInY*, and *SymmetricInR90*.

Templates

The *Templates* section allows you to place rails and/or components automatically into the rows.

Type allows you to select *Rail*, *Allowed Component*, or *Both* in the *Pre-Defined* pane of the form. You move them to the *Selected* pane using the arrows. Once they are in the *Selected* pane, they are available to place in the cellview. You can then add the components individually selected in the *Selected* pane.

Create Method

There are two create methods. You use the *Assisted* mode to create rows that will not take into account any areas or boundaries. You use the *Area Fill* mode to create rows that use an area or boundary to determine the number and placement of the rows.

Assisted toggle button method creates the specified number of rows and sites.

Number of Rows sets the number of rows you create, when you use the assisted create method. The number should be greater than or equal to one. When you place your pointer in the cellview window, the wire frame of the row objects appears in the cellview. The total sites of each row should be multiple of the Number of Sites you enter. If you

© 2020 All Rights Reserved.

want the maximum number of rows the number entered in this field should be one.
Environment variable: rowNumberOfRows

Number of Sites field lets you set the number of sites in a row. The minimum value is an integer >=1. The default is one.
Environment variable: rowNumberOfSites

Area Fill toggle button method uses an area or boundary to determine the number and placement of the rows.

Rectangular allows you to create an area in which as many sites possible will fit.

Rectilinear allows you to create a polygonal shape with rows that follow the edges of the polygon.

Partition cyclic field allows you to create an area consisting of pre-defined areas. The choices are prBoundary, snapBoundary or clusterBoundary. The field can display more than one clusterBoundary. The snapBoundary will always snap to a region that allows numeric multiple row(s) with the current settings.

Offset allows you to place asymmetrical rows in the boundary areas. The offsets are similar to blockages in that they do not allow the rows to come closer to the boundary or cluster edge than the user units defined in the Offset field.

Top sets the distance from the top of the boundary that the rows will not enter.
Environment variable: rowPartitionTopOffset

Bottom sets the distance from the bottom of the boundary that the rows will not enter.
Environment variable: rowPartitionBottomOffset

Left sets the distance from the left of the boundary that the rows will not enter.
Environment variable: rowPartitionLeftOffset

Right sets the distance from the right of the boundary that the rows will not enter.
Environment variable: rowPartitionRightOffset

Related Topics

Creating Rows

leHiCreateRow

© 2020 All Rights Reserved.

Create Slot Form

Note: The *Create Slot* command is available only for rectangles and paths. For information about creating slots while creating rectangles and paths, see Creating Rectangles and Creating Paths.

Layer enables you to specify an LPP where you want slots to be created in ROD objects. Because no real holes are created in ROD objects in the design, the *Slot Layer* information is used later by manufacturers to punch holes in the metal layer where the slot LPP occurs. You do not need to specify a slot LPP for polygons because real holes are created in the polygon metal layer in the design.

Shape Type indicates the type of the resultant shape that is created after slots are added to it.
Environment variable: slotShapeMode

> **Preserve** retains the original shape type after slots are added to the shape. A polygon shape remains a polygon after slots are added to it. Similarly, a ROD object remains a ROD object after slots are added to it.
>
> **Convert to Polygon** converts a shape to a polygon after slots are added to the shape.
>
> **Convert to ROD** converts a shape to a ROD object after slots are added to the shape.

Specify width, area enables you to specify the width and area parameters for a slot in the *Dimensions* field. The length of the slot is automatically calculated.
Environment variables: slotWidth and slotSpecifyArea

Dimensions

For information about the process rules associated with these fields, see Calculating Dimensions Associated with Slots.

© 2020 All Rights Reserved.

The following figure illustrates the width and length of slots.

Width/Length enables you to specify the width and length parameters for a slot. The area of the slot is automatically calculated.

Spacing

© 2020
All Rights Reserved.

The following figure illustrates the types of spacings between slots.

Column specifies the distance between the slots in a row.
Environment variable: slotInterSlotSpacing

Row specifies the distance between two slot rows.
Environment variable: slotRowSpacing

Edge specifies the distance between the slot edge and the shape edge.
Environment variable: slotEdgeToSlotOffset

The following figure illustrates the edge-to-slot distance.

© 2020 All Rights Reserved.

Stagger specifies the spacing between the edges of slots in consecutive rows. Only alternate rows are staggered.
Environment variable: slotStaggerOffset

© 2020 All Rights Reserved.

The following figures illustrate staggering of slots in paths:

Note: Staggering of slots is not supported in rectangles.

Calculating Dimensions Associated with Slots

The dimensions associated with slots are calculated based on the rules specified below:

- *Spacing* (*Column*): Represents the width of the metal between slots. The metal spacing between the slots is governed by the `minWidth` and `maxWidth` rules.
- *Spacing (Row)*: Represents the width of the metal between slots, which is governed by the `minWidth` and `maxWidth` rules.

© 2020 All Rights Reserved.

- *Spacing (Edge)* offset: Represents the width of the metal between a slot and the shape edge. Here too the metal spacing is governed by the `minWidth` and `maxWidth` rules.
- *Stagger* offset: Represents the metal spacing around slots. The metal spacing for this optional field is governed only by the `maxWidth` rule.

Based on the values you specify in the fields listed above, the values in the *Width* and *Length* fields are calculated automatically. The *Width* and *Length* values comply with the `minSpacing` rule. These values are calculated as follows:

- *Width*: The spacing between slots is taken into consideration to derive the `minSpacing` value if a 1-D spacing table exists in the technology file. Both interslot spacing and the length of the slot are taken into consideration to derive the `minSpacing` value if a 2-D spacing table exists in the technology file.

Width of the Slot = Derived `minSpacing` Value

Inter Slot Spacing = Width of the Metal

Length of the Slot = Parallel Run Length of the Metal

- *Length*: The spacing between rows is taken into consideration to derive the `minSpacing` value if a 1-D spacing table exists in the technology file. Both row spacing

© 2020 All Rights Reserved.

and the width of the slot are taken into consideration to derive the `minSpacing` value if a 2-D spacing table exists in the technology file.

Related Topics

Creating Slots

Editing Slots

leHiCreateSlot

© 2020 All Rights Reserved.

Create Snap Boundary Form

Create Method chooses which method is used to create the snap boundary object.
Environment variable: `createBoundaryMethod`

> **From Selected Set** creates the snap boundary using the rectangular bounding box of the selected set of objects.
>
> **Rectangle** creates the Snap boundary using two points.

Create as ROD Object creates the boundary as a ROD object.

> **Name** lets you assign a name to the new boundary in the current layout cellview. The name must be unique in the cellview. If you do not edit this field, the system assigns a unique name consisting of the prefix `boundary`. The prefix is followed by a number. For example, the first ROD boundary would be named `boundary0`, the second, `boundary1`, and so on.

Related Topics

Creating Snap Boundary

leHiCreateSnapBoundary

© 2020 All Rights Reserved.

Create Via Form

Mode controls the type of via created.
Environment variable: viaMode

Single creates a single via or an array of vias. See Single Mode Options.

Stack places the vias belonging to adjacent layers on top of each other. See Stack Mode Options.

Auto automatically places a via at the intersection of two paths. See the Auto Mode Options.

Fast Edit lets you perform interactive via editing. See the Fast Edit Mode Options.

Note: In Auto and Fast Edit modes, the Create Via form does not open automatically. To open the Create Via from, press `F3`.

Options opens the Create Via Options Form.

Compute From

- In *Single* and *Stack* modes, you can choose a **Compute From**:
  - *Rows* and *Columns:* Via(s) are created according to the number of rows and columns
  - *Width* and *Height*: Via(s) are created according to the width and height of the layer
  - *Shape(s):* Via(s) are created inside the overlap of the shapes on the canvas
  - *Drawn Area*: Vias(s) are created inside the area, rectangular or polygonal, that you draw on the canvas
- In the *Auto* mode, when you click the canvas, the via (s) is placed in the overlap of two shapes. By enabling *From Area*, rectangular or polygonal, you can define a region in which the layout editor will place the via(s) in all overlapping regions.
  Environment variable: viaAutoViaCutArrayDirection

  All the remaining form field values in both the *Single* and *Stack* modes are calculated automatically by the *Create Via* command when you click the overlapping region of two intersecting shapes. In the *Single* mode, the command calculates the rows, columns, and enclosures, by using the information specified in the form. In the *Stack* mode, the command calculates the rows and columns of the top via and the spacing, sizes, and enclosures of all the vias in the stack, such that all the vias in the stack fit in the area of overlap between the two intersecting shapes.

© 2020 All Rights Reserved.

Environment variable: viaComputeFromMode

Constraint Group enables you to select the constraint group.

Options

Net Name applies a net name to the via you are placing in the cellview. In case of a via stack, it associates the specified net with the entire via stack. If the net constraint exists, the constraint is applied to the via stack. If the specified net does not exist, a new net is created. If no net is specified, there is no association with the via or via stack.

Create as ROD Object creates the via as a ROD object.

> **Name** assigns a unique name to the new via in the current cellview if the *Create as ROD Object* check box is selected. This unique name consists of the prefix `via` followed by an integer. For example, the first ROD via is named `via0`, the second `via1`, and so on. You can edit the default name, if required, but ensure that the new name is unique in the cellview.

The *Create Via* form displays different options based on the mode selected for creating a via:

- Single Mode Options
- Stack Mode Options
- Auto Mode Options
- Fast Edit Mode Options

Single Mode Options

Via Definition list displays the valid viaDefs and valid via variants in a tree-like structure from the active constraint group (set in the *Via* list in the Layout Editor Options Form). By default, the viaDefs are listed in the same order as specified in the `validVias` constraint. If the `validVias` constraint does not exist in the active constraint group, the viaDefs are ordered by mask number. You can further update the order by using the appropriate *Sort* option in the Via Definition Selection widget.

If any filtering options are set up by using the *Filter* widget, the *Via Definition* list displays only the viaDefs or via variants that satisfy the filtering options. Each node in the tree represents a standard or custom viaDef. A node with a plus sign (+) represents a parent standard or custom viaDef with child viaVariants. For more information about viaVariants, see Technology File Via Definitions and Via Specifications in the *Virtuoso Technology Data ASCII Files Reference*.

© 2020 All Rights Reserved.

If a parent viaDef is not a valid via in the active constraint group, the viaDef is grayed out and cannot be selected. Choosing a viaDef or via variant from the list selects the via from the effective technology file.

The **Filter** button displays the Via Definition Selection widget, which contains the options for filtering and sorting viaDefs and displaying the filtered viaDefs in a list box.

Filter specifies an alphanumerical string to search for matching viaDefs from the `validVias` list. The wildcard characters, such as `*`, `$`, and `^` can be used in the search string. The search is case-sensitive. Only viaDefs matching the search string are displayed in the *Via Definition* list.

Sort By sorts the viaDefs displayed in the list box based on either viaDef names in alphanumerical order or the mask numbers of the bottom layer of the viaDef. Sorting is done in ascending order.

From and **To** specify the layer limits. viaDefs existing between these layers are displayed in the list box.

Searches for viaDefs by viaDef name or by the layers between which they are available are exclusive of each other. You can leave the *Filter* field blank and still search for viaDefs between two specific layers.

List Box displays the filtered and sorted viaDefs in a tree-like structure. The viaVariants are grouped under the respective parent standard or custom viaDef. The viaDef or viaVariant selected from the list box updates the *Via Definition* cyclic field as well as closes the Via Definition Selection widget.

Note: The search criteria are not retained when you close or hide the Create Via form.

© 2020 All Rights Reserved.

Consider the following example, where the search string is `M2_*`. All viaDefs with `M2_` at the beginning of the viaDef name and that use the layers between and inclusive of `Metal1` and `Metal8` are displayed in an alphabetical order in the list box.

The read-only field displays **Via Type / Technology Library** of the viaDef selected in the *Via Definition* cyclic field. The via type indicates the type of `viaDef` selected in the *Via Definition* cyclic field - `Standard` or `Custom`. If you select a via variant in the *Via Definition* cyclic field, the via type displays `Standard Via Variant` or `Custom Via Variant`, respectively. The technology library indicates the library to which the selected viaDef belongs.

The **Save Via Variant** button opens the Save Via Variant Form, where you can change some parameter values of a standard or custom viaDef and then save it as a via variant.

You can update the via even after it has been created to use the parent viaDef by using the button on the *Edit Via Properties* form or the *Property Editor* assistant. In the *Property Editor* assistant, the button is visible when you click the *ViaDef* value column.

Reset Parameters to enables you to reset the values in the form to the technology database values if you want to revert from the manually updated parameter values to the defaults. When you first open the *Create Via* form, the via parameter values are calculated based on the *Via Parameters Calculation Mode* setting in the *Create Via* section in the Layout Editor Options Form.

- *Minimum Rules then ViaDef*: All the via parameter fields, such as the number of cut rows and columns, cut sizes, spacing, and enclosures, are reset to the minimum rule or the viaDef default values, whichever is greater. The minimum rules are picked from the

© 2020 All Rights Reserved.

constraint group that you select from the *Via* list in the *Create Via* section in the Layout Editor Options Form. The calculated values are never less than the viaDef defaults. If minimum rules are not defined, viaDef defaults are used.

For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.1` is used as the enclosure value in the *Create Via* form. Similarly, if the `minSpacing` rule is `0.2` and the default cut spacing in viaDef is `0.18`, `0.2` is used as the cut spacing X and Y in the *Create Via* form.

- *Minimum Rules*: All the via parameter fields are reset to the minimum rule values. ViaDef defaults are not checked. Therefore, the calculated values can be less than the viaDef defaults.

  For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.05` is used as the enclosure value in the *Create Via* form. The viaDef default enclosure value is ignored.

- *Minimum Rule Enclosures*: Minimum rules are used to calculate only the enclosure field values. Values in the other fields are not reset.

- *ViaDef Defaults*: All the via parameter fields are reset to the viaDef defaults. Minimum rules are not checked.

- *ViaDef Defaults Enclosures*: ViaDef defaults are used to calculate only the enclosure field values. Values in the other fields are not reset.

  ViaDef Defaults and *ViaDef Default Enclosures* reset options are available only when *Compute Enclosures* is off. When *Compute Enclosures* is off, you can specify enclosure values less than the minimum rule values.

  Note: If you select a via variant from the *Via Definition* list, the *Reset Parameters to* list is not available.
  Environment variable: viaParamCalcMode

  The following table explains how the parameter values are reset if one of the *Reset Parameters to* option is selected. The table illustrates scenarios when *Compute Enclosures* is either on or off.

Compute Enclosures	Parameter Values	Reset Parameters to				
		Minimum Rules then ViaDef	**Minimum Rules**	**Minimum Rule Enclosures**	**ViaDef Defaults**	**ViaDef Default Enclosures**

© 2020 All Rights Reserved.

On	Enclosure Values	minimum rule and viaDef default	minimum rule	minimum rule	N/A	N/A
	Other Values	minimum rule and viaDef default	minimum rule	As is	N/A	N/A
Off	Enclosure Values	viaDef default	viaDef default	viaDef default	viaDef default	viaDef default
	Other Values	minimum rule and viaDef default	minimum rule	As is	viaDef default	As is

The *Create Via* form, in the *Single* mode, displays the following tabs: System Tab, User defined Tab, Cut pattern Tab, and Array pattern Tab. The type of viaDef you select - standard or custom - determines the tab that is available.

System Tab

This tab contains values for standard vias.

Justification sets the origin of a via. Cyclic field choices are: `lowerLeft`, `centerLeft`, `upperLeft`, `lowerCenter`, `centerCenter`, `upperCenter`, `lowerRight`, `centerRight`, `upperRight`, and `offset`.

> **X** sets the origin point of the via in the X direction when the cyclic field is set to `offset`. When columns are set to more than one and the cyclic field is set to a direction, X is grayed out and it displays the amount of the directional offset in the X direction.
>
> **Y** sets the origin point of the via in the Y direction when the cyclic field is set to `offset`. When rows are set to more than one and the cyclic field is set to a direction, Y is grayed out and it displays the amount of the directional offset in the Y direction.
>
> **Note:** For 1x1 via (one row and one column), all *Justification* values except for `offset` map to the same point. Therefore, the *Justification* field on the *Edit Via Properties* form displays only `centerCenter` and `offset`. For a via with one row and more than one columns, `lowerLeft`, `upperLeft`, and `centerLeft` map to the same point as do `lowerRight`, `upperRight`, and `centerRight`. Therefore, for such a via, the *Justification* field on the *Edit Via Properties* form displays `centerLeft`, `centerRight`, and `centerCenter`.

© 2020 All Rights Reserved.

Cut Section

Cut Class cyclic field displays cut class choices only if a `cutClasses` constraint has been defined for the cut layer of the selected viaDef in the `layerRules` section of the technology file. The following are examples of cut classes:

`Small` is a single cut via where both dimensions are short.

`Bar` is a large via that has one long and one short dimension. The shorter dimension is equal to the width or length of `Small`.

`Large` is a large via that has two long dimensions, equal to the longer dimension of `Bar`.

If you update the *Cut Class* selection, the values in the fields, such as *Rows*, *Columns*, *Width*, *Length*, *Row Spacing*, and *Column Spacing*, are dynamically updated.

X and **Y** set the width and length of the via cut, in user units (typically microns). If a *Cut Class* is selected, the cut *Width* and *Length* are automatically updated to the cut size values defined for that cut class in the `layerRules` section of the technology file.

Swap button swaps the values in the *Width* and *Length* fields.

Size X and **Y** set the number of rows or the number of columns of via cuts in a via array. You can specify a value up to `10000` in these fields. These fields do not seed the *Row* and *Columns* fields on the *Cut Pattern* tab. If you change the values in the *Rows* and *Columns* fields, the values in the other fields, such as *Width*, *Length*, *Row Spacing*, and *Column Spacing*, are dynamically updated.

Swap button swaps the values in the *Rows* and *Columns* fields. If the swapped values violate the minimum rules, the spacing and enclosure values are re-calculated based on the *Reset Parameters to* setting.

Spacing X sets the distance between the horizontal edge to the edge cuts in a via.

Spacing Y sets the distance between the vertical edge to the edge cuts in a via.

Swap button swaps the values in the *Row Spacing* and *Column Spacing* fields.

© 2020 All Rights Reserved.

Enclosures Section

Compute automatically calculates enclosures. This is on by default. You can deselect this option to manually specify the enclosure values.
Environment variable: viaCompuEnclosures

Show Enclosures/Hide Enclosures button toggles the display of the section in the form that sets the directional enclosure distances on Layer 1, the bottom routing layer, and Layer 2, the top routing layer.

Swap button swaps the enclosure values in the layer 1 and layer 2 fields. If the swapped values violate the minimum rules, the enclosure values are re-calculated based on the *Reset Parameters to* setting.

Layer 1 Left sets the horizontal distance on the left side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 1 Right sets the horizontal distance on the right side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 1 Top sets the vertical distance on the top side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 1 Bottom sets the vertical distance on the bottom side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Swap button swaps the values in the *Left/Right* and *Top/Bottom* fields of layer 1. If the swapped values violate the minimum rules, the enclosure values are re-calculated based on the *Reset Parameters to* setting.

Layer 1 Implant X sets the horizontal distance from the left and right cut edge to the implant enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 1 Implant Y sets the vertical distance from the top and bottom cut edge to the implant enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 2 Left sets the horizontal distance on the left side of the via from the cut edge to the enclosure edge on the layer designated layer 2 (top routing layer).

Layer 2 Right sets the horizontal distance on the right side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (top routing layer).

Layer 2 Top sets the vertical distance on the top side of the via from the cut edge to the enclosure edge on the layer designated layer 2 (top routing layer).

Layer 2 Bottom sets the vertical distance on the bottom side of the via from the cut edge to the enclosure edge on the layer designated layer 2 (top routing layer).

© 2020 All Rights Reserved.

Swap button swaps the values in the *Left/Right* and *Top/Bottom* fields of layer 2. If the swapped values violate the minimum rules, the enclosure values are re-calculated based on the *Reset Parameters to* setting.

Layer 2 Implant X sets the horizontal distance from the left and right cut edge to the implant enclosure edge on the layer designated layer 2 (top routing layer).

Layer 2 Implant Y sets the vertical distance from the top and bottom cut edge to the implant enclosure edge on the layer designated layer 2 (top routing layer).

Note: If you manually specify the enclosure values, they are cached as long as they obey the minimum rules. When you update other via parameters in the form, the manually specified enclosure values are not re-calculated unless they violate the minimum rules.

Purpose Section

This section appears in the *Create Via* form if you select a cdsVia from the *Via Definition* list.

Primary enables you to select the purpose for the metal layers. The purposes in this list are populated from the layer purposes defined in the `techLayerPurposePriortities` section of the technology file for the selected cdsVia metal layer.

Cut enables you to select the via layer purpose. By default, it displays the *As primary* value to indicate the same purpose as selected in the *Primary* list.

Other enables you to select the implant layer purpose. This list appears only if any implant layers are defined for the selected `cdsVia` device.

User defined Tab

This tab contains all user-defined via parameters and CDF parameters, which are not in the set of Cadence predefined via parameters.

In the case of cdsVia, this tab displays the Pcell parameters that exist on the via. You can change the parameters as required, but the *Create Via* form does not perform any checks for them.

Cut pattern Tab

This tab allows you to create cut patterns on via arrays.

Via Rows displays the number of rows in a via array. This is a read-only field that displays the number of rows specified in the *Rows* field on the *System* tab.
Environment variables: viaRows

© 2020 All Rights Reserved.

Via Columns displays the number of columns in a via array. This is a read-only field that displays the number of columns specified in the *Columns* field on the *System* tab.
Environment variables: viaColumns

Cut pattern Section

Pattern lets you select a predetermined pattern, such as Custom, Via Def, All On, All Off, Chess Board, H-V Stripes, H Stripes, V Stripes, Top, Bottom, Left, Right, U-L Corner, U-R Corner, L-L Corner, L-R Corner, and Cross.

Note: You can now define the default cut pattern for each standard via in the *standardViaDef* section of the DFII ASCII techfile. This default cut pattern is reflected on the *Cut pattern* tab of the *Create Via* form.

Repeat Pattern repeats the pattern specified in the *Pattern* field for the number of rows and columns specified in the *Rows* and *Columns* fields. If *Repeat Pattern* is not selected, the values in the *Rows* and *Columns* fields are the same as the values in the *Via Rows* and *Via Columns* fields, respectively.

Rows sets the number of rows of the repeated pattern. If *Repeat Pattern* is not selected, this field is disabled.

Columns sets the number of columns of the repeated pattern. If *Repeat Pattern* is not selected, the field is disabled.

© 2020 All Rights Reserved.

Depending on the number of rows and columns you specify in the *Rows* and *Columns* fields the row x column matrix is depicted by check boxes. You can select or deselect the check boxes to define the cut pattern that you want.

Note: If the number of rows and columns exceeds 10, then the cut pattern is not displayed on the *Cut pattern* tab of the *Create Via* form.

Array pattern Tab

This tab allows you to create array patterns on vias. This tab is enabled only if the selected viaDef in the *Via Definition* field is a cdsGenVia.

Array Spacing Section

Spacing X specifies the spacing to be applied between the arrays of cuts horizontally.

Spacing Y specifies the spacing to be applied between the arrays of cuts vertically.

Swap button swaps the values in the *Spacing X* and *Spacing Y* fields.

Pattern X specifies the number of cuts in the array horizontally.

Pattern Y specifies the number of cuts in the array vertically.

Note: For *Pattern X* and *Pattern Y* fields, you can provide multiple numbers separated by spaces. For example, if you specify 3 2 4 in the *Pattern X* field, then the via engine horizontally creates 3 cuts (separated by a “small” cut spacing), then a “large” *Spacing X*, then 2 cuts (separated by a “small” cut spacing), then a “large” spacing X, then 4 cuts (separate by a “small” cut spacing). This pattern is repeated or truncated depending on the number of cut columns specified on the *System* tab. For a 15x15 via, with 3 2 4 specified in the *Pattern X* field, the via engine will create a 3 2 4 3 2 1 via.

Swap button swaps the values in the *Pattern X* and *Pattern Y* fields.

Rotate turns the via 90 degrees counterclockwise. You can also click the middle mouse button to rotate the via.

© 2020 All Rights Reserved.

Sideways mirrors the via or via stack along the X axis. You can also press `Ctrl` and click the middle mouse button to mirror the via or via stack.

Upside Down mirrors the via or via stack along the Y axis. You can also press `Shift` and click the middle mouse button to mirror the via or via stack.

Stack Mode Options

Start Layer lets you select one of the two layers between which you want to place the via stack. The layer selected in the *End Layer* list does not appear in the *Start Layer* list.

End Layer lets you select one of the two layers between which you want to place the via stack. The layer selected in the *Start Layer* list does not appear in the *End Layer* list.

Justification sets the origin of a via. Cyclic field choices are: `lowerLeft`, `centerLeft`, `upperLeft`, `lowerCenter`, `centerCenter`, `upperCenter`, `lowerRight`, `centerRight`, `upperRight`, and `offset`.

Top Rows and **Top Columns** sets the number of rows and columns for the top via in the via stack. The default number of rows and columns is 1. Rows and columns cannot be specified for the individual vias in the via stack; they apply to the complete via stack. The command calculates the rows and columns of the vias below the top via in such a way so that all vias in the stack coincide.
Environment variables: viaStackRows and viaStackColumns

If the via stack contains custom vias, the default rows and columns of the custom vias are used for creating the via stack. You cannot modify the rows and columns of custom vias by using this form.

Via Definitions lets you select viaDefs for each layer pair between the specified *Start Layer* and *End Layer*. For each layer pair, the drop-down list displays the available viaDefs. The viaDefs are populated from the `validVias` constraint in the current constraint group in the technology file.

The **Cut Class** cyclic field displays cut class choices only if a `cutClasses` constraint has been defined for the cut layer of the selected viaDef in the `layerRules` section of the

© 2020 All Rights Reserved.

technology file. If the `cutclassGroups` are defined in the technology file, you can choose a cut class group for all the layer pairs, as shown in the figure below.

```
cutClasses(
;( layerName     )
;(    (cutClassName
;( ----------------------------------------

 (V1
     (V1              'cutClassGroups ("NORMAL")
     (V1BAR           'cutClassGroups ("BAR")
     (V1LRG           'cutClassGroups ("LARGE")
 )
 (V2
     (V2              'cutClassGroups ("NORMAL")
     (V2BAR           'cutClassGroups ("BAR")
     (V2LRG            cutClassGroups ("LARGE")
 )
```

Rotate turns the via 90 degrees counterclockwise. You can also click the middle mouse button to rotate the via.

Sideways mirrors the via or via stack along the X axis. You can also press `Ctrl` and click the middle mouse button to mirror the via or via stack.

Upside Down mirrors the via or via stack along the Y axis. You can also press `Shift` and click the middle mouse button to mirror the via or via stack.

Auto Mode Options

When you select this option, the form displays the following additional fields:

© 2020 All Rights Reserved.

Use Entire Overlap uses the entire overlap to create the via. If this option is not selected, an intersection between the user area and the overlap box is used to create the via. The figure below displays the effect of this option.

Environment variable: viaUseEntireOverlap

© 2020
All Rights Reserved.

Cut Class is a cyclic field that displays cut class choices only if a `cutClasses` constraint has been defined for the cut layer of the selected viaDef in the `layerRules` section of the technology file.

Via Alignment specifies the alignment of the via using the alignment choices: *auto*, *top-left*, *top-center*, *top-right*, *center-left*, *center-center*, *center-right*, *bottom-left*, *bottom-center*, and *bottom-right*. The default is *auto*.
Environment variable: viaAutoViaAlignment

Cut Class Orientation specifies the preferred orientation for the cut class as one of the following: *auto*, *horizontal*, *vertical,* or *uniform in stack*.
Environment variable: viaCutClassOrientation

Cut BBox Direction specifies the preferred direction for the cut box as one of the following: *auto*, *horizontal*, or *vertical*.
Environment variable: viaAutoViaCutArrayDirection

Min Number of Cuts specifies the minimum number of cuts to be automatically created for any overlapping area between two shapes.
Environment variable: viaAutoViaMinNumCuts

Max Stack Depth when selected, the overlap with the same number of transitions or less than the specified number are searched. You can specify the value of the maximum stack depth. If you specify `1` as the value in the `Max Stack Depth` field, and the layers under `Metal11` are `Metal2` and `Metal5`, only the overlap `Metal1`-`Metal2` is retained. The other overlaps, `Metal1`-`Metal5` (stack depth=4) and `Metal2`-`Metal5` (stack depth=3) are dropped.
Environment variable: viaAutoViaEnableMaxStackDepth and viaAutoViaMaxStackDepth

Connect Shapes on The Same Net connects the shapes that are on the same nets.

More Options

Fill Polygonal Overlap option is used when the overlap is polygonal. You can choose to either fill in all the tiles of the polygon overlap, *Fully*, or only the tile under the pointer, *Partially*.

Align Cuts specifies the alignment of cuts between different tiles of a polygonal overlap. When this option is enabled, cuts between different tiles of a polygonal overlap are aligned. When this option is disabled, the polygonal overlap is decomposed into a set of largest tiles and vias are created independently in each tile, which may result in cuts not being aligned between different tiles.
Environment variable: viaAlignCuts

© 2020 All Rights Reserved.

Prevent DRC Violations On The Overlap prevents the creation of a via if it introduces DRC violations on the overlap. If this option is not selected, the via is created even if it introduces DRC violations on the overlap.
Environment variable: viaPreventDRCOnOverlap

The example below depicts auto via creation using the *Prevent DRC Violations On The Overlap* option.

Auto via on Metal2 - Metal3 overlap creates a rectShapeDir violation on Metal2 (polygonal shape on Metal2 forbidden in this process node)

When the *Prevent DRC Violations On The Overlap* option is not selected, the auto via is created even though it introduces a DRC violation. This is depicted in the figure below.

Auto via is created

© 2020
All Rights Reserved.

When the *Prevent DRC Violations On The Overlap* option is selected, the auto via is not created as it introduces a DRC violation. A warning message is displayed in the CIW.

The following buttons are available in the *Single* and *Stack* modes.

Prevent DRC Violations With Neighbors prevents the creation of a via if it introduces DRC violations with the neighboring shapes. If this option is not selected, the via is created even if it introduces DRC violations with the neighboring shapes.
Environment variable: viaPreventDRCWithNeighbors

Prevent Via Larger Than Overlap prevents the creation of a via if it is larger than the overlap.

© 2020
All Rights Reserved.

Show Preview shows a preview of the via that will be created.

When you move the pointer over the overlap, the preview of the via is displayed

Environment variable: viaAutoViaPreviewShow

Show Hints displays the *Auto Via Preview Bindkeys* hint box on the canvas.

Environment variable: viaAutoViaHints

Rows +/-: Use `Ctrl` + mouse wheel up/wheel down to add or remove via rows.

Column +/-: Use `Shift` + mouse wheel up/wheel down to add or remove via columns.

Swap Rows and Columns: Use mouse wheel up/wheel down to swap rows and columns.

Lock/Unlock Via: Use the `Spacebar` key to lock/unlock the via displayed in the preview, as shown in scenario 1 below. When no via is displayed in the preview and the pointer is over an existing via, this feature locks/unlocks the via under the pointer, as shown in scenario 2 below.

© 2020 All Rights Reserved.

Fast Edit Mode Options

When you select this option, the form displays the following additional fields:

From Selected Vias lets you use fast edit command from a set of selected vias.
Environment variable: viaFastEditFromSelectedVias

Cut Class is a cyclic field which displays cut class choices only if a `cutClasses` constraint has been defined for the cut layer of the selected viaDef in the `layerRules` section of the technology file.

Via Alignment specifies the alignment of vias using the alignment choices: *auto*, *top-left*, *top-center*, *top-right*, *center-left*, *center-center*, *center-right*, *bottom-left*, *bottom-center*, and *bottom-right*.

Cut Class Orientation specifies the preferred orientation for the cut class as one of the following: *auto*, *horizontal*, *vertical,* or *uniform in stack*.

Cut BBox Direction specifies the preferred direction for the cut box as one of the following: *auto*, *horizontal*, or *vertical*.
Environment variable: viaAutoViaCutArrayDirection

Minimum Number of Cuts specifies the minimum number of cuts to be automatically created for any overlapping area between two shapes.
Environment variable: viaAutoViaMinNumCuts

More Options

Fill Polygonal Overlap option is used when the overlap is polygonal. You can choose to either fill in all the tiles of the polygon overlap, *Fully*, or only the tile under the pointer, *Partially*.

Prevent DRC Violations On The Overlap prevents the creation of a via if it introduces DRC violations on the overlap. If this option is not selected, the via is created even if it introduces DRC violations on the overlap.
Environment variable: viaPreventDRCOnOverlap

Prevent DRC Violations With Neighbors prevents the creation of a via if it introduces DRC violations with the neighboring shapes. If this option is not selected, the via is created even if it introduces DRC violations with the neighboring shapes.
Environment variable: viaPreventDRCWithNeighbors

© 2020 All Rights Reserved.

Prevent Via Larger Than Overlap prevents the creation of a via if it is larger than the overlap.

Show Preview shows a preview of the via(s) that will be edited or re-computed.

When you move the pointer over the overlap, the preview of the via is displayed

In Auto and Fast Edit via creation modes, a label *PREVIEW* is displayed close to the pointer. This ensures that you are aware that the displayed via is a preview and not a real via existing in the layout design. When you click the overlap area, the via is created and the *PREVIEW* label disappears.

If the layout editor is not able to display the preview of the via in a reasonable time, the label PREVIEW TOO COMPLEX is displayed.

© 2020 All Rights Reserved.

Note: In the Fast Edit mode, the vias are updated instead of being re-created, wherever possible.

Video

For details on auto via preview and fast edit via creation mode, see the video *Auto Via Preview & Smart Auto Via*.

© 2020 All Rights Reserved.

Show Hints displays the *Auto Via Preview Bindkeys* hint box on the canvas.

Related Topics

Creating Vias

Create Via Options Form

leHiCreateVia

© 2020 All Rights Reserved.

Create Via Options Form

For specifying the via options, there are two forms available:

- Create Via Options, used to specify via creation options during the `Create Via` command. See Create Via Form, Creating or Updating a Via Variant, and Setting Up Via Creation Options for more information.
- Select Via Options, used to specify via definitions during the `Create Wire` command. This form excludes the *Override Variant Parameters*, *Extend Enclosure Beyond Overlap*, and *Fill Overlap With Enclosure* options. See Changing Layers and Selecting Vias for more information.

Override Variant Parameters lets you to update a via variant and overwrite the existing via parameter values. If this check box is selected, it allows the via engine to update a via variant.
Environment variable: autoViaOverrideVariantParameters

Extend Enclosure Beyond Overlap lets you to extend the via enclosures beyond the overlap region between two intersecting shapes. The shape layer that exists beyond the overlap region helps to accommodate the extended enclosures.
Environment variable: viaExtendEnclosureBeyondOverlap

Create In Route lets you to create a via in a new route. If this option is selected, every new via is created in a new route. This check box is selected by default.
Environment variable: viaCreateInRoute

Fill Overlap With Enclosure lets you to extend via enclosure such that it covers the complete overlap area between the two intersecting shapes, that is, the bounding box of the via enclosure coincides with the bounding box of the overlap region.
Environment variable: viaFillEnclosureOverlap

Create Vias Using Top Two Layers Only lets you to create a via in a new route. If this option is selected, every new via is created in a new route.

Consider Blockages does not create any via in case a blockage exists between the top and bottom metal of the current overlap. Only via and cut layer blockages are considered, metal layer blockages are not considered.
Environment variable: viaCheckBlockageTypes

Valid Purposes lets you to specify a list of valid purposes for via creation in automated modes: auto, fast edit, compute from shape: single, compute from shape: stack. The via will not be created even if one purpose of the overlap shapes does not belong to the list of valid purposes specified in this field.
Environment variable: viaEnableValidPurposes, viaValidPurposes

© 2020 All Rights Reserved.

Ignore Purposes lets you specify the list of purposes that will be ignored for via creation in automated modes: auto, fast edit, compute from shape: single, compute from shape: stack. Environment variable: viaIgnorePurposes

Cut Spacing Method lets you to select whether the cut spacing is spread or is using the minimum spacing rule constraints. You can select *distribute* or *minimum* from the drop-down list.

Environment variable: viaCutSpacingMethod

© 2020 All Rights Reserved.

Transitions lets you to select the transitions when creating a via. You can select all the transitions by selecting the *All* check box. You can also select the transitions for which you want the via to be created.

Via Definitions lets you to define the default viaDefs/via variants to be used in the *Single*, *Stack*, and *Auto* via creation modes. The selected values appear as the default values in the *Single* and *Stack* modes when you open the Create Via Form.

Cut Class lets you to set up the default cut class for each viaDef/via variant to be used in the *Single*, *Stack*, and *Auto* via creation modes. The list displays the cut class choices if a `cutClasses` constraint is defined for the cut layer of the selected viaDef in the `layerRules` section of the technology file.

> The *Cut Class* value selected in the *Create Via Options* form appears as the default in the *Cut Class* list on the Create Via Form for the respective viaDef/via variant selected from the *Via Definitions* list in the Create Via Form.
>
> Some examples of the cut class values include *Small*, *Bar*, and *Large*. If you set the *Cut Class* value as *Auto*, the *Cut Class* value is computed automatically by the via engine; in the *Single* mode, you can explicitly specify the *Cut Class*. For automatic *Cut Class* computation, the via engine chooses the *Cut Class* that maximizes the number of effective cuts in the overlap region.

User Min Cut Spacing lets you to specify the X and Y cut spacing for a transition. Select the check box to enable the X and Y fields for the transition for which you want to specify the min cut spacing values.

© 2020 All Rights Reserved.

Custom Show Purpose Form

Match any of the following specifies that only those layers for which at least one of the specified criteria match must be displayed in the *Layers* panel.

Match all of the following specifies that only those layers for which all specified criteria match must be displayed in the *Layers* panel.

Add criteria adds a new row that can be used to define a new criteria.

Related Topics

Show

© 2020 All Rights Reserved.

Define Net/Pin Connections Form

Net lets you enter the name of a net you want to select. You can type the following in the *Net* Field

- A net name

  The list box highlights the net and scrolls the list to make the net name visible.

- A part of a net name (*net*, for example)

  The list box highlights all the nets with *net* as an element in the name (*net10, net12, net23, net30, net31*, for example) and scrolls the list to make the first example visible.

- A regular SKILL expression using part of a net name (**bias*, for example)

  The list box highlights all the nets with the string *bias* (*Nbias, NMbias, Pbias*, for example) and scrolls the list to make the first example visible.

Search finds and highlights in the list box the names of nets you type in the *Net* field.

The list box displays the names of all the nets in the design. To select a net from which to define externally connected pins, you can click the net you want to select or click and hold down the `Control` key to select more than one net. When you select a net in the list box, the software selects the net in the layout and displays the flight lines for the net.

Related Topics

Defining Must Connect Pins

Defining Pseudo Parallel Connected Net

Defining Strongly Connected Pins

Defining Weakly Connected Pins

© 2020 All Rights Reserved.

Delete Form

Net Interconnect deletes any selected paths, instances that are vias, and all non-pin shapes on the net. Pins, component connections (instTerms), and the net is not deleted.

Selected deletes any selected objects.

> **Segment of a Path** deletes any selected objects in partial select mode only, and can be used to delete sections of a path while retaining path connectivity.

Preserve Non-Physical-Only Instances deletes the physical-only instances (exist only in the block domain) among the instances selected. Instances and parents of instances that exist in both block and module domains are not deleted. A message appears in the CIW to convey that the non-physical-only instance have not been deleted. For more information about creating physical-only instances, see the Create Instance Form.
Environment variable: preventDeletionOfLogicalInstance

Chain Mode enables you to delete device chains.

> **All:** Deletes all instances in a chain.
>
> **Selected:** Deletes only selected instances.
>
> **Selected Plus Left:** Deletes the instance selected and all instances to the left of the selected instance in the chain.
>
> **Selected Plus Right:** Deletes the instance selected and all instances to the right of the selected instance in the chain.

Related Topics

Deleting Objects

leHiDelete

© 2020 All Rights Reserved.

Delete All Markers Form

Severity defines which level of markers the system deletes in the list box and in your design.

all selects all marker types for deletion.

none deselects all marker types.

fatal error deletes all markers of the type fatal error.

critical error deletes all markers of the type critical error.

signed off critical error deletes all markers of the type signed off critical error.

error deletes all markers with error messages.

signed off error deletes all markers of the type signed off error.

warning deletes all markers with warning messages.

acknowledged warning deletes all markers of the type acknowledged warning.

info deletes all markers of the type info.

annotation deletes all markers of the type annotation.

Environment variable: markerDeleteSeverities

Search Scope defines the scope of the deletion.

cellview deletes markers from the current cellview only.

hierarchy starting from top cellview deletes markers throughout the entire hierarchy.

hierarchy starting from current cellview deletes markers throughout the hierarchy starting with the current cellview.

Environment variable: markerDeleteScope

Options enables you to specify whether the cellview is to be opened in edit or read mode.

Cellview Access Mode specifies the mode for opening a cellview.
Environment variable: markerDeleteAccessModeIsEdit

read deletes all the markers and re-opens the cellviews in edit mode with no dialog box.

© 2020 All Rights Reserved.

edit deletes all the markers after opening the cellview in edit mode. In this mode, you can choose to select the *Confirm Access Mode Change* check box and specify the cellviews to be opened in read mode.

Confirm Access Mode Change opens the *Delete Marker* dialog box for you to confirm whether you want to open the selected cellview in edit mode. Markers are not deleted in cellviews that stay in read mode. This check box is enabled only if *Cellview Access Mode* to set to *edit*.
Environment variable: markerDeleteConfirmModeChange

Sources specifies the category of markers that are to be deleted in terms of the program that generated them.

all deletes all markers that match the severity and scope criteria.

select signifies that a list of marker owners will be computed and only the sources that have been selected will be deleted.

Environment variables: markerDeleteSourcesMode, markerDeleteSources

Related Topics

Deleting All Markers

© 2020 All Rights Reserved.

Delete Configuration Form

Configuration displays a list of configuration from which to choose the configuration to be deleted.

Related Topics

Displaying Object Information

© 2020 All Rights Reserved.

Delete Marker Form

Options enable you to specify whether the cellview is to be opened in edit or read mode.

Cellview Access Mode specifies the mode for opening a cellview.
Environment variable: markerDeleteAccessModeIsEdit

read deletes the selected markers and re-opens the cellviews in edit mode with no dialog box.

write deletes all the markers after opening the cellview in edit mode. In this mode, you can choose to select the *Confirm Access Mode Change* check box and specify the cellviews to be opened in read mode.

Confirm Access Mode Change opens the *Delete Marker* dialog box for you to confirm whether you want to open the selected cellview in edit mode. Markers are not deleted in cellviews that stay in read mode. This check box is enabled only if *Cellview Access Mode* to set to *edit*.
Environment variable: markerDeleteConfirmModeChange

Related Topics

Deleting Individual Markers

© 2020 All Rights Reserved.

Descend Form

View Name lets you choose a different view of the master cell to open with the *Descend* command.

Open in lets you select to open the cellview in a *new tab*, the *current tab*, or a *new window*.

Related Topics

Descending into a Cellview

leDescend

© 2020 All Rights Reserved.

Descend Options Form

Prompt For View Name By default, *Descend* descends into the same view name as the current cellview. If this field is on, the (view name) *Descend* form appears when you click an instance. You can choose a different view to open. If there is only one view of the master cell, *Descend* does not prompt you for a view name.

Open in lets you select to open the cellview in a *new tab*, the *current tab*, or a *new window*.

Related Topics

Descending into a Cellview

© 2020 All Rights Reserved.

Discard Edits Form

Yes deletes all of the edits you made since the last time you saved.

No cancels the command without deleting your edits.

Related Topics

Discarding All Edits

© 2020 All Rights Reserved.

Display Options Form

The *Display Options* form controls the appearance of objects and the behavior of commands in this cellview.

The form comprises the following sections:

- Display Controls
- Grid Controls
- Resolution
- Snap Modes
- Display Abstraction Controls
- Dimming

Display Controls

Open to Stop Level sets the display level to the Display Levels Stop level.
Environment variable: openToStopLevel

Nets shows flight lines between objects on the same net. If your design contains many nets, your screen may turn white, causing the `instTerms` on top of the instance to not be seen. To see the `instTerms`, turn on *Instance Pins* and the flight lines will not display, allowing the instTerms to be seen.
Environment variable: netsOn

Axes displays the cellview X and Y axes.
Environment variable: drawAxesOn

Access Edges shows routing edges of pins.
Environment variable: accessEdgesOn

Instance Origins shows origin of the instances.

Instance Pins shows pins in instances.
Environment variable: drawInstancePins

Note: When the *Instance Pins* option is selected, even if the virtual hierarchy is opaque you can view the virtual pins.

Instance Pin Names shows pin names in instances.
Environment variable: drawInstancePinNames

Instance Orientation shows orientation of the instances.

© 2020 All Rights Reserved.

Array Icons shows outlines of array cells when *Display Levels* suppresses cell details.
Environment variable: iconsOn

Environment variable: displayInstanceOrientation

EIP Surround (edit in place) displays the surrounding design when you edit a cell in place.
Environment variable: drawSurroundingOn

Label Origins marks the origins of labels with diamond markers.
Environment variable: textJustificationOn

Pin Names shows terminal names of pins that have pin name text displays or labels.
Environment variable: displayPinNames

Use True BBox when on, displays the instance master bounding box. When off, displays the cellview bounding box, which can cause large designs to open faster because masters are opened down to the display stop level only.
Environment variable: useTrueBBox

Dot Pins displays via origins with diamond markers when you set *Display Levels* to show instance content.
Environment variable: dotsOn

Cross Cursor displays the cursor as dot shaped or as dotted-line cross shaped.
Environment variable: cursorShape

Net Expressions displays the net expression instead of the terminal name of a pin. When there are net expressions in instances, the terminal name is displayed, not the net expression, even when *Net Expressions* is set on.
Environment variable: leDispNetExpr

Row Name displays the names of placed rows, which are displayed on the rows in the horizontal or vertical direction depending upon the rotation of the rows.
Environment variable: displayRowName

Stretch Handles displays the handles on a Relative Object Design parameterized cell (Pcell) that indicate that the Pcell can be stretched. A *stretch handle* is a named set of coordinates assigned to a specific parameter of the Pcell. Stretch handles look like small diamonds. For information about stretchable Pcells, see Stretchable Parameterized Cells in the *Virtuoso Relative Object Design User Guide.*
Environment variable: displayStretchHandles

Row Site displays the position of row sites, which are represented by a cross marker on the row.
Environment variable: displayRowSite

© 2020 All Rights Reserved.

Via Shapes displays via attributes. When Via Shapes is set to off (the default), vias are displayed as the bounding box of each via. When set to on via attributes are graphically displayed, resulting in detailed images. For large designs, setting via shapes to on will increase the amount of time it takes to render a design.
Environment variable: displayViaShapes

Dynamic Highlight marks the edge, object, or the point that is selected if you enter a point selection. In addition, it also marks the label of the object. When *Dynamic Highlight* is on and your cellview contains a large number of objects, pointer motion may slow down. You can increase the pointer motion speed by turning this off.
Environment variable: dynamicHilightOn

Transparent Group enables editing a group without the need for EIP in that group. This setting is applicable only for the current window. By default, it is off.
Environment variable: transparentGroup

Dragged Object Ghost enables you to view the ghost of the original object when it is being moved or stretched. This option is useful while working with instances. The example below shows an instance being moved. In the first figure, the *Dragged Object Ghost* check box is selected, whereas in the second, the check box is not selected.

Details of the Original Object (Ghost) Visible

Only Outline of the Original Object (Ghost) Visible

The check box is selected by default.
Environment variable: ghostOfDraggedObjects

Group Name Only lets you display only the group name for a figure group when *Display Levels*, *Start* and *Stop*, are set to `0`. The figure group should be set to *Visible* in the Palette Assistant.
Environment variable: displayFigGroupName

Selection Hint enables displaying selection information when an object is selected by clicking on it. The information is displayed next to the pointer and it lets the you know which object was selected. You can select an object from several overlapping objects.

Note: If you want to display selection hint for a wire or via stack, ensure that selection

© 2020 All Rights Reserved.

options for Spine and Via Stack are enabled.

Traversed instance BBox draws a bounding box around a traversed instance. A traversed instance has size greater than the *Size* set in the *Filter* section. If this check box is selected and the *Stop* display level is greater than zero, the bounding box of the instance displays even if you continue to zoom out of the design, as shown below.

If you zoom out when this check box is not selected, the instance becomes invisible and can be detected or selected only by hovering the pointer over the instance.

Note: Select the check box before zooming out. The bounding box does not appear if you select the check box after the instance becomes invisible.

This check box is off by default.
Environment variable: drawTraversedInstBbox

Traversed Instance Name displays the name of the instance for a traversed instance.

© 2020 All Rights Reserved.

This option is enabled only when the *Traversed instance BBox* option is selected.
Environment variable: drawTraversedInstName

Scroll Percent enables you to specify or change the scroll percentage for a window. The default value is `25`, which means that if you scroll the middle mouse button up or down, the design display area will zoom in or zoom out, respectively, by 25 percent.
Environment variable: scrollPercent

Instance Drawing Mode determines the type of boundary displayed in the top-level cellview. By default, the *BBox* displays. You can set *prBoundary* or *snapBoundary* to display.
Environment variable: instanceDrawingMode

Path Display determines whether you see path borders and centerlines, path borders only, or path centerlines only.
Environment variable: pathCL

Snap Pattern Display determines whether you see the boundary, the tracks, the fins, and periods (*Full*) or the boundary and the tracks (*Tracks*), or just the boundary (*Boundary*) for snap patterns, or periods (*Periods*), or periods and tracks (*Periods and Tracks)* for width spacing patterns.
Environment variable: snapPatternDrawingStyle

Set LPP Visibility enables you to control the visibility of an LPP based on its visibility and validity. By default, this option is set to *Do not check validity*. With this setting, only the *Visibility* attribute of the LPP is used to determine the visibility of a shape on that LPP; the validity of the LPP is ignored. If you set this option to *Check validity*, only the shapes on valid and visible LPPs are displayed in the design window.
Environment variable: lppVisibilityMode

Show Name Of, when *Display Levels* is set to show only instance outlines, sets whether the instance name (for example: `I1`) or the master cell name or both appear on each instance.
Environment variable: instLabel

Array Display

Full displays all instances in the array.

Border displays only the instances around the outside edge of the array.

Source displays only the instance at the origin of the array.
Environment variable: arrayDisplay

Display Levels sets the first (*Start*) and last (*Stop*) levels in the design hierarchy that can be seen in detail. The hierarchy levels are numbered 0 to 32. The top cellview is level

© 2020 All Rights Reserved.

0, instances inside of it are level 1, and so forth.
Environment variables: startLevel, stopLevel

Zoom/Pan Controls

Dynamic Zoom activates dynamic zoom (*View - Dynamic Zoom*) to objects on the design canvas. The default for this option is on.
Environment variable: autoZoomPan

Mode specifies the dynamic zoom setting to be used. You can choose from:

Pan To Selected: Whenever an object, or a set of objects, is selected/cross-selected in the Navigator assistant, the view will pan to the selected objects without changing the zoom factor (equivalent to *View - Pan*).

Zoom To Selected: Whenever an object, or a set of objects, is selected/cross-selected in the Navigator assistant, the view will zoom to the selected object(s). The zoom scale applied is based on the value specified in the *Zoom Scale (%)* field.

Zoom To Fit: Fits the complete design in the view. The zoom scale applied is based on the value specified in the *Zoom Scale (%)* field.

Environment variable: autoZoomMode

Zoom Scale (%) lets you set the scale of zoom in percentage. The default is `90%`.
Environment variable: autoZoomScale

Grid Controls

Type controls the grid display.

none turns off the grid display.
Environment variable: drawGridOn

dots displays a dot for each grid point.

lines makes a grid of lines, like a graph.
Environment variable: drawDottedGridOn

Dim Major Dots controls the dimming of dots of the major grid. When this option is not selected, the major grid dots are visible in bold face. This is a window-specific option and is available only when *Type* is set to *dots*.
Environment variable: dotStyleMajorGrid

Minor Spacing and **Major Spacing** set the number of user units between the visible grid. Minor grid points are white, major grid points are green by default.
Environment variables: gridSpacing and gridMultiple

© 2020 All Rights Reserved.

X Snap Spacing and **Y Snap Spacing** set the distance at which the pointer can snap between grid points along the X axis and the Y axis. This is your drawing grid.
Environment variables: xSnapSpacing and ySnapSpacing

Resolution

This section lets you change the display resolution. The resolution determines how much detail of a design is displayed in the cellview.

Resolution slider controls the level of detail of the design displayed in the cellview. The resolution options are *Low*, *Medium*, *High*, and *Very High*. The default value is *Medium*.
Environment variable: displayResolution and filterSizeForResolution

Note: Using the *Resolution* slider, the performance gets enhanced to display the design with a *Very High* resolution.

Highest Resolution for P&R Objects check box is available when the resolution is set to *High* or *Very High*. This check box enables to use a filter size for objects that is bigger or equal to the filter size used for shapes. When this option is not selected, smallest objects are not drawn, which improves the redraw time.
Environment variable: displayPRHighestResolution

Snap Modes

The snap mode locks your pointer to the grid while drawing or editing.

Create controls how line segments snap to the grid as you create objects.
Environment variable: snapMode

Edit controls how line segments of objects snap to the grid as you move or copy them and how edges or corners move as you stretch them.
Environment variable: segSnapMode

Display Abstraction Controls

Merge Shapes makes the display more readable by hiding all the extra edges in the design.
Environment variable: mergeEnable

Scope lets you select the hierarchies for which the *Enable Merge* command applies. The choices are *Pcells*, *Except current*, and *All*.

Pcells: All Pcells in the hierarchy including the current cellview

Except current: All hierarchies except the current cellview

All: All hierarchies including the current cellview

© 2020 All Rights Reserved.

Environment variable: mergeScope

Trim Shapes hides the layer that is cut using the `Trim` command.
Environment variable: trimEnable

Slot Shapes lets you create holes in wires. You overlap the shape to slot with another shape in slot purpose that is displayed as a hole in the shape.
Environment variable: slotEnable

Dimming

Dim Virtual Hierarchy lets you change the luminosity of the virtual hierarchy.
Environment variable: vhDimming

Note: When the *Dim Virtual Hierarchy* option is selected, the *Enable Dimming* and *Dim Selected Object Content* options are selected automatically and cannot be changed. Also, the *Automatic Dimming* option is disabled.

Enable Dimming lets you change the color luminosity of objects in the cellview.
Environment variable: dimmingOn

Scope

none - no dimming is applied. This is the default. Even if you change the Dim Intensity from zero percent to one hundred percent, the objects remain undimmed. The only exception is that the background cellview dims if its Display field is set to dimmed.

outside - dims everything but the instance, figure group, or any object you are editing in place. The dimmed objects represent the objects that are non-editable at the current edit level. With the outside setting on,

- If a Pcell exists at the edit level and is visible at the edit level, its contents are not dimmed. For example, in the following hierarchy, both `PcellA` and `PcellB` are be visible at *Display Stop Level* 32. However, at the `Top` edit level when the dimming

© 2020 All Rights Reserved.

scope is *outside*, the contents of `PcellA` areundimmed and those of `PcellB` are dimmed.

- If the *Via Shapes* option is on, vias belonging to the edit level are not dimmed.
- If the *Instance Pins* option is on, instance pins belonging to edit level-1 are not dimmed.

all - the design is completely dimmed as soon as you select this option. If you select an object and Dim Intensity is set to hundred percent, only the selection box is displayed.

eipSurround - dims all the objects surrounding the EIP hierarchy.
Environment variable: dimmingScope

Dim Grid Lines dims the grid lines on the canvas.
Environment variable: dimGridLines

Automatic Dimming dims the design automatically either when you select an object or when the design contains highlight sets or probes. Only selected, highlighted, or probed objects remain undimmed. This option allows dimming even when the dimming Scope is set to none. This option is not available when Scope is set to all.
Environment variable: dimmingAutoEnabled

Dim Intensity lets you set the luminosity of dimmed objects. When set to `0`, nothing is dimmed. When set to `100`, dimmed objects are not visible. The default is 50%. *Dim Intensity* has no effect if *Scope* is set to *none*, except when the *Display* field of the background cellview is set to *dimmed*. In that case, only the background cellview is dimmed.
Environment variable: dimmingIntensity

Dim Selected Object Content dims the selected object.

True Color Selection only displays the selected objects in their original colors when other objects are dimmed. The selection highlight is not displayed. This option is disabled when *Scope* is set to *none* and *Automatic Dimming* is off.
Environment variable: displaySchDashText

© 2020 All Rights Reserved.

Cellview specifies that you want to store, load, or delete the display settings to or from the cellview.

Library specifies that you want to store, load, or delete the display settings to or from the library of the edit cellview.

Tech Library specifies that you want to store, load, or delete the display settings to or from the technology library of the edit cellview.

File specifies the file to which you want to store or from which you want to load the settings. When you select the file radio button the path field is editable so that you can enter the correct directory where you want to save the file. If you do not remember the name of the directory use the *Browse* button.
Environment variable: displayOptionsFile

Save To saves the current settings to either the cellview, library of the cellview, technology library of the cellview, or a specified file. If you are saving to a file, the settings from both the Layout Editor Options and Display Options forms are saved.

Load From sets the current settings to either the cellview, library of the cellview, the technology library of the cellview, or a specified file. If you saved to a file, the settings for both the Layout Editor Options and Display Options forms are loaded.

Delete From deletes the display settings that were saved to either the cellview, library of the cellview, or technology library of the cellview.

The selections do not take effect until you click *Apply* on the *Display Options* form.

Related Topics

Setting Display Options

leHiEditDisplayOptions

© 2020 All Rights Reserved.

Dynamic Display Form

Show Info Balloon when on, information balloons will appear displaying information about an object or shape when the pointer is moved over the existing object. The type of information can be customized and is object dependent.
Environment variable: balloonOn

Measurement Display On displays information about objects and shapes dynamically, during the creation or editing of objects. The type of information can be customized and is object dependent.
Environment variable: measurementDisplayOn

Highlight Aligned Edges On highlights edges of the shapes aligned to the pointer in horizontal and vertical directions.
Environment variable: highlightAlignedEdges

The *Dynamic Display* form comprises the following tabs:

- Setup Tab
- Info Balloon Tab
- Dynamic Measurement Tab
- Highlight Aligned Edges Tab

Setup Tab

Object Display Setting allows you to expand categories to display shape and object types. You can display and customize information about existing objects or dynamically, during the creation or editing of objects.

Note: You must first turn on the controls for information balloons or textual display of measurements.

VLS is an expandable tree, which displays the categories *Shapes* and *Objects* in which data types are organized.

Shapes indicates the types of shapes that you can customize for information balloon and dynamic measurement display.

Objects indicates the types of objects that you can customize for information balloon and dynamic measurement display.

Configuration allows you to save and load the customized object display shapes and objects.

© 2020 All Rights Reserved.

Save opens the *Save Configuration* form. See Save Configuration Form for more information.

Default saves the selected configuration as the default configuration.

Delete opens the removes the selected configuration.

Contents See Parameter Types for Shapes and Objects.

Select

All allows you to select all available parameters for each shape and object rather than selecting parameters individually. By default, all parameters are selected for the *default* configuration.

None allows you to deselect all available parameters for each shape and object rather than deselecting parameters individually.

Info Balloon Tab

Pause Time indicates the amount of time before the information balloon will appear when the pointer is over an object. Valid values are between 0 and 10.
Environment variable: balloonPauseTime

Fade Time indicates the amount of time (in seconds) for the information balloons to fade away after the balloon appears. Valid values are between .001 and 100.
Environment variable: balloonFadeTime

Transparency indicates the transparency of the balloon information box. When set to 0, the box is completely transparent. When set to 100, objects behind the box are not visible. The default is 50.
Environment variable: balloonTransparency

Dynamic Measurement Tab

Text Height (pixels) indicates the height of the text displayed during dynamic measurement. Valid values are any integer between `4` and `50`.
Environment variable: measurementTextHeight

Color indicates the color of the text displayed during dynamic measurement.
Environment variable: dynamicMeasurementTextColor

© 2020 All Rights Reserved.

Highlight Aligned Edges Tab

Width indicates width of the highlight that is displayed on the edges of the shapes aligned. You can choose from: *Thin*, *Medium*, or *Thick*.
Environment variable: highlightAlignedEdgesWidth

Color indicates the color of the highlight displayed on the edges of aligned shapes.
Environment variable: highlightAlignedEdgesColor

© 2020 All Rights Reserved.

Parameter Types for Shapes and Objects

Note: Distance is the absolute distance as expressed in a straight line between the last point entered and the current pointer location. Delta X and Y is the difference between the coordinates of a reference point and the current pointer location on the x or y axis. This reference point is the point entered to create part of an object, or the selection point used to begin a move or a copy.

For *Custom SKILL Function*, see Using a Custom SKILL Function.

Shape	**Available Parameters**	**Dynamic Measurement Specific Parameters**
Rectangle	X-Length, Y-Length, Area, Perimeter, XL Status, LPP, Label, NetName (if applicable), LPP-Color	Delta XY
Polygon	Area, X-Length, Y-Length, Perimeter, XL Status, LPP, Label, NetName (if applicable), LPP-Color	Delta XY, Distance
Path	XL Status, Area, Width, Length, LPP, Label, NetName (if applicable), LPP-Color The length of a path is the total centerline path length and does not include beginning or ending extensions. The info balloon also displays the area information.	Delta XY, Distance
MPP	XL Status, Area, Width, Length, LPP, Label, NetName (if applicable), LPP-Color The length of a multipart path is the total centerline path length and does not include beginning or ending extensions or subparts.	Delta XY, Distance
Pathseg	XL Status, Area, Width, Length, LPP, Label, NetName (if applicable), LPP-Color The info balloon also displays the area and total length information.	Delta XY, Distance
Circle	Area, Perimeter, Radius, XL Status, LPP, Label, LPP-Color	Delta XY
Ellipse	Area, Perimeter, XL Status, LPP, Label, LPP-Color	Delta XY

© 2020 All Rights Reserved.

Shape	Available Parameters	Dynamic Measurement Specific Parameters
Donut	Area, Perimeter, Radius, XL Status, LPP, Label, LPP-Color	Delta XY
Label	LPP, Label (text of label)	Delta XY
TextDisplay	LPP, Label (text of label)	
Instance	XL Status, Label (lib/cell/view names)	Delta XY
Blockage	Area, X-Length, Y-Length, Perimeter, LPP, Label (any label attached to the blockage), LPP-Color, Distance	Delta XY, Distance
Row	X-Length, Y-Length, Area, Row Sites, Label (row base name)	Delta XY
Boundary	Area, Perimeter, Label (boundary name), LPP **Note:** For the PR boundary of a soft block, ‘Area’ displays the soft block utilization information.	Delta XY, Distance
Via	XL Status, Label (viaDefName), NetName, LPP-Color	Delta XY
Custom Placement	Area, Perimeter	Delta XY
Marker	Message (type tool, short message, severity)	
Pin	XL Status, LPP, Label (pin name), NetName, LPP-Color	Delta XY
Modgen	Label (any label attached to the modgen)	Delta XY
Ruler	Length, Points	Delta XY
Row Region	Area, Perimeter, Label, Custom SKILL Function	

© 2020 All Rights Reserved.

Shape	Available Parameters	Dynamic Measurement Specific Parameters
Virtual	Area, Label, Stop Level, Type, Placement Status. (These parameters are selected by default) Perimeter, Distance, Delta XY, Custom SKILL Function (These parameters are not selected by default) These fields are also displayed in the info balloon. If Delta XY is selected for dynamic measurement, it is also displayed in the info balloon.	Delta XY

Related Topics

Displaying Object Information

leHiEditObjectInfo

© 2020 All Rights Reserved.

Dynamic Selection Assistant Options Form

Use this form to specify how selected objects are displayed in the assistant and the behavior used when opening a cellview from the assistant's context menu.

Display Controls enables you to specify how the selected objects are displayed in the assistant.

> **Automatically zoom to selected** automatically zooms to the selected objects in the design display area. If this option is selected, you can zoom shapes in the hierarchy by selecting these shapes in the Dynamic Selection assistant.
>
> **Note:** You can also zoom to the corresponding shape on the canvas if you double-click the shape in the Dynamic Selection assistant.
>
> **Automatically expand hierarchy** automatically expands the hierarchical instances, groups, and ROD objects shown in the assistant. The hierarchy is expanded down to the display level set in the Display Options Form. If the display level is set to `0`, the hierarchy cannot be expanded. The assistant does not necessarily list all the objects contained in a particular hierarchical object, only those that were under the pointer when the display was frozen.
>
> **Hide non-selectable objects** removes from the assistant any objects that cannot be selected.
>
> **Reverse LPP order** displays the layer-purpose pairs (LPPs) for the selected objects from bottom to top. By default, the LPPs of selected objects are displayed from top to bottom.
>
> **Flat view** displays the flattened version of the hierarchy. If this option is selected, *LPP*, *Data*, and *Hierarchy* columns are displayed in the Dynamic Selection assistant.

Open mode specifies whether the cellview is opened for *edit* or *read*.

Open in specifies whether the cellview is opened in a *new tab*, the *current tab*, or a *new window*.

© 2020 All Rights Reserved.

Dynamic Selection Context-Sensitive Menu

The *Dynamic Selection* assistant's context-sensitive menu contains the following items.

Command Name	Description
Open Instance (<mode>/ <tab>)	Opens the master cellview of the instance currently under the mouse pointer with the behavior specified in the command name. For example, *Open (edit/new tab)* opens the cellview in edit mode in a new tab; *Open (read/current tab)* opens the cellview in read mode in the current tab. You specify the default behavior for this command in the Dynamic Selection Assistant Options Form. **Note:** This command is available only if the mouse pointer is over an instance. To open a different view of the current cell, use the *Open Cellview* command instead.
Open Cellview	Displays the Open form where you can specify ■ Which view of the current cell you want to open ■ Whether that view is opened in edit or read mode ■ Whether it is opened in a new tab, the current tab, or in a new window. **Note:** This command is available only if the mouse pointer is over an instance.
Edit In Place	Enables you to edit in place the currently selected cellview in the layout window. **Note:** This command is available only if the mouse pointer is over an instance, a group, or an object contained in a hierarchy.

© 2020 All Rights Reserved.

Expand All Below	Expands the hierarchy to display the hierarchical objects under the current node. **Note:** This command is available only for hierarchical objects. The hierarchy is expanded down to the display level set in the Display Options form; so if, for example, the display level is set to 0, the hierarchy cannot be expanded. The assistant does not necessarily list all the objects contained in a particular hierarchical object, only those that were under the pointer when the display was frozen.
Collapse All Below	Collapses the hierarchy to hide the hierarchical objects under the currently selected node. **Note:** This command is available only for hierarchical objects.
Options	Opens the Dynamic Selection Assistant Options Form, where you specify: ■ How selected objects are displayed in the assistant ■ The default open behavior
Freeze DSA/UnFreeze DSA	Freezes/Unfreezes the display

Related Topics

Dynamic Selection

© 2020 All Rights Reserved.

Edit Properties Form

Next highlights the next object in the group and resets the form to show that object's properties.

Previous highlights the previous object in the group and resets the form to show that object's properties.

Apply updates the properties of the object.

Common lets you edit properties common to a group of selected objects. The Common functionality does not support ROD objects.

Layer displays the layer of a common property if it is the same for all objects. If the layer is *not* the same for all objects, *AS IS* appears in the field. You can search for specific layer-purpose pair using the search filter in the *Layer* list box.

Note: If the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form, only the LPPs selected in the *Layers* panel of the Palette are visible in the *Layer* list box.

Left, Right, Width, Bottom, Top, and **Height** display the value of a common property if it is the same for all objects. If the value of a common property is not the same for all objects, the words *AS IS* appears in the fields.

Attribute Tab

The *Attribute* tab represents the characteristics of the object. The available attributes vary, depending on the type of object. If the object is a ROD multipart path (MPP), the *Subpart* field appears. See Attribute Types for Objects for more information on available attributes.

Connectivity Tab

The *Connectivity* tab displays routing and net information about selected pins and shapes on a net. Only pins display *I/O Type* or *Access Direction*.

Net Name displays the name of the net to which the pin is connected. This field can be used to rename or delete the net.

Note: When you edit the net name of an object (via or pathseg) that is a route element, the *Net Name* field appears as the *Route Net Name* field. This indicates that editing the net of an object in a route changes the net of the entire route.

© 2020 All Rights Reserved.

Terminal Name sets the name of the terminal associated with this pin. The terminal name should always be the same as the net name.

Net Expression assigns a net expression of the terminal listed in the *Terminal Name* field.

Property sets the override property name to the net expression.

Default defines the net to be used if no override property is defined in the hierarchy above this point (in the schematic view). Unless a different signal name is entered, the terminal name is used.

I/O Type assigns a property used by routers to identify the direction of the signal into or out of this cellview. The signal can be input, output, inputOutput (bidirectional), switch (carries data either in or out, but not simultaneously), or jumper (passes data through this cellview).

Net Criticality sets a weighting factor that determines the routing priority for this net for the Cadence place-and-route tools.

Access Direction assigns a property used to identify the part of the pin to which the routers can connect routing. Applies only to rectangle pins. The selections are: top, left, bottom, right, any or none.

Parameter Tab

The *Parameter* tab displays any parameters defined for a parameterized cell. or the selected instance, such as fluid guard ring. For information about the parameters displayed for a fluid guard ring, refer to the Edit Instance Properties Form for a Fluid Guard Ring section.

Property tab sets the properties of the object and lets you add your own properties to an object.

Add opens the Add Property form, which lets you add a new cellview property.

Delete deletes a selected property created with *Add*.

Modify opens the Modify Property form, which lets you change the definition for any property created with *Add*.

ROD Tab

The *ROD* tab displays the ROD properties of the object.

© 2020 All Rights Reserved.

Edit Properties Options: Virtuoso Photonics Solution (ICADVM20.1 Photonics Only)

Note: You will need the `Virtuoso_Photonics_Option` license to use the functionality discussed in this section. For information on obtaining the required license, contact your local Cadence representative.

The following additional options are displayed for a photonics pin.

Attribute Tab

Width lets you specify the width of the facet of the waveguide that connects to this pin.

Angle lets you specify the angle of the facet of the waveguide at the intersection of the pin.

Radius lets you specify the radius of the curve of the waveguide center at the point it meets the facet.

Connectivity Tab

Signal Type lets you select the signal type of the net of a photonics pin. For a photonics pin, the values for this option are *optical*, *singleModeOptical*, or *multiModeOptical*.

Related Topics

About Editing Properties

leHiEditProp

© 2020 All Rights Reserved.

Add Property Form

Lets you add your own properties to the currently selected object.

Name specifies the name to assign to this property. This name will appear on the Edit Properties or the Edit Cellview Properties form.

Type specifies the value type for the property. This can be one of the following:

- *String* (any text, displayed in a cyclic field)
- *Int* (integer)
- *Float* (floating-point number)
- *Time* (date and time)
- *Boolean* (on or off)
- *ILList* (list of IL values)

Value shows either a text entry field, a Boolean button, or a cyclic field, depending on the *Type* setting.

Choices sets the possible values that appear in the cyclic field for a string property. One of the values listed here must match the default shown in *Value*.

Minimum and **Maximum** display the minimum and maximum values for integer, floating-point number, and time properties. These ranges appear in parentheses after the property name in the Edit Properties Form or the Edit Cellview Properties Form.

Related Topics

Defining a New Cellview Property

© 2020 All Rights Reserved.

Modify Property Form

Note: You can modify and delete only those properties you added using the Add Property Form.

Name displays the name of the property.

Type controls the value type of the property.

- *String* (any text, displayed in a cyclic field)
- *Int* (integer)
- *Float* (floating-point number)
- *Time* (date and time)
- *Boolean* (on or off)
- *ILList* (list of IL values)

Value sets the default value for this property. This line can show a text entry field, a Boolean button, or a cyclic field, depending on the *Type* setting.

Choices sets the possible values that appear in the cyclic field for a string property. One of the values listed here must match the default shown in *Value*.

Minimum and **Maximum** specify the minimum and maximum values for integer, floating-point number, and time properties. These ranges appear in parentheses after the property name in the Edit Properties Form or the Edit Cellview Properties Form.

Related Topics

Modifying a Property

© 2020 All Rights Reserved.

ROD Property Form

The ROD property form displays the ROD name, handle, and alignment information of the object or instance.

ROD Name displays the name of the ROD object.

Handle displays the system handle and user handle information. A handle is an attribute of, or item of information about, a ROD object, such as the coordinates of a point on the bounding box around an object, the width of the bounding box of an object, or the resistance of an object.

> **System handle** displays handle names assigned to the object. When you choose a different handle from the cyclic field, the *Value* field updates to reflect the current handle.
>
> **Value** is a non-editable field displaying the location of the system handle displayed in the *System handle* field. These values are predefined reference points.
>
> **User handle** displays handle names assigned by the user to the object. When you choose a different handle from the cyclic field, the *Value* field updates to reflect the current handle. If a user handle is not assigned, the fields are empty.
>
> **Value** displays the current location of the system handle displayed in the *User handle* field. You can change these values by typing new coordinates in this field.

Alignment displays the alignment information about two ROD objects. You can edit these fields if the objects have alignment values assigned to them, otherwise the fields are empty. When looking at the alignment information for two objects, the selected object is *Align object*. The separation is the distance from *Reference object* (unselected) to *Align object* (selected).

> **Reference object** displays the name of the object assigned to be the reference object of *Align object*. If there is more than one object assigned to be *Reference object*, you can choose that object from the cyclic field.
>
> **Align object** displays the name of the object aligned to *Reference object*. This field is not editable.
>
> **Reference handle** displays the name of the handle *Align object* is aligned to.
>
> **Align handle** displays the name of the handle *Reference object* is aligned to.
>
> **X separation** displays the distance in the X direction from either the reference point handle or the reference point on *Reference object* to the alignment point handle on *Align object*. The value can be positive or negative or a Cadence SKILL language expression that evaluates to a positive or negative number.

© 2020 All Rights Reserved.

Y separation displays the distance in the Y direction from either the reference point handle or the reference point on *Reference object* to the alignment point handle on *Align object*. The value can be positive or negative or a SKILL expression that evaluates to a positive or negative number.

For complete information on setting ROD handles and alignment, see the *Virtuoso Relative Object Design User Guide*.

© 2020 All Rights Reserved.

Edit Cellview Properties Form

Attribute represents the characteristics of the object. None of these fields can be changed. See Attribute Types for Objects.

Library displays the name of the library containing this cellview.

Cell displays the name of this cell.

View displays the name of this view.

bBox displays the coordinates of the invisible box surrounding the design in the cellview.

Type displays the view type that corresponds to the view name for this cellview. View types are defined in your technology file.

DBU per micron displays the number of database units (DBU) per user unit (usually microns). Database units and user units are defined in the `.cdsenv` file or by the library manager.

Mode shows whether this cellview is opened in edit mode (`Edit`) or read-only mode (`Read`).

Save Needed shows whether you made changes to this cellview that need to be saved to disk.

CellType displays the various cellType attribute values. You can select one from the list to set the cellType of the current cellview. For valid cellType values, see **Cell Type Valid Values**.

Symmetry Type defines the orientation of this cell. The valid values include `no`, `X`, `Y`, `XY`, and `any`.

SitePattern

Site Definition cyclic field displays all the `siteDefs` defined in the technology file. Select a site definition from the list to set it for the current cellview. For more information about site definitions, see Site Definitions.

Origin displays the position associated with the sitePattern.

Rotation lists the possible rotation values that can be associated with the sitePattern.

© 2020 All Rights Reserved.

Tip

You can alternatively set the sitePattern for a cellview using SKILL in the CIW. For example, to set the sitePattern for a design with site definition as `Core`, type the following in the CIW:

```
geGetEditCellView~>sitePattern = list(list("Core" list(0.0 0.0) "R0"))
```

Property shows the properties for this cellview. Most properties correspond to settings in the Display Options form. Other Cadence applications such as DRC or Extract can also add properties.

Add opens the Open Property form, which lets you add a new cellview property.

Delete deletes a selected property created with *Add*.

Modify opens the Modify Property form, which lets you change the definition for any property created with *Add*.

Common is not used when you edit cellview properties. This button is used when you edit properties of objects inside this cellview with the *Edit – Basic – Properties* command.

Next and **Previous** (at the top of the form) are not used when you edit cellview properties but when you edit properties of objects inside this cellview with the *Edit – Basic – Properties* command.

Connectivity is not used when you edit cellview properties. This button is used when you edit properties of objects inside this cellview with the *Edit – Basic – Properties* command.

Parameter is not used when you edit cellview properties. This button is used when you edit properties of objects inside this cellview with the *Edit – Basic – Properties* command.

ROD is not used when you edit cellview properties. This button is used when you edit properties of ROD objects inside this cellview with the *Edit – Basic – Properties* command.

Related Topics

About Editing Properties

© 2020 All Rights Reserved.

Edit Cut Pattern Form

Note: Edit Cut Pattern is applicable for standard and custom vias.

Quick Edit allows you to edit the cut pattern in conjunction with the *Patterns* cyclic field to select a predetermined pattern. The form displays a maximum of ten rows and ten columns when you select this option. You can create a cut pattern with rows and columns that are fewer than the rows and columns of the array, since the selected pattern repeats across the entire array, starting from the lower left corner of the array and repeating upwards and to the right. The possible patterns are shown below.

Wall Paper allows you to edit the edit the cut spaces cut by cut or use the Patterns cyclic field to select a predetermined pattern. To edit cut by cut click the square to turn off the cut. This leaves a space in the cuts when you create the via. The form displays a maximum of ten rows and ten columns when you select this option. You can create a cut pattern with rows and columns that are fewer than the rows and columns of the array, since the selected pattern repeats across the entire array, starting from the lower left corner of the array and repeating upwards and to the right.

Paper Size allows you add or delete rows or columns to the cut pattern when *Wall Paper* is selected. The form displays a maximum of ten rows and ten columns when you select this option.

- The left arrow deletes a column with each click.
- The right arrow adds a column with each click.
- The up arrow adds a row with each click.
- The down arrow deletes a row with each click.

Manual Edit displays all of the rows and columns. You can edit the cut spaces cut by cut or use the *Patterns* cyclic field to select a predetermined pattern like those shown below. To edit

© 2020 All Rights Reserved.

cut by cut, click the square to turn off the cut. This leaves a space in the cuts when you create the via.

All On

All Off

Top

Bottom

Left

Right

© 2020 All Rights Reserved.

H Stripes (Horizontal Stripes)

V Stripes (Vertical Stripes)

Chess Board

Cross

U-R Corner (Upper Right Corner)

U-L Corner (Upper Left Corner)

© 2020 All Rights Reserved.

L-L Corner (Lower Left Corner)

L-R Corner (Lower Right Corner)

Patterns lets you select a predetermined pattern, such as All On, All Off, Top, Bottom, Left, Right, H Stripes, V Stripes, Chess Board, U-L Corner, U-R Corner, L-L Corner, L-R Corner, Cross, Farm, and As Is.

Related Topics

Editing Cut Patterns

© 2020 All Rights Reserved.

Edit Via Array Pattern Form

Note: Edit Via Array Pattern is applicable only for a cdsGenVia.

Spacing X specifies the spacing to be applied between the arrays of cuts horizontally.

Spacing Y specifies the spacing to be applied between the arrays of cuts vertically.

Swap button swaps the values in the *Spacing X* and *Spacing Y* fields.

Pattern X specifies the number of cuts in the array horizontally.

Pattern Y specifies the number of cuts in the array vertically.

For *Pattern X* and *Pattern Y* fields, you can provide multiple numbers separate by spaces. For example, if you specify 3 2 4 in the *Pattern X* field, then the via engine horizontally creates 3 cuts (separate by a “small” cut spacing), then a “large” *Spacing X*, then 2 cuts (separate by a “small” cut spacing), then a “large” spacing X, then 4 cuts (separate by a “small” cut spacing). This pattern is repeated or truncated depending on the number of cut columns specified on the *System* tab. For a 15x15 via, with 3 2 4 specified in the *Pattern X* field, the via engine will create a 3 2 4 3 2 1via.

Swap button swaps the values in the *Pattern X* and *Pattern Y* fields.

© 2020 All Rights Reserved.

Edit Instance Properties Form for a Fluid Guard Ring

For information about the Edit Instance Properties form for a Fluid Guard Ring, see Edit Instance Properties Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Attribute Types for Objects

This section describes the object attributes displayed in the Edit Properties form.

- Blockage Attributes
- Curved Shapes Attributes (ICADVM20.1 Photonics Only)
- Donut Attributes
- Dot Pin Attributes
- Ellipse or Circle Attributes
- Group Attributes
- Virtual Hierarchy / VH Clone Attributes
- Instance Attributes
- Mosaic Attributes
- Label Attributes
- Line Attributes
- Path Attributes
- Path Segment Attributes
- ROD Multipart Path - Master Path Fields Attributes
- Pin Name or Other Text Display Attributes
- Polygon Attributes
- Rectangle Pin Attributes
- Polygon Pin Attributes
- Row Attributes
- PR Boundary Attributes
- Snap Boundary Attributes
- Area Boundary Attributes
- Cluster Boundary Attributes
- Routing, Slot, Pin, or Feed Through Blockage Attributes

© 2020 All Rights Reserved.

- Placement Halo Blockage Attributes
- Via Attributes
- Via Stack Attributes
- Modgen Attributes

Blockage Attributes

Owner Information sets owner as *Cluster*, *Instance*, or *No Owner*.

Points sets the coordinates of each point of the blockage.

Area displays the total area of the blockage in user units.

Type displays the blockage type.

Layer selects the layer on which the blockage is enforced.

Material selects the material on which the blockage is enforced.

Use Effective Width enables the *Effective Width* field, when clicked.

Effective Width sets the value which is checked against the layer spacing rule or table spacing rules, if defined.

Spacing displays the row offset value in user units. *Spacing* attribute is disabled if the *Use Effective Width* attribute is selected.

Allow PGNet allows shapes with signal types power and ground to be placed within the blockage area.

Curved Shapes Attributes (ICADVM20.1 Photonics Only)

Curvy Line

Layer displays the layer on which the curvy line is drawn.

bBox displays the coordinates of the box surrounding the curvy line.

Points displays the coordinates of each point of the curvy line.

Begin Point displays the coordinates of the beginning point of the curvy line.

© 2020 All Rights Reserved.

End Point displays the coordinates of the end point of the curvy line.

Segment displays the coordinates of the segment of the curvy line.

Curvy Path

Layer displays the layer on which the curvy path is drawn.

bBox displays the coordinates of the box surrounding the curvy path.

Points displays the coordinates of each point of the curvy path.

Begin Point displays the coordinates of the beginning point of the curvy path.

End Point displays the coordinates of the end point of the curvy path.

Begin Facet Angle displays the beginning facet angle for the curvy path.

End Facet Angle displays the end facet angle for the curvy path.

Begin Facet Radius displays the beginning facet radius for the curvy path.

End Facet Radius displays the end facet radius for the curvy path.

Segments displays the coordinates of the segments of the curvy path.

Segments Count displays the number of segments.

Curvy Surface

Layer displays the layer on which the curvy surface is drawn.

bBox displays the coordinates of the box surrounding the curvy surface.

Points displays the coordinates of each point of the curvy surface.

Boundary displays the coordinates of the boundaries of segments the curvy surface.

Boundary Count displays the number of boundaries.

Holes displays the coordinates of the holes of segments of the curvy surface.

Number of holes displays the number of holes.

© 2020 All Rights Reserved.

Donut Attributes

Layer sets the layer on which the donut is drawn.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the donut.

Center: x and **y** set the X and Y coordinates of the center of the donut.

Inner Radius and **Outer Radius** set the inner and outer radius of the donut.

Dot Pin Attributes

Layer sets the layer of the dot.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the bounding box of the dot.

Center: x and **y** set the X and Y coordinates of the dot origin.

Width sets the width of the dot in user units (usually microns).

Height sets the height of the dot in user units (usually microns).

Placement Status displays the placement status of the instance in a cyclic field. The choices are Unplaced, None, Placed, Fixed, and Locked/Cover. If the placement status is *Unplaced*, the instance will not be visible. See **Placement Status Valid Values**.

ROD Name displays the name of the dot if it is a ROD object.

Ellipse or Circle Attributes

Layer sets the layer of the ellipse or circle.

© 2020 All Rights Reserved.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

Left, **Right**, **Bottom**, and **Top** set the coordinates of the box that defines the ellipse or circle.

Group Attributes

Name displays the name of the group.

Origin X and **Origin Y** sets the origin point of the group.

Orientation sets whether the group is rotated or mirrored. See **Orientation Valid Values**.

Query Group Member button, when clicked, displays the appropriate Edit Properties form for the types of objects contained in the group. For example, if the first object in the group is a rectangle, the Edit Rectangle Properties form appears. To continue through the objects in the group, use the Next and Previous buttons on the form. You can edit the properties of each member of the group using these Edit Properties forms. This chapter describes all of the Edit Properties forms that might appear as a result of the query.

Virtual Hierarchy / VH Clone Attributes

Name displays the name of the virtual hierarchy or virtual hierarchy clone.

Master displays the library, cell, and view name of the virtual hierarchy or virtual hierarchy clone.

Type displays the type of virtual hierarchy or virtual hierarchy clone, *Generated* or *Created*.

Frozen sets whether the virtual hierarchy or virtual hierarchy clone is frozen or not.

Family displays the family of the virtual hierarchy or virtual hierarchy clone.

Origin X and **Origin Y** sets the origin point of the virtual hierarchy or virtual hierarchy clone.

Rotation sets whether the virtual hierarchy or virtual hierarchy clone is rotated or mirrored. See **Orientation Valid Values**.

Number of Members displays the number of members in the virtual hierarchy or virtual hierarchy clone.

© 2020 All Rights Reserved.

Instance Attributes

Library, **Cell**, and **View** set the library, cell, and view names of the master cell for this instance.

Note: You can use the SKILL function leEnableInstPropEditFields to enable or disable the *Library* or *Cell* fields for an instance in the Property Editor form.

Browse lets you select the library, cell, and view names by clicking on them in the browser.

Origin: X and **Y** set the X and Y coordinates of the origin of the instance.

Name sets the name assigned to this instance. The layout editor automatically assigns instance names that begin with the letter `I`, followed by a number.

Rotation sets whether the instance is rotated or mirrored. See **Orientation Valid Values**.

Cell Type read only field displays the master cell type used to describe the nature of a physical block or cell. Layout editor does not support any level one editing on any of these design objects. These cell types are used in Virtuoso Layout Suite XL and Virtuoso Floorplanning. See **Cell Type Valid Values**.

Placement Status displays the placement status of the instance in a cyclic field.The choices are Unplaced, None, Placed, Fixed, and Locked/Cover. If the placement status is *Unplaced*, the instance will not be visible. See **Placement Status Valid Values**.

Mosaic Attributes

Library, **Cell**, and **View** set the library, cell, and view names of the master cell for this instance.

Name sets the name assigned to this instance. The layout editor automatically assigns instance names that begin with the letter `I`, followed by a number.

Origin: x and **y** set the X and Y coordinates of the origin of the instance.

Rows and **Columns** set the number of rows and columns in an array of instances.

Delta X and **Delta Y** set the spacing between rows and columns in an array of instances.

Rotation sets whether the mosaic is rotated or mirrored. See **Orientation Valid Values**.

© 2020 All Rights Reserved.

Label Attributes

Layer sets the layer on which the label is drawn.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the label.

Text sets the text that appears in the label.

Origin: x and **y** set the X and Y coordinates of the label origin.

Height sets the height of the label, in user units (usually microns).

Rotation sets whether the label is rotated or mirrored. See **Orientation Valid Values**.

Font sets the text style of the label.

Justification sets the location of the label origin. The origin appears as a small square on the label when you place or select it.

Drafting prevents the label from being rotated more than 90 degrees.

Overbar is a display option that determines how text strings containing underscore characters are displayed in a layout window.

When the overbar is disabled (default), the software displays underscore characters (_) as part of the text string. When the overbar is enabled, the software interprets underscore characters (_) in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the examples.

Property Name	Appears in Design Window As
_abcde	$\overline{\text{abcde}}$
ab_cde	$\text{ab}\overline{\text{cde}}$
_abc_de	$\overline{\text{abc}}\text{de}$
ab_cd_ef_gh_ij	$\text{ab}\overline{\text{cd}}\text{ef}\overline{\text{gh}}\text{ij}$

Line Attributes

Layer sets the layer of the line.

© 2020 All Rights Reserved.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox sets the coordinates of the box surrounding the line.

Points sets the coordinates of each point of the line.

ROD Name displays the name of the line if it is a ROD object.

Path Attributes

Layer sets the layer of the path.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the path.

Points sets the coordinates of each point of the path.

Width specifies the path width in user units (usually microns).

End Type controls how the path ends are drawn.

truncate: the path terminates at the end point with no overlap.

extend: the path terminates one-half the path width past the end point.

round: the path terminates one-third the path width past the end point, in an octagonal (chamfered) shape. Tapering starts at the endpoint of the path and the end segment is one-third the path width, with the other two segments calculated using one-third the width times the square root of two.

variable: the path terminates beyond the end point at the distance you enter in the *Begin Extension* and *End Extension* fields.

Begin Extension and **End Extension** sets the length of the beginning and ending extension in user units if *Type* is set to *variable*.

Route Status indicates the status of the route containing the path.

Routed indicates normal routing status.

Fixed indicates a route that should not be edited automatically, though it can be edited manually.

© 2020 All Rights Reserved.

Locked/Cover indicates a route that should not be edited automatically or manually. The route forms a part of the base array. To edit the route, change the routing status to *Routed* or *Fixed*.

Part of a Route indicates whether or not the path is part of a route. By default, it is FALSE for paths.

ROD Name appears if the path is a ROD object.

Subpart appears if the path is a multipart path and has subparts.

(ICADVM20.1 Virtuoso RF Solution Only) For curved paths, *End Type*, *Begin Extension*, *End Extension*, *Part of a Route*, and *Route Status* attributes are disabled.

Path Segment Attributes

Layer sets the layer of the path.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the path.

Width specifies the path width in user units (usually microns).

This field always displays the width in orthogonal units, even though a path segment may be diagonal.

Begin Point sets the coordinates of the beginning point of the path segment.

End Point sets the coordinates of the beginning point of the path segment.

Begin Style controls how the path segment beginning is drawn. See **Pathseg Begin and End Styles**.

End Style controls how the path segment end is drawn. See **Pathseg Begin and End Styles**.

Begin Extension and **End Extension** set the length of the beginning and ending extension in user units if *Type* is set to *variable*.

Route Status indicates the status of the route containing the pathseg.

Routed indicates normal routing status.

© 2020 All Rights Reserved.

Fixed indicates a route that should not be edited automatically, though it can be edited manually.

Locked/Cover indicates a route that should not be edited automatically or manually. The route forms a part of the base array. To edit the route, change the routing status to *Routed* or *Fixed*.

Part of a Route indicates whether or not the pathseg is part of a route.

Topology indicates the topology pattern of the pathseg. The valid topology values are *None, RING*, *STRIPE*, *FOLLOWPIN*, *IOWIRE*, *COREWIRE*, *BLOCKWIRE*, *PADRING*, and *BLOCKRING*. The default value is *None*.

Note: The *Topology* field is visible only if the *Part of a Route* field is set to *FALSE*. When the *Part of a Route* field is set to *TRUE*, the Edit Properties form title is updated to Edit PathSeg in Route Properties.

ROD Name appears if the path is a ROD object.

Subpart appears if the path is a multipart path and has subparts.

Setup Constraint Group displays the constraint group from which the data is initialized.

ROD Multipart Path - Master Path Fields Attributes

Layer sets the layer of the master path.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the path.

Points sets the coordinates of each point of the master path.

Width specifies the path width in user units (usually microns).

Type controls how the path ends are drawn.

Begin Extension and **End Extension** set the length of the beginning and ending extension in user units if *Type* is set to *variable*.

Choppable indicates whether or not a ROD path can be chopped. The value must be `t` or `nil`. When a path has subparts and the master path is choppable, all subpaths and sets of subrectangles must be choppable also. When a path has subparts and the master path is not

© 2020 All Rights Reserved.

choppable, each subpath and/or set of subrectangles can be choppable or not. The default is choppable.

ROD Name displays the name of the MPP.

Subpart opens one of the three Edit ROD Subpart forms, Offset Subpath, Enclosure Subpath, or Subrectangle. See the ROD Subpart Form for a description of the fields and options.

Pin Name or Other Text Display Attributes

Layer sets the layer of the text display.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the text display.

Text displays the text of the text display for the associated pin and is not editable.

Height sets the height of the text display in user units (usually microns).

Rotation sets whether the text display is rotated or mirrored. See **Orientation Valid Values**.

Font sets the text style of the text display.

Justification sets the location of the origin of the text display origin. The origin appears as a small square on the pin name when you place or select it.

Drafting prevents the text from being rotated more than 90 degrees.

Overbar is a display option that determines how text strings containing underscore characters are displayed in a layout window.

When the overbar is disabled (default), the software displays underscore characters (_) as part of the text string. When the overbar is enabled, the software interprets underscore

© 2020 All Rights Reserved.

characters (_) in the text string name as toggle switches that control where overbars begin and end. Overbars appear above the text string, as shown in the examples.

Property Name	Appears in Design Window As
_abcde	$\overline{abcde}$
ab_cde	$ab\overline{cde}$
_abc_de	$\overline{abc}de$
ab_cd_ef_gh_ij	$ab\overline{cd}ef\overline{gh}ij$

Polygon Attributes

Layer sets the layer of the polygon.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox sets the coordinates of the box surrounding the polygon.

Points sets the coordinates of each point of the polygon.

ROD Name displays the name of the polygon if it is a ROD object.

Rectangle Pin Attributes

Layer displays the layer of the rectangle.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

Left, **Right**, **Bottom**, and **Top** displays the coordinates of the pin.

Edit is a radio button that lets you edit the coordinates. This is the topmost edit button.

Width and **Height** display the width and height of the pin.

Placement Status displays the placement status of the pin in a cyclic field. The choices are Unplaced, None, Placed, Fixed, and Locked/Cover. If the placement status is *Unplaced*, the instance will not be visible. See **Placement Status Valid Values**.

Pin Name displays the name of the pin.

© 2020 All Rights Reserved.

Terminal Name displays the name of the terminal.

Route Method displays the routing method of the pin in a cyclic field. The choices are symbolic and geometric.

Note: When *Update Net Signal Type* is enabled, terminals with nets that have a *Signal Type* of Power or Ground in the schematic are automatically given a *Route Method* of Geometric when either the *Generate All From Source* or *Update Components and Nets* commands are run.

MPT Coloring: **Color** and **State** displays the color and state of the pin.

Polygon Pin Attributes

Layer displays the layer of the rectangle.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

bBox displays the coordinates of the box surrounding the pin.

Points displays the coordinates of each point of the pin.

Placement Status displays the placement status of the pin in a cyclic field. The choices are Unplaced, None, Placed, Fixed, and Locked/Cover. If the placement status is *Unplaced*, the instance will not be visible. See **Placement Status Valid Values**.

Pin Name displays the name of the pin.

Terminal Name displays the name of the terminal.

Route Method displays the routing method of the pin in a cyclic field. The choices are symbolic and geometric.

Note: When *Update Net Signal Type* is enabled, terminals with nets that have a *Signal Type* of Power or Ground in the schematic are automatically given a *Route Method* of Geometric when either the *Generate All From Source* or *Update Components and Nets* commands are run.

MPT Coloring: **Color** and **State** displays the color and state of the pin.

Note: (ICADVM20.1 Virtuoso RF Solution Only) For curved polygon pins, you can view attributes described above.

© 2020 All Rights Reserved.

Row Attributes

Base Name allows you to enter a unique name for each row object you create. You enter the base name and the software adds an underscore followed by a number which is greater than or equal to 1. As you enter new sites, each site is incremented by 1. For example, if your base name is ROW_1, the name assigned to the first site is ROW_1_1, the next is ROW_1_2, the next is ROW_1_3, and so on. If you do not enter a custom row name the software assigns the name row_number, which increments by 1 each time you add a site and rows.

Row Width is a display only field displaying the width of the row.

Rotation allows you to set the row orientation. See **Orientation Valid Values**.

Origin X and **Y** allows you to set a corner of the row at a specific point in the cellview.

Site Definition Name is a read only field displaying the name of the current row.

Technology Library is a read only field displaying the technology library the original row was defined in.

Site Information is a read-only field displaying the type of site definition (core or pad) that is the current site in the Site Definition Name field. This information contains the type (core or pad), the width and height of the site definition, and three types of symmetry. This information is specified in the technology library site definitions. The symmetry can be SymmetricInX, SymmetricInY, or SymmetricInR90.

Number of Sites sets the number of sites in each row. The number of sites is any integer equal to or greater than zero.

Templates allow you to place rails and/or components automatically into the rows.

Type radio buttons display Rails, Allowed Components, or Both in the Pre-Defined section of the form. You move them into the Selected section of the form using the arrows. Once they are in the Selected section of the form, you add the components individually selected from the Selected field.

PR Boundary Attributes

Points display the coordinates of each point of the PR boundary.

BBox displays the coordinates of the box surrounding the PR boundary.

Area displays the total area of the PR boundary in user units.

© 2020 All Rights Reserved.

Rectangle and **Rectilinear** display whether a boundary is a rectangle or a polygon. The PR boundary is always a rectangle.

Halo allows you to optionally create a new or edit an existing area halo associated with the PR boundary object. This area halo is always on offset to the bounding box of the PR boundary object.

IOBox displays the **Left**, **Bottom**, **Right**, and **Top** coordinates of the bounding box of the IOBox. IOBox represents the area reserved for pads.

CoreBoxSpec displays the **Left**, **Bottom**, **Right**, and **Top** coordinates of the bounding box. CoreBoxSpec is a rectangular grid of sites where multiple rows can be placed in horizontal or vertical direction. This is the area where standard cells can be placed. An IOBox, together with the CoreBox, indicates the area for the IO cells.

Site Pattern displays the site definition as core or pad, as defined in the technology file.

Horizontal specifies whether the row orientation is horizontal or not.

Spacing displays the row offset value in user units.

Flip Type displays which rows in the site can be flipped. Cyclic choices include *Even*, *Odd*, *None*.

Spacing Type displays the rows to which the spacing value applies. Cyclic choices include *Every*, *Even*, *Odd*.

© 2020 All Rights Reserved.

In the example below, the prBoundary has an IO box and a coreBox. The coreBox has four horizontal rows of siteDef A.

All rows are separated by a distance row spacing. Every odd numbered row is flipped. The number of rows and row geometry is computed based on the overall bBox of the coreBox and the siteDef of the coreBox.

API is available to retrieve the number of rows and bBox of each row. For information about accessing boundary attributes, see Boundary Attributes and Boundary SKILL Functions in the *Virtuoso Design Environment SKILL Reference*.

Snap Boundary Attributes

Points sets the coordinates of each point of the snap boundary.

bBox displays the coordinates of the box surrounding the snap boundary.

Area displays the total area in user units of the snap boundary.

Area Boundary Attributes

Name sets the name of the area boundary.

© 2020 All Rights Reserved.

Points sets the coordinates of each point of the area boundary.

bBox displays the coordinates of the box surrounding the area boundary.

Area displays the total area in user units of the area boundary.

Rectangle and **Rectilinear** displays whether a boundary is a rectangle or a polygon.

Cluster Boundary Attributes

Name sets the name of the cluster boundary.

Cluster Name displays the name of the associated cluster.

Points sets the coordinates of each point of the cluster boundary.

bBox displays the coordinates of the box surrounding the cluster boundary.

Area displays the total area in user units of the cluster boundary.

Rectangle and **Rectilinear** displays whether a boundary is a rectangle or a polygon.

Routing, Slot, Pin, or Feed Through Blockage Attributes

Owner Information sets the name of the (optional) owner of the blockage.

Points sets the coordinates of each point of the cluster boundary.

Area displays the total area in user units of the blockage.

Type displays the type of blockage you are editing. In this case, the blockage would be of the type: routing, slot, pin, feed, or placement.

Placement Blockage Type displays the type of placement blockage: Hard, Partial, Soft or Macro-Only.

Max Density displays the maximum percentage of the total design area that the metal in the specified layer occupies.

© 2020 All Rights Reserved.

The valid values of the *Max Density* field based on the placement blockage type are described below:

Placement Blockage Type	Max Density (Valid Values)	Editable (Yes/No)
Hard	0%	No
Partial	Value greater than 0% and less than 100%. The value can be a floating point number. Default value: *50%*	Yes
Soft	0%	No
Macro-Only	100%	No

Layer selects the layer on which the blockage is enforced.

Layer Filter enables you to search for a layer-purpose pair and narrow down the available choices in the *Layer* list.

Material selects the material on which the blockage is enforced.

Use Effective Width enables the *Effective Width* field, when clicked.

Effective width sets the value which is checked against the layer spacing rule or table spacing rules, if defined.

Allow PGNet allows shapes with signal types power and ground to be placed within the blockage area.

Color lets you select the color in which the blockage is created. The default value is *grayColor*. The options available in the *Color* field are based on the layer selected in the *Layer* field. If you select a layer in the *Layer* field that support *mask1Color* and *mask2Color*, the *Color* field will have the options, *grayColor*, *mask1Color*, *mask2Color*, *blackColor*, and *multiColor*.

© 2020 All Rights Reserved.

Placement Halo Blockage Attributes

Left Offset specifies the number of user units the placement halo is offset from the left of the instance bBox.

Right Offset specifies the number of user units the placement halo is offset from the right of the instance bBox.

Bottom Offset specifies the number of user units the placement halo is offset from the bottom of the instance bBox.

Top Offset specifies the number of user units the placement halo is offset from the top of the instance bBox.

Via Attributes

Via Definition cyclic field lists all valid via definitions available in the effective technology file. Choosing a via from the cyclic field selects the via from the effective technology file.

Net Name applies a net name to the via you are placing in the cellview.

Technology Library / Via Type is a read only field that displays the technology library from which the selected via definition is obtained. If you select a via definition from the list which is defined in a different library, the field is updated to that library name. The field also displays the type of `viaDef` displayed in the *Via Definition* field. The `viaDefs` are `standard` or `custom`.

Rotation displays the current rotation of the via.

Origin: X and **Y** set the X and Y coordinates of the via origin.

Justification sets the origin of a via. Choose `offset` to manually set the X and Y origin points. Cyclic field choices are: `offset`, `lowerLeft`, `centerLeft`, `upperLeft`, `lowerCenter`, `centerCenter`, `upperCenter`, `lowerRight`, `centerRight`, `upperRight`. The value `offset` is not available for a cdsVia.

Reset Parameters to enables you to reset the values in the form to the technology database values if you want to revert from the manually updated parameter values to the defaults. You can reset the values to *Minimum Rules then ViaDef*, *Minimum Rules*, or *Minimum Rule Enclosures*.

Compute Mode Section

Rows and **Columns** set the number of rows or the number of columns of via cuts in a via array. If you change the values in the *Rows* and *Columns* fields, the values in the

© 2020 All Rights Reserved.

other fields, such as *Width*, *Length*, *Row Spacing*, and *Column Spacing*, are dynamically updated.

Note: When you update the value in the *Row* or *Column* field, this will the update other values in the form and apply the constraints. This will happen when you finish editing the values in the *Row* or *Column* fields and click at some other area in the form.

Swap button swaps the values in the *Rows* and *Columns* fields. If the swapped values violate the minimum rules, the spacing and enclosure values are re-calculated based on the *Reset Parameters to* setting.

Width and **Height** set the width and height of the via, in user units (typically microns).

Cut Section

Cut Class cyclic field displays cut class choices only if a `cutClasses` constraint has been defined for the cut layer of the selected viaDef in the `layerRules` section of the technology file. The following are examples of cut classes:

`Small` is a single cut via where both dimensions are short.
`Bar` is a large via that has one long and one short dimension. The shorter dimension is equal to the width or length of `Small`.
`Large` is a large via that has two long dimensions, equal to the longer dimension of `Bar`.

If you update the *Cut Class* selection, the values in the other fields, such as *Rows*, *Columns*, *Width*, *Length*, *Row Spacing*, and *Column Spacing*, are dynamically updated.

Width and **Length** set the width and length of the via cut, in user units (typically microns). If a *Cut Class* is selected, the cut *Width* and *Length* are automatically updated to the cut size values defined for that cut class in the `layerRules` section of the technology file.

Swap button swaps the values in the *Width* and *Length* fields.

Row Spacing sets the distance between the horizontal edge to edge cuts in a via.

Column Spacing sets the distance between the vertical edge to edge cuts in a via.

Swap button swaps the values in the *Row Spacing* and *Column Spacing* fields.

Edit Cut Pattern opens the Edit Cut Pattern Form and enables you to edit cut patterns on via arrays.

Compute Enclosures automatically calculates enclosures.

© 2020 All Rights Reserved.

Note: The *Compute Enclosures* setting does not impact the via you create in the *Auto* mode.

Enclosures set directional offset distances on Layer 1, the bottom routing layer, and Layer 2, the top routing layer.

Layer 1 Left sets the horizontal distance on the left side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 2 Left sets the horizontal distance on the left side of the via from the cut edge to the enclosure edge on the layer designated layer 2 (top routing layer).

Layer 1 Right sets the horizontal distance on the right side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 2 Right sets the horizontal distance on the right side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (top routing layer).

Layer 1 Top sets the vertical distance on the top side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 2 Top sets the vertical distance on the top side of the via from the cut edge to the enclosure edge on the layer designated layer 2 (top routing layer).

Layer 1 Bottom sets the vertical distance on the bottom side of the via from the cut edge to the enclosure edge on the layer designated layer 1 (bottom routing layer).

Layer 2 Bottom sets the vertical distance on the bottom side of the via from the cut edge to the enclosure edge on the layer designated layer 2 (top routing layer).

Layer 1 Implant X sets the horizontal distance from the left and right cut edge to the implant enclosure edge on the layer designated layer 1 (bottom routing layer). This field is for informational purposes only, and will always be grayed out.

Layer 2 Implant X sets the horizontal distance from the left and right cut edge to the implant enclosure edge on the layer designated layer 2 (top routing layer). This field is for informational purposes only, and will always be grayed out.

Layer 1 Implant Y sets the vertical distance from the top and bottom cut edge to the implant enclosure edge on the layer designated layer 1 (bottom routing layer). This field is for informational purposes only, and will always be grayed out.

Layer 2 Implant Y sets the vertical distance from the top and bottom cut edge to the implant enclosure edge on the layer designated layer 2 (top routing layer). This field is for informational purposes only, and will always be grayed out.

Route Status indicates the status of the route containing the via.

© 2020 All Rights Reserved.

Routed indicates normal routing status.

Fixed indicates a route that should not be edited automatically, though it can be edited manually.

Locked/Cover indicates a route that should not be edited automatically or manually. The route forms a part of the base array. To edit the route, change the routing status to *Routed* or *Fixed*.

Part of a Route indicates whether or not the via is part of a route.

The following fields are available only for a cdsVia:

Metal Purpose displays the metal layer purpose.

Cut Purpose displays the via layer purpose.

Other Purpose displays the implant and well layer purpose.

MPT Coloring

For a selected via, you can set the colors for Layer1, CutLayer, and Layer2.

The table below summarizes the MPT coloring fields that are available for the layers.

Via Layer	Available Colors
Layer1	grayColor | mask1Color | mask2Color | mask3Color
CutLayer	grayColor | mask1Color | mask2Color | mask3Color
Layer2	grayColor | mask1Color | mask2Color | mask3Color

Via Stack Attributes

Rotation displays the current rotation of the via stack.

Justification sets the origin of a via stack. Cyclic field choices are: `bottom-left`, `center-left`, `top-left`, `bottom-center`, `center-center`, `top-center`, `bottom-right`, `center-right`, or `top-right`.

Origin: X and **Y** set the X and Y coordinates of the via stack origin.

Compute From Section

© 2020 All Rights Reserved.

Top Rows and **Top Columns** sets the number of rows and columns for the top via in the via stack.

Width and **Height** set the width and height of the via stack, in user units (typically microns).

Definition Section

Start Layer lets you select one of the two layers between which you want to place the via stack. The layer selected in the *End Layer* list does not appear in the *Start Layer* list.

End Layer lets you select one of the two layers between which you want to place the via stack. The layer selected in the *Start Layer* list does not appear in the *End Layer* list.

Transitions: Via Definitions lets you select viaDefs for each layer pair between the specified *Start Layer* and *End Layer*.

Cut Class cyclic field displays cut class choices only if a `cutClasses` constraint has been defined for the cut layer of the selected viaDef in the `layerRules` section of the technology file.

Modgen Attributes

Name sets the name of the modgen.

Origin sets the origin point of the modgen.

Rotation sets whether the modgen is rotated or mirrored. See **Orientation Valid Values**.

Rows and **Columns** in the modgen.

Horizontal and **Vertical Alignment**

> **Alignment** sets the alignment of the modgen.
>
> **Custom Spacing** sets the custom spacing of the modgen.
>
> **Reference**: **Layer** and **Purpose** sets the layer and purpose of the reference layer of the modgen.

Number of Members in the modgen.

© 2020 All Rights Reserved.

Placement Status Valid Values

Value	Description
Unplaced	Unplaced instances can be placed and moved. Indicates that the current placement is arbitrary and should be changed. Unplaced objects do not contribute to the bounding box of the containing design, and they will not be returned by a region query.
None	Default value.
Placed	Placed instances can be moved by automatic and interactive commands. Indicates that an effort has been made by the software to place this object but it may be placed in future attempts.
Fixed	Fixed instances can be moved by interactive commands.
Locked/Cover	Locked instances cannot be moved by any commands. The placement of this object should not be changed.

Cell Type Valid Values

Blocks/Cells	Type
Custom Cells	none
Soft Cells	blockBlackBox
	softMacro
Hard Cells	block
	blockRing
	cover
Core Cells	core
	coreSpacer
	coreAntenna
	coreFeedThru
	coreTieHigh
	coreTieLow

© 2020 All Rights Reserved.

Blocks/Cells	Type
	coreSpacreEndCapPre
	coreSpacerEndCapPost
	coreWellTap
I/O Cells	coverBump
	pad
	padInput
	padOutput
	padInOut
	padPower
	padSpacer
	padAreaIO
	cornerTopLeft
	cornerTopRight
	cornerBottomLeft
	cornerBottomRight
	corner

Pathseg Begin and End Styles

truncate: the path segment begins at the begin point with no overlap.

© 2020 All Rights Reserved.

extend: the path segment extends from the begin point by one-half the path width.

variable: the path segment extends from the begin point by whatever value you enter in the *Begin Extension* field, in the example below by 0.25 microns from the begin path point.

custom: the path beginning extends in an octagonal (chamfered) shape. Each dimension of the chamfer is equidistant, except where the distance adjusts to accommodate 45-degree angled paths.

© 2020 All Rights Reserved.

The custom path style adjusts to the path width when a 45-degree path end overlaps with other custom path ends. The path ends match when the end points overlap, as in the example below.
.

Orientation Valid Values

Value	Description
RO	no rotation
R90	rotated 90 degrees
R180	rotated 180 degrees
R270	rotated 270 degrees
MY	mirrored over the Y axis
MYR90	mirrored over Y, rotated 90 degrees
MX	mirrored over the X axis
MXR90	mirrored over X, rotated 90 degrees

© 2020 All Rights Reserved.

Edit Slot Form

See the Create Slot Form for a description of the fields and options.

Related Topics

Editing Slots

© 2020 All Rights Reserved.

Edit Synchronous Clone Form

Apply same action to all other clones lets you edit a clone without the Edit Synchronous Clone form being displayed while editing other clones.

Environment variable: enableSyncCloneEditing

Cancel closes the form without editing the clone.

OK edits the clone and closes the form.

© 2020 All Rights Reserved.

Enter Points Form

XY enables you to specify the points for an object, such as a rectangle, a polygon, an arc, a path, a circle, an ellipse, a donut, geometric wires, instances, pins, labels, vias, multipart paths, fluid guard rings, MPP guard rings, slots, and P&R objects. The object appears on the canvas at the points specified in this field. In this field you can also specify points for commands such as, Pan, Zoom, Move, Copy, Chop, and Edit In Place.

Use Relative XY indicates that the points specified in the *XY* field are the points relative to the last entered points of the command (delta X and Y).

You can enter points in the *XY* field in the following formats:

Format	"Use Relative XY" Deselected	"Use Relative XY" Selected
x y	The points are added as absolute points.	The points are added as relative points.
x1 y1 x2 y2 x3 y3 or x1 y1, x2 y2, x3 y3 or x1 y1; x2 y2; x3 y3 or (x1 y1)(x2 y2)(x3 y3) or x1:y1 x2:y2 x3:y3 or {x1 y1}{x2 y2}{x3 y3} or [x1 y1][x2 y2][x3 y3] or <x1 y1><x2 y2><x3 y3>	.The points are added as absolute points.	Case1: If a point has already been added to the command, the points in this list are added as relative points. Case2: If a point has not been added to the command, a warning message is displayed in CIW. No points are added to the canvas.

© 2020 All Rights Reserved.

Format	"Use Relative XY" Deselected	"Use Relative XY" Selected
(x1 y1) @(x2 y2) (x3 y3) or x1 y1 @x2 y2 x3 y3 or x1 y1 x2@ y2 x3 y3 or {x1 y1} {x2 @y2} {x3 y3} or x1 y1 x2 y2@ x3 y3	(x1 y1) and (x3 y3) are added as absolute points and (x2 y2) are added as relative points.	All the points are added as relative points.

Note: @(x y) or @x y or x@ y or x @y or x y@ formats imply that x y are relative points. You can prefix or suffix @ to a point to make it a relative point.

Caution

A combination of the formats mentioned in the above table is not supported in Virtuoso Layout Suite L. For example, {x1 y1}[x2 y2] is not supported as it uses a combination of curly and square brackets.

You can also enter points in the *XY* field for nested commands in the following format:

Format	Nested Commands Example	Description
(x1 y1)(x2 y2) (x3 y3) (x4 y4)	You run the `Rectangle` command and then run the `Ruler` command.	The points (x1 y1) and (x2 y2) are used to create a ruler. The points (x3 y3) and (x4 y4) are used to create a rectangle.

Note: All point adjustment commands, such as DRD, Grid, Snap Mode, Acute Angle, and Gravity are disabled if the points are entered using the *Enter Points* form.

Environment variable: useRelativeXY

© 2020 All Rights Reserved.

Related Topics

Using the Enter Points Form

leGetCoordinateForm

© 2020 All Rights Reserved.

Export Label Form

Use the **Export Label** form to export the name, coordinates, and layer-purpose pair for some or all of the labels in your design to a text file.

Export only selected exports information for only the currently selected labels only. If this option is not selected, information for all the labels in the current design is exported. If this option is not selected, exports information for all the labels in the current design.

Output File specifies the name of the file containing the label information. Type the path and name of the file into the field or use the *Browse* button to navigate to the required location on disk.

© 2020 All Rights Reserved.

Export Image Form

See *Cadence User Interface SKILL Reference* for information about this form.

© 2020 All Rights Reserved.

Filter Layers Form

Include only

Valid layers populates the *Layers* assistant with layers defined as valid in the LAYER DEFINITION section of the technology file in `techDisplays`.

LSW rule layers populates the *Layers* assistant with layers defined in the LE RULES section of the technology file in `leLswLayers`, if defined. Filtering using this selection on the form does not retain the order in which they are listed in the technology file.

User layers populates the *Layers* assistant with layers defined in the LAYER DEFINITION section of the technology file as `User-Defined Layers` in `techLayers`.

Layers with functions populates the *Layers* assistant with layers defined in the LAYER RULES section of the technology file in `layerFunctions`. If the Layers With Functions option is not selected, none of the sub-categories of functions are considered.

Layers starting with populates the *Layers* assistant with layer names typed in the field. The first character you enter adds any layer in the *Layers* assistant whose first character matches the entered character. The Layers Starting With text is case insensitive. Additional characters are matched to corresponding layer names as you add them to the field. A space is interpreted by the software as the end of one layer name and the beginning of another layer name. For example, entering "`m p`" in the field would add the layer names `metal`, `marker`, `poly`, `pimplant`, `pwell`, `pin` etc. Entering "`m pi`" would add `metal`, `marker`, `pimplant`, and `pin`, but not `poly` or `pwell`.

Purposes starting with populates the *Layers* assistant with purpose names typed in the field. The first character you enter adds any purpose in the *Layers* assistant whose first character matches the entered character. The Purposes Starting With text is case insensitive. Additional characters are matched to corresponding purpose names as you add them to the field. A space is interpreted by the software as the end of one purpose name and the beginning of another purpose name. For example, entering "`d p`" in the field would add the purpose names `dwg`, `dg2`, `pin`, etc. Purposes are recognized only by their abbreviations, not by their full names. Entering "`drawing`" results in no filtering taking place and a warning appearing (`*WARNING* Filter Layers: criteria would result in no layers shown. Aborted.`).

Exclude

Layers starting with excludes from the *Layers* assistant the layer names typed in the field. The first character you enter excludes any layer in the *Layers* assistant whose first character matches the entered character. The Layers Starting With text is case insensitive. Additional characters are matched to corresponding layer names as you add them to the field. A space is interpreted by the software as the end of one layer name

© 2020 All Rights Reserved.

and the beginning of another layer name. For example, entering "`m p`" in the field would exclude the layer names `metal`, `marker`, `poly`, `pimplant`, `pwell`, `pin` etc. Entering "`m pi`" would exclude `metal`, `marker`, `pimplant`, and `pin`, but not `poly` or `pwell`.

Purposes starting with excludes from the *Layers* assistant the purpose names typed in the field. The first character you enter excludes any purpose in the *Layers* assistant whose first character matches the entered character. The Purposes Starting With text is case insensitive. Additional characters are matched to corresponding purpose names as you add them to the field. A space is interpreted by the software as the end of one purpose name and the beginning of another purpose name. For example, entering "`d p`" in the field would exclude the purpose names `dwg`, `dg2`, `pin`, etc. Purposes are recognized only by their abbreviations, not by their full names. Entering "`drawing`" results in no filtering taking place and a warning appearing in the CIW (`*WARNING* Filter Layers: criteria would result in no layers shown. Aborted.`).

Related Topics

Palette Assistant

© 2020 All Rights Reserved.

Find Marker Form

Severity specifies the type of markers displayed. Select *all* or *none* to select all types of markers or none of the markers to be displayed.

fatal error tool generated error. Can not be signed off, as it will always result in a non-functioning design or a design that can not be manufactured. Such error are generated usually from the process end (design rule) where the far/foundry does not want to allow sign-off.

critical error tool generated error. Usually requires a design review to sign- off.

signed off critical error displays error that has been reviewed by the design team, and has been signed-off.

error displays error markers.

signed off error displays error that has been signed-off.

warning displays warning markers.

acknowledged warning displays warning that has been acknowledged by the designer.

info tool generated information.

annotation user generated information.

Environment variable: markerSeverity

Search Scope specifies what part of the design hierarchy you want to search.

cellview searches only for objects in the cellview displayed in the current window.

hierarchy starting from searches all cellviews specified in the design hierarchy. The hierarchy either starts with the *top cellview* or the *current cellview*.

library searches all cellviews in the specified library. You must type in the library name, and the library must exist in the current search path.

Environment variable: markerSearchScope

Options

Zoom To Markers enlarges the position of the marker in the window by zooming in on the specified marker; when off, displays the marker but does not zoom in.

Environment variable: markerZoom

© 2020 All Rights Reserved.

Cellview Access Mode specifies the mode for opening a cellview. You base your selection on what you want to do in the cellview.

> **read** displays the cellview but does not permit you to edit and save. Find marker searches for the markers with the criteria specified in the find marker form and opens the cellview containing the first desired marker in read mode. However, if that cellview is already opened in Edit mode, the cellview is not re-opened in read mode.
>
> **write** displays the cellview and permits you to edit and save. Find marker searches for the markers with the criteria specified in the find marker form and opens the cellview containing the first desired marker in Edit mode. If the cellview is already opened in read mode, the cellview is re-opened in Edit mode so that you can delete markers if required.

Environment variable: markerAccessMode

Find by specifies which markers will display.

> **master cell only** searches only for markers in the master cell.
>
> **all occurrences** searches for all occurrences of markers.

Environment variable: markerSearchBy

Related Topics

Finding Markers

© 2020 All Rights Reserved.

Find/Replace Form

The *Find/Replace* form comprises the following sections:

- Search
- Region
- Replace
- Search Results

Search

Search for sets the type of object searching for.

Add Criteria adds a line of fields to the form that lets you make your search more specific. For more information, see Objects Criteria Table.

> **Note:** The Search command will not accept ranges in the Add Criteria fields. For example, if you enter two path widths you get the error message, “No figures found that meet the current search criteria.”
>
> **cyclic field** lets you search for specific attributes, instance, library, or view name, or properties for this type of object. The cyclic field selections change to reflect the attributes of the type of object you select in the *Search for* field.
>
> **condition/logic cyclic field ==** or **!=** lets you search for character strings or numerals that are an exact match (==) or are not a match (!=); to get matches that contain a string, a leading and/or trailing asterisk must be used. The symbols <, <=, >, >= are used, where applicable. *EXIST* and *!EXIST* locate or exclude discrete blocks of information anywhere in the string.
>
> **value field** sets the value of the attribute or property to use in the search. In the text field, you can specify the search string using the following wildcard characters, as described below.

Wildcard	Description	Example
`*`	Matches any number of characters.	`po*` finds `point`, `points`, `polygon`, but not opposite, `export`.
`.`	Matches any alphabetical character.	`p.n` finds `pan`, `pin` but not `polygon`, `phon`.

© 2020 All Rights Reserved.

`[ ]`	Matches any character within the brackets.	`t[io]p` finds `tip`, top but not `tap`
`-`	Matches any one of a range of characters.	`t[d-p]` finds `tip`, `top` but not `tap`.
`\`	Allows characters [, *, and \ to be treated literally.	

Expression	Result
`Ci*e.`	Finds `Circle1`, `Circlea`, `Cie2` but not `circle1`, `circle12`, `Cie`, and `Circle*1`.
`[a-z,A-Z]ire.[0-9]`	Finds `wire`, `Wire`, `Fire`, `fire` but not `rewire` and `afire`.
`poly\*. or poly[*].`	Finds `poly*1`, `poly*2` but not `polygon`.
`A.B[0-9]C[0-9]`	Finds `A1B1C1`, `A2B2C3` but not `ABC` and `AB2C4`.

Note: If a layer is selected in the cyclic field and the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form, only valid layers displayed in the *Layers* assistant are populated in the value field. You can search for a specific layer-purpose pair using the search filter.

Delete deletes this line of criteria from the form.

Match all of the above displays objects that meet all the search criteria specified above.

Match any of the above displays objects that meet any of the search criteria specified above.

Case Sensitive makes the search case-sensitive.

Search Hierarchy Range sets how many levels of the hierarchy are to be considered for search.

current cellView the current cellview is considered. This option is selected by default in the drop-down list.

current to bottom the current level of hierarchy to the level specified in the *Display Level To* field in the Display Options Form (stop level) is considered.

current to stop level the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options Form (stop level) is considered.

© 2020 All Rights Reserved.

current to user level the current level in the hierarchy to the level that you specify.

Region

Entire CellView considers the entire cellview for the Search command. This option is selected by default.

Viewing Area considers the portion of the cellview that is displayed in the canvas at that instant for the Search command.

Draw a Rectangle Region on the Canvas or **Draw a Polygon Region on the Canvas** enables you to specify a rectangular or polygonal area for the Search command. Select the required option, and then click in the canvas to specify the vertices of the rectangle or polygon.

Search sets the object searching for. The option is enabled only when the viewing area or the rectangle or polygon area in the *Region* section.

Enclosed Figures searches the objects completely enclosed by the viewing area or the rectangle or polygon area.

Enclosed and Crossed Figures searches the objects completely or partially enclosed by the viewing area or the rectangle or polygon area.

Replace

(*The options in this section are available only in Layout XL and higher tiers.)*

Replace lets you replace the object's *Find* results with a new object, attribute value, or property value. The cyclic field contains the frequently used properties. In the text field, you can specify the value of the property with which you want to replace the original value. The value specified replaces the property values of all the objects matching the search criteria. For example, to replace parameter values for a parameterized cell, set *Replace* to *property* and type the parameter name.

Add Criteria adds a line of fields to the form that lets you make your replace more specific. For more information, see Objects Criteria Table.

© 2020 All Rights Reserved.

You can use partial strings and wildcards in the *Replace* field. The table below and the following examples describe the process:

Replace String-Tagged Characters	Description
`{n}`	Interprets n as an integer from 1 to 9. Tagged fields correspond to `*` or `.` or `[c..]` or `.*` in the search string. The first nine wildcards in the search pattern are tagged. If a tagged field that is used in the replace string does not exist, it is replaced by a NULL string.
`*nth "*"`	Interpreted as {n}, n ranging from 1 to 9. If more than nine wildcards exist in the search string and more than nine `*`'s in the replace string, the system substitutes the first nine `*`'s in the replace string with the corresponding nine tagged fields from the search string. The remaining `*`'s in the replace string are substituted with a NULL string.

Note: If the wildcards appear together in a sequence in the find string, each corresponds to a different tag except when . and * appear together, in that order. In this case, the wildcards are treated together as one tag.

Example	Find Expression	Replace Expression
Replace `I0`, `I1`, `I2`, …`I10` with `M1`, `M2`, …`M10`	`I*`	`M*` or *M{1}* **Note:** The `{1}` tag or `*` in the replace string corresponds to the characters represented by the first asterisk in the find string.
Swap the start and end values of strings with prefix `ABC`	`ABC<*:*>`	`ABC<{2}:{1}>` **Note:** The `{2}`and `{1}` tags in the replace string correspond to the characters represented by the second and first asterisk in the find string, respectively.
Replace `PN`, `PN_`, `PN_IN` with `PIN`, `PIN_`, `PIN_IN`	`PN*`	`PIN*` or `PIN{1}`

© 2020 All Rights Reserved.

Replace `AfooB`, `AblahB` with `PfoofooQ`, `PblahblahQ`, respectively.	`A*B`	`P{1}{1}Q`
Replace `AxyzB` with `CxD`	`A.*[x-z]B`	`C*D` **Note:** `*` in the replace string is interpreted as the first tagged field, which is the "`.`" wildcard. `C***D` or `C{1}{2}{3}` will yield `CxyzD`
Replace `AxyzB` with `CxyD`	`A.*[x-z]B`	`C*D` **Note:** * in the replace string is interpreted as the first tagged field in the find string, which is the sequence `" .* ". C**D` or `C{1}{2}` yields `CxyzD`.
Replace `A.B` with `A*B`	`A*B` or `A.B`	`A\*B`
Replace `A[1]`,`A[2]`, `A[3]` with `A{1}`,`A{2}`, `A{3}`	`A\[*] or A\[[0-9]]`	`A\{{1}}` or `A\{*}` **Note:** The `\` before the first `{` in the replace string treats the `{` as a literal and not as a tagged field.

Note: If the find string does not have a wildcard character and the replace string does, a substitution is not achieved.

Search Results

Zoom To Figure centers the cellview around the current object in the search group when you click *Apply*, *Previous*, or *Next*.

Current Figure lets you specify the number of the objects in the search group that you want to highlight as you click *Previous* or *Next*. The value in this field is the range of the current figures found in the search group. Press `Tab` after you specify the figure number to highlight the figure.

Figure Count, after you click *Apply*, shows the number of objects found by *Find/Replace*. When a "current to bottom" search is done, the figure count returns the total number of figures found. In this case, the *Current Figure* field returns the figures with unique database IDs (dbIds).

Add Select selects the current object in the search group.

© 2020 All Rights Reserved.

Deselect deselects the current object in the search group.

Replace edits or replaces the current object with the settings shown in the *Replace* field.

Previous highlights the previous object in the search group.

Next highlights the next object in the search group.

Find searches for objects matching the settings in the Find/Replace form.

Select All selects all objects found by *Find/Replace*.

Deselect All deselects all objects found by *Find/Replace*.

Replace All replaces all objects found by *Find/Replace* with the settings shown in the *Replace* field.

Close closes the Find/Replace form.

Related Topics

Using Find/Replace

© 2020 All Rights Reserved.

Flatten Form

Hierarchy Level controls the number of design hierarchy levels that are flattened. You can move an instance up one level or move the contents of all displayed instances up to the current cellview level. You can select *one level*, *displayed level*, or *user level*. You can select a value if you select the *user level* option.
Environment variable: flattenMode

Pcells flattens any selected parameterized cells.
Environment variable: flattenPCells

Vias flattens the selected vias. This option is selected by default. If this option is not selected, geometries in the selected via instances are preserved.
Environment variable: flattenVias

Preserve

Pins preserves the connectivity information of flattened pins.
Environment variable: preservePins

Pin geometries preserves the geometric information of flattened pins. Deselect the *Preserve: Pins* option to select this option.
Environment variable: preservePinFigs

Note: Use the environment variable flattenKeepLabels to retain labels on shapes when an instance is flattened.

Term name preserves the terminal name of flattened pins.
Environment variable: preserveTermName

ROD objects preserves the attributes of ROD objects (such as object name, alignments, multipart path subparts, multipart rectangles and subrectangles) when an instance is flattened. The option is off by default. When this option is off, ROD objects become ordinary unnamed objects, the sub-parts of multipart paths become ordinary paths and rectangles, the master rectangles in multipart rectangles become ordinary, unnamed rectangles, and the subrectangles of multipart rectangles become ordinary rectangles.
Environment variable: preserveRodObjects

Selections retains the selection of all objects at instance level after the objects are flattened.

Detached blockages deletes detached blockages which would otherwise remain after an instance containing detached blockages is flattened.
Environment variable: delDetachedBlockages

Labels controls whether labels on shapes are retained when an instance is flattened.

© 2020 All Rights Reserved.

Environment variable: flattenKeepLabels

Related Topics

Flattening Instances

Generating Transparent Layout Hierarchy

leHiFlatten

© 2020 All Rights Reserved.

Flip Form

Specify reference point enables the text field where you can specify the points to which the object should be flipped.

Flip Horizontal mirrors the object around the X axis based on the specified reference point.

Flip Vertical mirrors the object around the Y axis based on the specified reference point.

Related Topics

Flipping Objects

leHiFlip

© 2020 All Rights Reserved.

Group Return

Return Mode selects the type of return mode. The choices are: group or group and instance.

Remember return mode uses the value selected in the Return Mode when the *Group Return* command is run again.

© 2020 All Rights Reserved.

Freeze Pcell Instance Form

Library lets you select the name of the library in which you want to create the non-Pcell instance.

Note: If one Pcell instance is selected for freeze, the library name in which the Pcell instance is saved is automatically populated in the *Library* field. If multiple Pcell instances are selected, the value `Default` is populated in the *Library* field. The `Default` value implies that each non-Pcell instance will be created in its respective library.

Cell lets you type the name of the non-Pcell instance.

Note: A unique name is automatically populated in the *Cells* field. However, you can update the name of the cell. Also, when multiple Pcell instances are selected, the value `Default` (non-editable) is automatically populated in the *Cells* field.

View displays the view of the non-Pcell instance.

Overwrite Existing Master enables you to overwrite the existing master.

Related Topics

Freezing and Unfreezing Pcell Instances

© 2020 All Rights Reserved.

Guard Ring Form

Guard Ring Template specifies the template to be used for creating the guard ring.

Technology Library cyclic field lets you choose the technology library in which the template you want to use is defined.

Net Name specifies the net with which you want to associate the master path or subpart.

Enclose by specifies the distance between the guard ring and the objects that the guard ring will surround.

Environment variable: grEnclosedBy

Rectangular sets the shape of the guard ring to a rectangle.

Rectilinear allows the shape of the guard ring to be a polygon.

Related Topics

Creating an MPP Guard Ring

© 2020 All Rights Reserved.

Guard Ring Chop Options Form

For information about the Guard Ring Chop Options form, see Guard Ring Chop Options Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Guard Ring Create Tunnel Options Form

For information about the Guard Ring Create Tunnel Options form, see Guard Ring Create Tunnel Options Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Heal Options Form

For information about the Heal Options form, see Heal Options Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Halo Editor Form

Existing Halos displays all halo blockages you have created from the fields at the bottom of the form. In the case displayed above one placement, two routing, and one slot halo have been created and added to the field using the Add/Update button.

Delete deletes all blockages selected in the Existing Halos field.

Add/Update adds any newly created halo. Update changes any selected halo to the offsets entered in the Offsets fields.

Blockage Type selects the type of blockage to create. The choices are: placement, routing, fill, slot, pin, feedthru, or screen.

Environment variable: `blockageType`

Material selects the material on which the blockage is enforced. The choices are: all, other, nwell, pwell, nplus, pplus, ndiff, pdiff, poly, metal, diffusion, li, metal, cut or diffusion. Not all materials are legal for all types of blockages. When Blockage Type is routing all materials are available except cut and li. When Blockage Type is placement the Materiel field is grayed out. Other denotes undefined materials.

Environment variable: `blockageMaterial`

Layer selects the layer associated with the selected material. The layers displayed are those defined in the technology file as having one of the above materials. When Blockage Type is placement the Layer field is grayed out.

Environment variable: `blockageLayer`

Offsets specifies the number of user units the placement halo is offset from the instance master's prBoundary.

> **Top** specifies the number of user units the placement halo is extended from the top of the instance bBox.
>
> Environment variable: `haloTopOffset`
>
> **Bottom** specifies the number of user units the placement halo is extended from the bottom of the instance bBox.
>
> Environment variable: `haloBottomOffset`
>
> **Right** specifies the number of user units the placement halo is extended from the right of the instance bBox.

© 2020 All Rights Reserved.

Environment variable: `haloRightOffset`

Left specifies the number of user units the placement halo is extended from the left of the instance bBox.

Environment variable: `haloLeftOffset`

OK completes the command and closes the form.

Cancel closes the form without executing the command.

Help displays information about the command.

Related Topics

Creating Blockage Halos

© 2020 All Rights Reserved.

Highlight Options Form

Highlight controls the objects highlighted when a net is selected or probed.

Pins specifies that the pins on probed nets are highlighted.
Environment variable: probeNetDisplayCheckPins

Instances specifies that the instances on probed nets are highlighted.
Environment variable: probeNetDisplayCheckInstance

Visible Shapes Only specifies that shapes, pins, and vias on the visible layers are highlighted.
Environment variable: probeVisibleLpp

FlightLines specifies that either all the shapes and opens on the probed net are highlighted (`physical`) or segments linking all pins on the probed net are highlighted (`logical`).
Environment variable: probeNetDisplayFlightLines

Note: Flight lines are visible in VLS XL and higher tiers.

From Selected Objects (Pins, Instances, Shapes) specifies that the nets associated with probed or selected pins, instances, and shapes are highlighted.
Environment variable: probeTermDisplayRadio, probeInstDisplayRadio, probeShapeDisplay

Hierarchy Depth specifies the hierarchy level you want to display for net probing.
Environment variable: probeNetDisplayHierarchyLevel

Dynamic Highlight specifies that any object over which the pointer hovers is probed.
Environment variable: dynamicProbe

Drawing Mode, allows to choose the drawing mode for the next probe. If you choose:

True Color and Highlight Color, the probed shapes are redrawn with their layer-purpose pair and with the highlight color.

Highlight Color Only, the probed shapes are redrawn only with the LPP chosen from Highlight Color list.

True Color Only, the probed shapes are redrawn only with their LPP. The drawing mode with True Color option enables you to visualize the different layers of the connected shapes in the celllview.

Environment variable: trueColorProbe

© 2020 All Rights Reserved.

Video

For a video demonstration of the *True Color* option, see *True Color Probe*.

Tip

To highlight the probed nets, it is recommended that display dimming is enabled. When the drawing mode is *True Color and Highlight Color* or *True Color Only,* the *Enable Dimming* option on the Display Options form is selected automatically. However, you can deselect the *Enable Dimming* option.

Highlight Color shows the highlight drawing layers to be used on the next probe that will be drawn. You can choose *Cyclic* or select the LPP.
Environment variable: probeHighlight

Name Filtering lets you selectively display (or hide) the probes only for the specified list of nets.

Filter net names lets you enable or disable probe filtering by net name.
Environment variable: probeNetNameFilterByName

Filter Mode lets you show or hide the probes for specified nets.
Environment variable: probeNetNameFilterByNameMode

Net Names lets you specify multiple net names and add them to the list. The net names should be seperated by a space. This is a case-sensitive field.
Environment variable: probeNetNameFilterByNameDefaultList

Add Selected Nets lets you add nets to the filter list by selecting the nets in the *Navigator* assistant or from the layout canvas. To remove net names from an already existing filter list, select the net name(s) and click the *Remove From List* button. To remove the entire filter list, click *Remove All*.

Save To Cellview lets you save the current filtering options to the cellview that you can reload later in a different window by clicking the *Load From Cellview* button. You can also delete the saved filtering options from the cellview by clicking the *Delete From Cellview* button.

Note: The *Save To Cellview* option is enabled only when the current cellview is editable.

Related Topics

Probing Nets, Instances, Pins, and Shapes

© 2020 All Rights Reserved.

Install Guard Ring Form

For information about the Install Guard Ring form, see Install Guard Ring Form in the *Virtuoso Fluid Guard Ring User Guide*.

© 2020 All Rights Reserved.

Instance to Mosaic Form

Mode sets the mode for conversion of the selected instances into mosaics. You can choose from *Convert Instances into Mosaics* or *Convert Instances into Single Mosaic*.

> **Convert Instances into Mosaics** is the default and converts a number of selected instances into an equivalent number of 1x1 mosaics.
>
> **Convert Instances into Single Mosaic** converts a number of selected instances of the same master into a single mosaic.

Environment variable: convertEachInstanceIntoMosaic

Related Topics

Converting Instances to Mosaics

© 2020 All Rights Reserved.

Layer Generation Form

Input Layers (the first and third fields) sets the layer or layers you want to modify with *Layer Generation*. You can search for specific layer-purpose pair using the search filter in the first and third fields.

Boolean Operators (second field) combines the selected objects on the input layers.

> **GROW BY** redraws the Layer Generation form so you can enter a resizing value. Enter the number of user units (resizing value) by which to enlarge selected objects on the input layer in the field to the right.

Output Layer (fourth field) sets the layer on which the new shapes will appear. You can search for specific layer-purpose pair using the search filter.

Note: The layer selection using the search filter is available for *Input Layers* and *Output Layer* fields only if the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form.

Related Topics

Changing Objects on a Layer

leHiLayerGen

© 2020 All Rights Reserved.

Layout Editor Options Form

This form comprises the following sections:

- Editor Controls
- Gravity Controls
- Tap
- Snapping
- Wire Editing

Editor Controls

Repeat Commands sets all commands to automatically repeat, if you first select the command and then the object to edit. Does not affect commands if you first select an object, then the command.
Environment variable: modalCommands

Display Reference Point displays a bold asterisk (∗) on the current reference point in your layout.
Environment variable: displayRefPoint

Auto Set Reference Point automatically sets a reference point whenever you enter a new point. You can set the reference point to a particular point by pressing + key on the keypad. The pointer changes to an asterisk (∗). The point you click is the new reference point. You can set the layout editor so that the reference point does not change by selecting *Auto Set Reference Point*. This allows you to place several objects using a single reference point. You use reference points to measure the distance between points you enter.
Environment variable: autoSetRefPoint

Recursion Check turns recursion checking on. Recursion checking prevents the creation of recursive hierarchy (where a cell has an instance of itself at some level) by commands such as *Create Instance*, *Make Cell*, and *Find/Replace*.
Environment variable: recursionCheck

Maintain Connections turns on an automatic re-connection mode for pins, paths, and pathsegs that have been separated during any editing activity that moves pins. The check box is off by default. See Preserving Pin-Path and Pin-Pathseg Connections.
Environment variable: maintainConnections

Abut Server turns automatic abutment and automatic unabutment on and off.

© 2020 All Rights Reserved.

When on, unabutment automatically restores abutted devices when moved apart. This ensures that dropped vias are restored when chained devices are moved apart. It also eliminates errors such as devices with no diffusion overlap on one side.

When off, automatic abutment and automatic unabutment are off.

Also, when set off in the layout editor, unabutment retains the abutted device configuration even when the device is unabutted.
Environment variable: abutServerOn

Select Created Object ensures that the object you create remains selected after you complete the create, copy and paste commands. If you create multiple objects, all of them remain selected after being created. With this option selected, any object(s) that you may have selected before starting the create command remains selected after the create command completes. The option is off by default.

Note: The Repeat commands should be disabled for this option to work for copy and paste commands.

Environment variable: selectCreatedObj

Rotate Around Combined Center enables you to rotate multiple selected objects around the center of the selected bBox. This check box is selected by default.

Environment variable: rotateAroundCombinedCenter

Flip Around Combined Center enables you to flip multiple selected objects around the center of the selected bBox. This check box is selected by default.
Environment variable: flipAroundCombinedCenter

Turn Off Infix in Smart Snapping turns off infix when commands, such as *Create Measurement* and *Quick Align*, that use smart snapping are active.

Environment variable: turnInfixOffWhenSmartSnapping

Use Layer Palette LPPs Only ensures only valid layers displayed in the *Layers* assistant are populated in the layer field of the *Property Editor*, *Move*, *Copy*, *Label,* and *Find/Replace* forms. When this option is selected, you can select an LPP not present in the filtered layers of the Palette but exists in the technology file. The name of the LPP is case-sensitive and should match the name specified in the technology file.

Environment variable: validLppFilterOn

Check Large Coordinates the *Copy* command does not allow objects to go out of the INT_MAX boundary and hence does not distort the copied image. The value of INT_MAX is 2147483647 (2^{31} -1) (signed int max value). When you copy an object, until the copy operation is committed, an image of the copied object is displayed when you drag. This image is referred as the copied image.

© 2020 All Rights Reserved.

Environment variable: checkLargeCoordinates

Palette LPPs Display Order enables you to change the display order of layer-purpose pairs in the canvas. You can do this by dragging up and down the layer-purpose pairs listed in the Palette when Palette mode is set to *Multi-select*.

Environment variable: pteLppsDisplayOrderControl

Display Active LPP in Foreground displays the shapes on the active layer in the foreground, on top of the shapes drawn on all other layers.

Environment variable: pteDisplayActiveLppInForeground

Instance Editing Mode specifies whether the *Flip, Rotate,* and *Create Instance* (*when creating an array of instances*) commands use the instance bounding box (*BBox*), the place and route boundary (*P&R Boundary*), or the *Snap Boundary* as the instance border when operating on instances or mosaics. For example, if you have a standard cell with a large label that extends well beyond the PR boundary, the bounding box of that cell will be much larger than its place and route boundary. Set this option to *P&R Boundary* to have *Flip* and *Rotate* ignore the bounding box and instead use the PR boundary when operating on the cell.

Note: The default is *BBox*. If you set this environment variable to *P&R Boundary* or *Snap Boundary* and the instance or mosaic does not have the corresponding boundary, then *BBox* is used instead.

Environment variable: editInstanceBoundaryMode

Half Width Check sets whether or not you can create or edit a path that has a segment with length less than half the path width. You can either disable the checking (*off*) or enable it (*notify* or *enforce*). If you set it to *off*, a warning message is displayed only if the first and last segments of the path are less than half the path width. If you set the checking to *never*, no warning message is displayed if the length of any segment of the path is less than half the path width.
Environment variable: checkHalfWidthPathSeg

Conic Sides sets the number of segments to use when you convert conics to polygons. The maximum number of sides is 4000.

Environment variable: numSides

Foundry Constraints

Constraint Group specifies the name of the constraint group that contains the foundry constraints.

© 2020 All Rights Reserved.

Note: This field is available only when the `AlternateFoundryCG` environment variable is set to a non-empty string. In case an invalid constraint group is specified in this environment variable, the value in this field is set to `foundry`.

Gravity Controls

Gravity On sets the drawing pointer to automatically snap to objects in the cellview. The pointer snaps to all the visible objects at any hierarchy level.

Environment variable: gravityOn

Distance slider sets the snap distance between the pointer and an object. You can choose from *Small*, *Medium*, *Large*, or *Very Large* using the slider.

Environment variable: gravityDistance

Selectable Objects Only sets the drawing pointer to snap only to the selected objects in the cellview.

Environment variable: gravitySelectableOnly

Tap

Auto Tap

Automatically sets the layer of the wire being created to be the same as that of an existing clicked wire. Attributes of objects are also tapped. In Virtuoso XL and GXL, in addition to object attributes, an existing net name can be tapped. For more information, see Tapping Wires in *Virtuoso Interactive and Assisted Routing User Guide*.

Wire controls whether auto occurs on the first entry point when you use the *Create – Shape – Geometric Wire* (L) or *Create – Wire* (XL, GXL) command. When *Auto* is on, the current entry layer will seed the form and update attributes of the wire you create based on the first point you enter. The command does not seed the *Net Name* field in the *Create Geometric Wire* form in Layout editor. The *Net Name* field is seeded in Layout XL and Layout CE.

Auto Wire is also available from the right-click context menu when the *Create – Shape – Geometric Wire* or *Create – Wire* command is active.
Environment variable: autoTap

Shape controls whether auto occurs on the first entry point when you create rectangles, polygons, paths, circles, donuts, ellipses, and blockages. During the create command, when *Shape Auto* is on, the current entry layer will seed the form and update attributes of the object you create based on the first point you enter. See

© 2020 All Rights Reserved.

Using the Tap Command for more information.
Environment variable: autoTapShape

Tap Layer sets the tapped layer as the layer for the wire or shape being created, and updates the Layer Selection Window. *Layer* option is enabled only when one of *Wire* or *Shape* from *Auto* option is on.

Tap Attributes captures attributes from the wire or shape you click. When tapping a wire or path, you can the LPP, extension type, and width of the wire. The *Attributes* option is enabled only when one of *Wire* or *Shape* from *Auto* option is on.

Select from Overlaps prompts you to choose a layer or net name from the *Choose Object to* form when clicking on a coordinate with objects on overlapping layers. The option is enabled only when one of *Wire* or *Shape* from *Auto* option is on. For more information about tapping LPPs, see Tapping Wires.

Include All (Wire) Via Layers prompts you with the *Choose Object to* form when tapping a via in XL and GXL. For more information, see Tapping Vias.
Environment variable: layerTapPickWireVia

Tap Purpose List prioritizes the purposes considered when changing the current layer. For example, if you a `metal1 pin` layer, but you want to create a wire on `metal1 drawing` layer, you must have the purpose list set to `drawing`. See Using the Tap Section in the Layout Editor Options Form for more information. You can specify an ordered list of layer purposes in the field.

Environment variable: useTapPurposeList, tapPurposeList

Snapping

Snap Pattern Snapping enables snap pattern snapping.
Environment variable: snapPatternSnapping

Instance to Row snaps instances to rows during the commands *Edit – Move*, *Edit – Copy, Edit – Stretch*, and *Create – Instance*. Instances must be associated with a legal site. To set the site definition for an instance master, use the Edit Cellview Properties Form. Ensure that the site definition set for the instance master and the site definition corresponding to the row to which you want to snap the instance are the same.

Instances snap to the nearest site within a row and match the orientation of the site. The instance must be either over a legal row or the distance between the instance and the legal row must be within the specified aperture in *Layout Editor Options* form. Rows can be horizontal or vertical and instances can be any orientation. If the instance has a snap boundary, the snap boundary snaps to the row site. If the instance has a prBoundary but no snap boundary, the prBoundary snaps to the row site. If the instance does not have either a snap boundary or a prBoundary, then the lower left corner of the

© 2020 All Rights Reserved.

instance snaps to the row site. If multiple instances are selected, the instance closest to the reference point will be snapped to the row.
Environment variable: rowSiteEnable

Note: Snapping rows to manufacturing grid is not supported.

Wire Snapping

Wires lets you select the snapping mode as *Manufacturing Grid*, *Routing Grid*, *Off Grid*, *Track Pattern*, and *Snap Pattern*.

Enforce snaps the wire to the centerline of the track pattern. Also, if this option is selected, the wire inherits the width of the active width spacing pattern and the direction in which the wire is created is the same as the direction of the track pattern.
Environment variable: weEnforceDirFromRoutingPattern

Instance Pins Patterns lets you view the patterns available in an instance.
Environment variable: weInstancePinPattern

Wire Editing

This section comprises the Wire and Create Via sub-sections.

Wire

Default Wire Constraint Group cyclic field allows the selection of one of the available constraint groups, which may be defined at the technology or the design level. The constraint group defines the appropriate layers and vias for the application being used. Only the layers, vias, and any rules specified in the selected constraint group are used when creating wires. Selection of a constraint group in the *Wire* cyclic field specifies the application default constraint group for wire creation and editing commands. The default active constraint group is virtuosoDefaultSetup. See Constraint Groups for more information about defining constraint groups.
Environment variable: wireConstraintGroup

Important

Existing wires are not adjusted based on changes to the constraint group in the *Wire Editing* section. The recommended flow is to set the constraint group at the beginning of a session and to not change the constraint group while you are working on a design. To vary the constraints on different nets, set desired constraints on specific nets by using the *Process Rule Editor*.

© 2020 All Rights Reserved.

If you update the constraint group while working on a design, remove the existing results (delete all routing) that would be effected by the constraint group change and re-route.

Via Parameters Calculation Mode sets the calculation mode for via parameters to one of the following:

- *Min Rules & ViaDef*: For all the via parameter fields, such as the number of cut rows and columns, and cut sizes, the minimum rule or the viaDef default values, whichever is greater, are used. If minimum rules are not defined, viaDef defaults are used. The spacing and enclosure values are calculated automatically from the minimum rules. The minimum rules are picked from the constraint group that you select from the *Via* list in the *Create Via* section. The calculated values for the spacing and enclosure fields are never less than the viaDef defaults.

  For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.1` is used as the enclosure value in the *Create Via* form. Similarly, if the `minSpacing` rule is `0.2` and the default cut spacing in viaDef is `0.18`, `0.2` is used as the cut spacing X and Y in the *Create Via* form.

- *Minimum Rules*: For all the via parameter fields, only the minimum rules, such as `minWidth` and `minSpacing`, are used.

  For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.05` is used as the enclosure value in the *Create Via* form. The viaDef default enclosure value is ignored.

- *ViaDef Defaults*: For all the via parameter fields, viaDef defaults are used.

  Note: When *ViaDef Defaults* option is selected, the via engine does not check any rules.

Note: The assisted routing commands, such as *Point to Point Routing*, *Guided Routing*, *Finish Wire*, and *Finish Entire Net*, use the *Via Parameters Calculation Mode* options (*Min Rules & ViaDef*, *Minimum Rules*, and *ViaDef Defaults*) for via creation. For more information on assisted routing commands, see Using Assisted Routing Commands in *Virtuoso Interactive and Assisted Routing User Guide*.

Allow Loops permits creation of loops when creating or editing wires. The option is off by default. When off, the *Create Geometric Wire* command (*Create Geometric Wire* and *Create Wire* commands in XL/GXL) does not allow the creation of loops on the same layer. When the layer is changed while wire creation, loop checking is reset and applies only to the new layer. For information about loop removal, see Removing Loops From Wires.
Environment variable: allowLoops

© 2020 All Rights Reserved.

Adjust Edited Vias Parameters enables the re-calculation of via alignment of the vias if you stretch the wires connected to them. This check box is not selected by default.
Environment variable: adjustEditedViasParams

For information about the options that additionally appear in the form when opened in VLS XL, see *Virtuoso Interactive and Assisted Routing User Guide*.

Create Via

Via specifies the constraint group from which vias will be available when adding vias using *Create – Via* or while editing vias using the *Edit Via Properties* form, the *Property Editor* assistant, or the *Create Via* form. The via spacing values in the constraint group selected here populate the *Create Via* form, as well as reflect on the *Edit Via Properties* form and *Property Editor* assistant. For more information about defining constraint groups, see Constraint Groups in the *Virtuoso Space-based Router User Guide*.
Environment variable: viaConstraintGroup

Note: The spacing on the via does not always match the spacing value defined in the constraints. The via engine tries both an odd and an even spacing to avoid asymmetric enclosures.

Via Parameters Calculation Mode seeds the *Create Via* form the first time when a specified viaDef is used. Then onwards, the *Create Via* form is populated from Parameter Caching. If required, you can later reset the calculation mode by using the *Reset Parameters to* list on the Create Via Form. The cyclic field values for this option include:

- *Min Rules & ViaDef*: For all the via parameter fields, such as the number of cut rows and columns, and cut sizes, the minimum rule or the viaDef default values, whichever is greater, are used. The spacing and enclosure values are calculated automatically from the minimum rules. The minimum rules are picked from the constraint group that you select from the *Via* list in the *Create Via* section. The calculated values for the spacing and enclosure fields are never less than the viaDef defaults.

  For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.1` is used as the enclosure value in the *Create Via* form. Similarly, if the `minSpacing` rule is `0.2` and the default cut spacing in viaDef is `0.18`, `0.2` is used as the cut spacing X and Y in the *Create Via* form.

- *Minimum Rules*: For all the via parameter fields, only the minimum rules, such as `minWidth` and `minSpacing`, are used.

© 2020 All Rights Reserved.

For example, if the `minExtensionDistance` rule is `0.05` and the default enclosure in viaDef is `0.1`, `0.05` is used as the enclosure value in the *Create Via* form. The viaDef default enclosure value is ignored.

- *ViaDef Defaults*: For all the via parameter fields, viaDef defaults are used.

File specifies the file to which you want to store or from which you want to load the settings. You can enter the correct directory where you want to save the file. If you do not remember the name of the directory use the *Browse* button.

Environment variable: editorOptionsFile

Save To saves the current setting from the *Layout Editor Options* form to the file specified in the *File* field.

Load From sets the *Layout Editor Options* form settings to the values saved in the specified file.

Preserving Pin-Path and Pin-Pathseg Connections

To set the layout editor to automatically reconnect paths to pin and pathseg to pin connections that are broken during any pin editing activity, do one of the following:

- In the Layout Editor Options Form, check *Maintain Connections*.
- Press `F6` to toggle *Maintain Connections* on and off.

A message in the CIW lets you know the status of *Maintain Connections*.

Maintain Connections is a background process that monitors the connectivity of pins that abut or overlap path or pathseg ends on the same layer. When any editing activity causes the pin to move away from the path or pathseg end, *Maintain Connections* automatically modifies the path or pathseg so that it remains connected to the pin.

© 2020 All Rights Reserved.

Maintain Connections searches all instances in the top cellview to find all the pins, then finds all the overlapping paths and pathsegs associated with each pin to check for broken connections. If the top-level design contains many instances and each instance contains many pins, opening a design can become more time consuming. When you open large designs, you should set *Maintain Connections* to off.

Editing activities that can cause a pin to change location include

- Moving or stretching a pin
- Moving or stretching an instance containing a pin
- Editing pin properties (changing the pin shape or pin location)
- Changing Pcell parameters

The *Maintain Connections* option operates on pins, paths, and pathsegs with the following characteristics:

- The pin and path or pathseg are on the same or equivalent layer and have the same layer purpose.
- The path or pathseg end overlaps or abuts the pin.
- The pin is rectangular.
- If the pin is an instance, it must be placed at the same level as the path or pathseg.
- If the pin and path or pathseg are ROD objects, they cannot be aligned.

One way to see the effects of *Maintain Connections* is to change the parameters of an instance that contains pin-path or pin-pathseg connections on the same layer. After you change the parameters, the instance changes shape to the new value, and the path or pathseg and pin retain their connectivity.

1. Select a Pcell containing pin-path or pin-pathseg connections.

© 2020 All Rights Reserved.

2. Choose *Edit – Basic – Properties*.
3. Click *Parameter* and change the size of the Pcell.
4. Click *Apply*.

   The instance changes shape to the new width value, and the paths/pathsegs reconnect to the pins.

Limitations

Maintain Connections does not retain connectivity in the following relationships:

- Pin-to-pin connections

© 2020 All Rights Reserved.

- Pins placed in the middle of paths

Pins A and C remain connected to the path, but pin B might not.

- Aligned ROD objects

  ROD alignments take precedence over maintaining connections.

  Before a move with *Maintain Connections*:

The pin in instance IB is aligned to the ROD path.

  After a move with *Maintain Connections*:

Moving instance `IB` causes the whole ROD path to move. The aligned path does not stretch, so moving instance `IB` breaks the electrical connection between the two pins.

Related Topics

Setting Layout Editor Options

leHiEditEditorOptions

© 2020 All Rights Reserved.

Load Multipart Path Template Form

Template File allows you to load ASCII template files.

Related Topics

Loading an ASCII Multipart Path Template File

© 2020 All Rights Reserved.

Magnifier Options Form

Anchored at Cursor Position fixes the magnifier at its last position. The magnifier does not move along with the pointer. You can move the pointer and the design under and around the pointer is magnified in the magnifier. This option is always grayed out and cannot be set by using the form. Use the hash bindkey (`#`) to select this option. Even when selected, the option remains grayed out. The `#` bindkey toggles between the *Anchored at Cursor Position* and *Not Anchored* options.
Environment variable: magDisplayMode

Anchored at specifies where the magnifier should be positioned with respect to the pointer position. By default, the magnifier appears at the *top left corner* of the design display area with respect to the pointer. The available choices include *top right corner*, *bottom left corner*, and *bottom right corner*. If you select *Anchored at*, the *Not Anchored* option is turned off.
Environment variable: magDisplayMode, magAnchorPosition

Not Anchored enables you to customize the position of the magnifier either to the center of the area being magnified or as an offset value from the pointer position. If you select *Not Anchored*, the *Anchored at* option is turned off.
Environment variable: magDisplayMode, magFloatPosition

> **Center** positions the magnifier such that its center point coincides the pointer position.
>
> **Position Magnifier** enables you to specify the magnifier position with respect to the pointer position. The available choices for positioning the magnifier with respect to the pointer position include *top left*, *top right*, *bottom right*, and *bottom left*.
> Environment variable: magOffsetCorner
>
> **offset by** specifies the distance of the magnifier from the pointer position. You can set any value from `0` through `100`. The default value is `15`. For example, with the default *top left* setting and offset `15`, the magnifier is positioned at the top left side of the pointer and offset both on X and Y axis by a distance of 15 units. If you set the *offset by* to `0`, one of the vertices of the magnifier coincide the pointer position.
> Environment variable: magCursorOffset

Magnifier Size enables you to set the pixel size of the square magnifier window. You can set any value from `50` through `250`. The default value is `200`.
Environment variable: magSize

Keep Within Window enables you to display the magnifier in the design display area. This check box is selected by default. If the check box is not selected, the magnifier might not display completely based on the other settings in the form.
Environment variable: magConfined

© 2020 All Rights Reserved.

In the example below, *Position Magnifier* is set to *bottom left* and *offset by* is `15`.

- If the *Keep Within Window* check box is not selected, the magnifier is partially visible.

- If the *Keep Within Window* check box is selected, the magnifier is completely visible.

Save as Defaults saves the specified magnifier settings in the `.cdsenv` file for the current root application. You can retrieve the updated default values by using the *Get Defaults* button.

Get Defaults retrieves the default values saved in the `.cdsenv` file for the current root application and resets the values updated, if any, in the form.

© 2020 All Rights Reserved.

Related Topics

Using the Magnifier

© 2020 All Rights Reserved.

Make Cell Form

Library sets the library in which to create the new cellview.

Cell sets the name of the new cellview.

View sets the view name of the new cellview.

Type lets you choose the type of cell you are going create. The valid choices are *none* (which lets you choose an unspecified object), *block*, *blockRing*, *cover*, *coverBump*, *pad*, *padSpacer*, *core*, *coreSpacer*, *coreAntenna*, *corner*, *softMacro*, *via*, *blockBlackBox*, *padAreaIO*.
Environment variable: makeCellCellType

Replace selected figures replaces the selected objects in the cellview with an instance of the new cell.

Origin

Origin drop-down list lets you select the origin point. You can select from *LowerLeft*, *LowerRight*, *UpperLeft*, *UpperRight*, or *Custom*. When you select *Custom*, the *X, Y* and *Set by cursor* fields are enabled.

X, Y sets an X and Y origin point for the new cell when you enter the points in the *X, Y* fields.

Set by cursor sets an X and Y origin point for the new cell when you enter a new point in the cellview with the pointer.

Hierarchy

Transparent instance generates made cell instances that are transparent. This means the binder can recognize the instances and nets within a made cell instance and can bind them in a one-to-one correspondence with the schematic instances.
Environment variable: makeCellTransparent

Preserve connectivity retains the correct logical connectivity on the made cell.
Environment variable: makeCellPreserveConn

Create pins creates pins for nets that are external to the newly made cell, if the nets do not already have a pin created.
Environment variable: makeCellCreatePins

Pins below boundary positions the created pins outside the design's boundary.
Environment variable: makeCellPinsBelowBoundary

© 2020 All Rights Reserved.

P&R Boundary generates a rectangular or a polygonal place and route boundary for the made cell.

- If the selected PR boundary shape is *Rectangle*, layout editor automatically calculates the PR boundary dimensions and creates the boundary.
- If the selected PR boundary shape is *Polygon*, the layout editor opens the new made cell in the same window with the *Create – P&R Objects – P&R Boundary* (*Polygon*) command selected. You can manually draw the PR boundary on the canvas and the design automatically returns to the top level when you finish creating the polygon.

Environment variable: makeCellPRBoundary

Cluster

Cluster drop-down list connects or places the selected objects in or with a cluster or instance/cluster boundary. The cellview cannot already contain a cluster or instance/cluster boundary with the same name as the one you have selected. The association between the cluster and the instances/cluster boundary is retained.

Type is a read-only field that displays what type of cluster you have selected in the Cluster Name cyclic Field inclusive, exclusive, or suggested.

Related Topics

Making Cells

Generating Transparent Layout Hierarchy

leHiMakeCell

© 2020 All Rights Reserved.

Make Readonly/Editable Form

The left pane of the form displays a tree-like list of the master cellviews whose instances exist in the current cellview. It displays the library/cell/view information of these master cellviews. The last column in the pane displays the mode of the listed master cellviews.

Note: The *Make Readonly/Editable* form does not list the masters of the Pcell instances in the cellview.

Make ReadOnly changes the mode of the selected master cellview to read-only.

Make Editable changes the mode of the selected master cellview to editable.

Filter section specifies a criteria for listing only specific master cellviews in the left pane.

Use searches for master cellviews of all or of the instances selected in the cellview.

masters displays in the left pane master cellviews of all the instances in the cellview.

masters in the selected set displays in the left pane master cellviews of only the instances selected in the cellview.

Name specifies a name pattern to use while searching for the master cellviews.

condition/logic cyclic field == or **!=** lets you search for character strings or numerals that are an exact match (==) or are not a match (!=); to get matches that contain a string, a leading and/or trailing asterisk must be used.

value field sets the value of the name of the master cellview to use in the search.

Type specifies the type of the master cellview to use in the search.

condition/logic cyclic field == or **!=** lets you search for character strings or numerals that are an exact match (==) or are not a match (!=); to get matches that contain a string, a leading and/or trailing asterisk must be used.

cyclic field lets you search for specific type of master cellview, such as a *blockRing* or a *softMacro*. You can also select to search based on *all* types of master cellviews or based on *none* of the types.

Filter searches for master cellviews matching the criteria specified in the *Filter* section and displays the result in the left pane**.**

Related Topics

Making Hierarchy Editable

© 2020 All Rights Reserved.

MarkNet Options Form

The form comprises the following tabs:

- Options Tab
- Via Layers Tab
- Connected Layers Tab
- Stop Layers Tab

Options Tab

Hierarchy Range sets how many levels of the hierarchy are highlighted when a net is selected.
Environment variable: markNetRange

Top to bottom highlights the selected net through the entire hierarchy of the cellview.

Current to bottom highlights the net from the current level of hierarchy to the level specified in the *Display Level To* field in the Display Options Form (stop level).

Top to current highlights the net from the level specified in the *Display Level From* field in the Display Options Form (start level) to the current level of hierarchy.

Top to stop level highlights the net from the top level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options Form (stop level).

Current to stop level highlights the net from the current level in the hierarchy to the level specified in the *Display Levels Stop* field in the Display Options Form (stop level).

User defined makes the *Mark Net Start Level* and *Mark Net Stop Level* fields editable.

Start sets the hierarchy level at which the highlighting starts. This field is editable only when *Mark Net Hierarchy Range* is set to *User defined*.
Environment variable: markNetStartLevel

Stop sets the level at which the highlighting stops. This field is editable only when *Mark Net Hierarchy Range* is set to *User defined*.
Environment variable: markNetStopLevel

Region

© 2020 All Rights Reserved.

Entire CellView traces nets in the entire cellview. This option is selected by default.

Viewing Area traces nets in the portion of the cellview that is displayed in the canvas at that instant.

Draw a Rectangle Region on the Canvas or **Draw a Polygon Region on the Canvas** enables you to specify a rectangular or polygonal area for tracing nets. Select the required option, and then click in the canvas to specify the vertices of the rectangle or polygon.

Auto Highlight Color marks up to ten nets with a different color by automatically cycling through Cadence-defined layer purpose pairs y0 drawing through y9 drawing. If the appearance of these layer purpose pairs is not what you want, you can change them using the *Display Resource Editor*.
Environment variable: markNetAutoColorCycle

Highlight Color enables you to manually change the color used to highlight the nets. You can select one of the Cadence-defined layer purpose pair y0 drawing through y9 drawing. If the appearance of these layer purpose pairs is not what you want, you can change them using the *Display Resource Editor*.

Thick Line enables you to display the traced nets in bold.
Environment variable: markNetThickline

Highlight Incrementally displays the traced connectivity in a progressive manner. When this check box is selected, the connectivity is highlighted after processing every 10,000 shapes.

Highlight Individual Shapes adds a separate highlight for each shape.

With merged highlights

With separate highlights

Retain Via Info for Save MarkNets enables you to save via shapes when you later run the *Connectivity – Nets – Save All Mark Nets* command. The via shapes are retained in the cellview created for storing the traced shapes. This option is not selected by default. Therefore, by default, the via shapes are not included in the separate cellview generated by the *Save All Mark Nets* command.

© 2020 All Rights Reserved.

The “Number of shapes processed by Mark Net”, as reported in the CIW, is not impacted by this check box setting.
Environment variable: markNetSaveViaInfo

Select From Overlaps enables you to select the LPP from where the *Mark Net* command starts tracing the net when multiple overlapping shapes are selected. If this check box is selected, clicking overlapping shapes opens the *Choose LPP to Mark* form in which you can select the required LPP. If the check box is not selected, the *Mark Net* command starts tracing the net from any one of the overlapping shapes.
Environment variable: markNetSelectOverlapLayer

Customized Connected Layers, **Customized Via Layers**, and **Enable Stop Layers** enable the settings defined on the Via Layers Tab, Connected Layers Tab, and Stop Layers Tab abs to be used for tracing a net. These check boxes are selected by default. If the *Customized Via Layers* check box is not selected, the via layers defined in the technology are used. If the *Customized Connected Layers* or the *Enable Stop Layers* check box is not selected, the connected layers or stop layers are not used.
Environment variables: markNetViaLayer, markNetConnectedLayer, markNetStopLayer

Exclude Purposes enables you to exclude the specified purposes from tracing. The default purpose values that are listed are *annotate*, *boundary*, *error*, *flight*, *label*, and *warning*. You can delete any of the default purpose values. This check box is selected by default.
Environment variables: markNetExcludePurposes,

Save/Load custom settings for Via, Connected and Stop Layers

Browse opens the UNIX Browser window. Use this window to specify the name and location of the file to which you want to save the layer settings. You can also use this window to locate an existing file containing customized layer settings. The path and file name that you specify are displayed in the *Browse* field. If a `markNetOptions` file has been specified using the markNetOptionsFile environment variable, the settings for the via, connected, and stop layers are read from this file at the start of a new Virtuoso session.

Save To saves the layer settings to the file specified in the *Browse* field.

Load From populates the *Via Layers*, *Connected Layers*, and *Stop Layers* tabs with the layer settings stored in the file specified in the *Browse* field.

Via Layers enables you to save or load via layer settings to or from the file specified in the *Browse* field.

Connected Layers enables you to save or load connected layer settings to or from the file specified in the *Browse* field.

© 2020 All Rights Reserved.

Stop Layers enables you to save or load via layer settings to or from the file specified in the *Browse* field.

Exclude Purposes enables you to save or load exclude purposes settings to or from the file specified in the *Browse* field.

By default, settings for all the three layers — via, connected, and stop — are saved and loaded to or from the file specified in the *Browse* field.

Via Layers Tab

On this tab, you define via layers in the (*layer1 viaLayer layer2*) format, where each component layer can be specified as an LPP. If the *Customized Via Layers* check box on the Options Tab tab is not selected, the *Via Layers* tab is disabled and the via layers defined in the technology file are used.

Via Layers list box displays all via layers — those defined by you and ones that have been picked from the technology file.

Enable lets you select the via layers that you want to use for tracing a net. To select all the via layers, select the check box at the top of the column. To select individual via layers, select the check box against each required via layer.

Delete lets you delete via layers that are invalid or not required. To delete all the via layers, select the check box at the top of the column and click *Delete*. To delete individual via layers, select the check box against a via layer and click *Delete*.

Caution

You cannot undo the deletion.

Add Via Layers lets you define the via layers in the (*layer1 viaLayer layer2*) format. For each layer, specify the layer name in the *layer* column and the purpose name in the *purpose* column. It is optional to specify the purpose name.

Add displays the newly added via layer in the (*layer1 viaLayer layer2*) format in the *Via Layers* list box.

Reset to TechDB Defaults lets you use the via layers defined in the technology file. Clicking this button populates the list box with the via layers from the technology file and deletes any via layers that you may have added.

© 2020 All Rights Reserved.

Connected Layers Tab

On this tab, you define connected layers in the *(layer1 purpose) (layer2 purpose)* format. The connected layers map to the equivalent layers defined in the technology file. If the *Customized Connected Layers* check box is not selected on the Options Tab tab, the *Connected Layers* tab is disabled and connected layers are not used for tracing.

Connected Layers list box displays all connected layers — those defined by you and ones that have been picked from the technology file.

> **Enable** lets you select the connected layers that you want to use for tracing a net. To select all the connected layers, select the check box at the top of the column. To select individual connected layers, select the check box against each required connected layer.
>
> **Delete** lets you delete connected layers that are invalid or not required. To delete all the connected layers, select the check box at the top of the column and click *Delete*. To delete individual connected layers, select the check box against a connected layer and click *Delete*.

Caution

You cannot undo the deletion.

Add More Entries displays a text field in the *Connected Layers* list box. In this field, you specify the values in the *(layer1 purpose) (layer2 purpose)* format.

Reset to TechDB Defaults lets you use the connected layers defined in the technology file. Clicking this button populates the list box with the connected layers from the technology file and deletes any connected layers that you may have added.

Stop Layers Tab

On this tab, you define the layers on which the *Mark Net* command must stop tracing a net. The stop criterion is specified in the *layer stopLayer* format, where both layers can be specified as LPPs. The *Mark Net* command stops tracing the connected shapes on the *layer* LPP if it finds an overlapping shape on the *stopLayer* LPP. If the *Enable Stop Layers* check box is not selected on the Options Tab tab, the *Stop Layers* tab is disabled and the *Mark Net* command highlights all the connected shapes.

layer [purpose] stopLayer [purpose] list box displays the stop layers that you have defined.

layer [purpose] [color] [lock] stopLayer [purpose] list box displays the stop layers that you have defined.

© 2020 All Rights Reserved.

Enable lets you select the stop layers that you want to use for tracing a net. To select all the stop layers, select the check box at the top of the column. To select individual stop layers, select the check box against each required stop layer.

Delete lets you delete a stop layer that is invalid or not required. To delete all the stop layers, select the check box at the top of the column and click *Delete*. To delete individual stop layers, select the check box against a stop layer and click *Delete*.

Caution

You cannot undo the deletion.

Add Stop Layers enables you to define the stop layers in the (*layer stopLayer*) format. For each layer, specify the layer name in the *layer* column and the purpose name in the *purpose* column. The purpose name is optional. Appropriate messages are generated if you try to add duplicate entries for a stop layer or specify the same value for *layer* and *stopLayer*.

Add Stop Layers enables you to define the stop layers in the (*layer stopLayer*) format. For each layer, specify the layer name in the *layer* column, the purpose name in the *purpose* column, select the color ('*any*', '*gray*', '*mask1*' or '*mask2*' ....'*maskn*') in the *color* column, select the lock state ('*any*', '*yes*', or '*no*') in the *lock* column. The purpose name is optional. Appropriate messages are generated if you try to add duplicate entries for a stop layer or specify the same value for *layer* and *stopLayer*.

Note: To stop tracing a net at a particular via layer, Cadence recommends that you remove or disable that via layer entry on the Via Layers Tab tab, instead of defining it as a stop layer on the *Stop Layers* tab. Having the entry only on one of the tabs prevents ambiguity and undesirable results.

Add displays the newly added stop layer in the *layer [purpose] stopLayer [purpose]* list box.

Related Topics

Tracing Nets Using the Mark Net Command

© 2020 All Rights Reserved.

Marker Text Window

Location gives the library, cell, and cellview names.

Reason gives the reason associated with the marker. This information relates to the specific technology file rules that have been broken or connectivity errors.

Related Topics

Explaining Markers

© 2020 All Rights Reserved.

Modify Corner Form

Type of Corner controls whether you create a rounded corner (radial) or a 45-degree edge (chamfer). If you choose to create a chamfer corner, the *Radius* field changes to *Distance*.

Radius sets the radius of the rounded corner, in user units. This field is available if you set the *Type of Corner* to *radial*.

Distance sets the distance in user units from the vertex to begin beveling. This field is available if you set the *Type of Corner* to *chamfer*.

Number of Sides sets the number of straight segments used to create a rounded corner. This setting is used only if *Type of Corner* is set to *radial*.

Snap to manufacturing grid the geometry is snapped to the manufacturing grid.

Optimize sides for smooth curve the layout editor will optimize the number of sides to get a smooth curve for the corner. This option is enabled only if the *Snap to manufacturing grid* option is selected.

Related Topics

Modifying Corners

leHiModifyCorner

© 2020 All Rights Reserved.

Move Form

Snap Mode controls the direction in which you can move the object.
Environment variable: snapMode

Change To Layer lets you move an object to another layer. You can search for specific layer-purpose pair using the search filter in the *Change To Layer* list box.

Treat as absolute point the points specified in Delta X,Y fields will be considered as absolute points.

Delta X, **Y** lets you specify the number of user units to move the point of origin of an object on the X and/or Y axes.

Apply moves the origin point of the selected object as specified in the Delta X and Delta Y fields.

Display Draglines controls whether draglines are displayed during the *Move and Stretch* commands. By default, the draglines indicate connections from the pins of the object you are moving to pins of the nearest objects.
Environment variable: flightLineEnable

Connections

Chain Mode enables you to move device chains.

All: Moves all instances in a chain.

Selected: Moves only selected instances.

Selected Plus Left: Moves the instance selected and all instances to the left of the selected instance in the chain.

Selected Plus Right: Moves the instance selected and all instances to the right of the selected instance in the chain.

Click the buttons at the bottom to change the orientation of the selected instance.

Rotate rotates the component 90 degrees counterclockwise and clockwise.

Sideways mirrors the component on the y axis (flips it horizontally).

Upside Down mirrors the component on the x axis (flips it vertically).

(ICADVM20.1 Virtuoso RF Solution Only) In the Virtuoso RF mode, the Move form includes an additional *Wire Bond* section. The options in this section let you control the move

© 2020 All Rights Reserved.

behavior of bond wires and bond fingers. For more information, see Editing Bond Wires and Bond Fingers.

Snap

Pins and boundary to grid snaps the moved object to a grid depending on the block type. If it is a digital block, the place and route boundary is snapped to the placement grid and pins with placement status `placed` are snapped to the routing grid. If it is an analog block, both place and route boundary and `placed` pins are snapped to the manufacturing grid. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not snapped, even if they were touching the boundary before the move.

Pins to boundary automatically snaps pins to the place and route boundary.

Allow pin resizing specifies that the pins need to be resized during pin snapping.

Environment Variable: allowPinResizing

Note: The *Allow Pin Resizing* setting that you make in the *Move* form is applicable only for the current instance. For all other instances, the *Allow Pin Resizing* setting in the *Floorplan Global Options* form is considered as the default setting. Changes that you make to the *Allow Pin Resizing* setting in the *Move* form cannot overwrite the global setting.

For more information about pin snapping, see Snapping Soft Block Pins to Grid.

Tip

When constraint-aware editing is on, the *Move* command honors the following physical constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*.

- For more information on Constraint-Aware Editing mode, see Constraint-Aware Editing.
- For detailed definitions of the supported constraints, see Default Constraint Types in the *Virtuoso Unified Custom Constraints User Guide*.

Related Topics

Moving Objects

leHiMove

VLS XL

© 2020 All Rights Reserved.

Moving Objects

Moving Objects Using the Move Form

© 2020 All Rights Reserved.

Pad Opening Info Form

Use the Pad Opening Info form to search for bond pad shapes in a hierarchical cellview, generate reports and labels for the pad shapes that are found, and promote the generated labels to the top level of the cellview.

The form offers three modes to support the main tasks you can perform: *Generate Report*, *Generate Labels*, and *Promote Labels To Top*.

The options available depend on the task you are performing. The table below summarizes which options are available in which modes.

Options/Mode	Generate Report	Generate Labels	Promote Labels
Search Criteria	X	X	X
Sort	X	X	
Reporting	X		
Label Control		X	
Promote Label			X

Delete Bond Pad Labels is available in all modes and deletes any existing bond pad labels at the top level of the current cellview. Only bond pad labels generated by the *Pad Opening Info* command are deleted; labels generated by other tools or commands are not affected.

Search Criteria

The *Search Criteria* options let you specify the layer-purpose pair and hierarchy levels to be searched for pad shapes, and the dimensions of the pad shapes you are interested in. You can also search for existing labels in the design and limit the search to specific regions of the cellview.

Bond Pad

Layer/Purpose specifies the layer-purpose pair to search for bond pad shapes. The default is `"Bondpad drawing"`. You can search for specific layer-purpose pair using the search filter in the *Layer/Purpose* list box.

Note: The layer selection using the search filter is available only if the *Use Layer Palette LPPs Only* option is selected in the Layout Editor Options form.
Environment variable: bondPadLPP

© 2020 All Rights Reserved.

Search Depth specifies the number of hierarchical levels to traverse when searching for pad shapes. For example, if you specify 5, the command searches for shapes on the specified layer-purpose pair at hierarchy levels 0 through 5. The maximum value that you can specify in this field is `32`.

The **Dimension** fields let you refine the search further by specifying the dimensions of the pads you want to label. The X dimension is required and defaults to `>` and `0`. You can optionally specify an operator and value for the Y dimension as well. The command returns only shapes with a bounding box that meets the specified criteria.

> **Note:** For non-rectangular (e.g., octagonal) pads, use the width of the enclosing bounding box as the pad dimension.

Region

Specify Search Regions lets you narrow the search area to specific regions of the cellview.

Clear All clears any search regions currently defined.

Add Region adds another region definition to the list.

Create Rect lets you click and drag in the layout canvas to draw a rectangular search region. The corresponding coordinates are automatically entered in the *Region* field. You can edit these coordinates manually if required.

Create Polygon lets you draw a polygonal search region by clicking on each point of the polygon in the layout canvas. To finish a polygon, either double-click in the canvas or click and press `Return`. The corresponding coordinates are automatically entered in the *Region* field. You can edit these coordinates manually if required.

Label/Name lets you specify the layer and hierarchy level to search for existing labels.

Object Type lets you select the type of object you want to search, *Label*, *Text Display*, or *Both*.

Layer/Purpose specifies the layer-purpose pair to search for existing labels.

Search Depth sets the hierarchy level on which to search for existing labels.

> **Note:** The command returns only the labels found at the hierarchy level you specify. For example, if you set the **Bond Pad Search Depth** to 5 and the **Label/Name Search Depth** to 2, then the command shows all the pad shapes found in levels 0 through 5, but returns only the text labels found at level 2. Existing labels on other levels are not displayed.

Highlight draws the pad shapes found in the appropriate highlight color in the canvas.

© 2020 All Rights Reserved.

Sort

The *Sort* options let you specify the location for the first pad label to be generated and the direction in which subsequent pads are labeled.

1st PAD Coords sets the location for the first label generated by the command. Click the button and then the required location in the canvas to enter the corresponding coordinates in the field. You can also edit the coordinates manually using the following format: `(1031.975 877.84)`.

Sort Direction specifies the direction to use when extracting pad locations. Choose between *Clockwise* and *Counter Clockwise*.

Reporting

The *Reporting* options are shown only in *Generate Report* mode and let you specify what reports are generated by the command. If the predefined standard report format does not meet your requirements, you can customize reports as required.

Make Standard Report generates a report in standard format. Check the box and use the *Browse* button to specify the location and name for the report file. See Generating a Standard Report for more information.

Make Customized Report lets you define and generate reports that are customized to your requirements in terms of content and filtering criteria. See Generating a Customized Report.

> **Add** opens the Create Encore Report Form where you define the format of your customized report. You can add as many customized reports as required.
>
> **Del** removes the selected customized report from the list.
>
> **Edit** opens the Create Encore Report Form where you can modify the selected customized report.

Label Control

The *Label Control* options are displayed only in Generate Labels mode. They let you specify what label names are generated and how they will be drawn in the canvas. For more information on generating labels, see Generating Labels for Pads.

Pad Name

> **Read Label**

© 2020 All Rights Reserved.

Label (Pattern) specifies a naming convention to be used when generating labels. The label pattern feature supports standard bus label notation as illustrated by the examples below.

Label Pattern	Labels Generated
`A<5:2>`	`A5, A4, A3, A2`
`B<0:6:2>`	`B0, B2, B4, B6`
`A<1:2>B<1:2>`	`A1B1, A1B2, A2B1, A2B2`
`A<1:2>C`	`A1C, A2C`
`A[0:1]-B[0:1]`	`A0, B0, A1, B1`
`A<<0:1>>`	`A<0>, A<1>`
`A(<0:2>)`	`A(<0>), A(<1>), A(<2>)`
`A([0:2])`	`A(0), A(1), A(2)`
`A{<0:2>}`	`A{<0>}, A{<1>}, A{<2>}`
`A{[0:2]}`	`A{0}, A{1}, A{2}`
`A([0:6:2])C`	`A(0)C, A(2)C, A(4)C, A(6)C`

From File reads the label names from a file. Enter the path to the file in the field provided, or use the *Browse* button to locate it on your system.

Label Options

Font specifies the font used to draw the labels on the canvas.

Height sets the height of the label text.

Justification specifies the alignment of the label in relation to the pad shape (for example, *lowerLeft*, *centerCenter*, *upperRight*, etc).

Layer/Purpose specifies the layer-purpose pair on which the label name is drawn. The default is `"text drawing"`.
Environment variable: padLabelLPP

Promote Label

The Promote Label options are shown only in Promote Label To Top mode. They let you specify the appearance of the promoted labels in the canvas and control whether the

© 2020 All Rights Reserved.

corresponding lower-level labels are removed from the cellview. For more information, see Promoting Labels to the Top Level.

Label Options

Font specifies the font used to draw the labels on the canvas.

Height sets the height of the label text.

Justification specifies the label alignment in relation to the pad shape (for example, *lowerLeft*, *centerCenter*, *upperRight*, etc).

Layer/Purpose specifies the layer-purpose pair on which the label is drawn. The default is `"text drawing"`.
Environment variable: padLabelLPP

Remove Original Labels deletes the original labels at the lower level after they have been promoted to the top level.

Related Topics

Generating a Standard Report

Generating a Customized Report

Generating Labels for Pads

Promoting Labels to the Top Level

leHiDisplayPadOpeningInfoForm

© 2020 All Rights Reserved.

Palette Options Form

Keep windows synchronized initializes all new windows to synchronized state.

Environment Variable: pteWindowSynchronizationMode

Search Mode enables you to switch between *Filter* and *Find* search modes.

Environment Variable: pteFindModeOn

Multi-select mode lets you select multiple layer-purpose pairs at one time. Only one layer—the one selected first—is set as active.

Environment Variable: pteMultiSelectionMode

Auto Redraw sets the scope of the auto redraw feature to *Off*, *Current window*, or *All windows*.

Environment Variable: pteAutoRedrawMode

Layer Set Activation specifies how visibility and selectability status of layer-purpose pairs is affected when a layer set is made active. The available options are *Filter only*, *Filter and apply visibility, selectability*, and *Force visibility and selectability only*.

Environment Variable: pteLayerSetActivationMode

Scope of 'All Layers' Layer Set defines the scope of the layer-purpose pairs displayed in the *Layers* panel. You can set the scope to either *LSW layers* or *Techfile layers*.

Environment Variable: pteAllLayersScope

Header Column Click specifies whether the visibility, selectability, validity, and membership status of layer-purpose pairs is turned on or off or sorted when you click the respective column header.

Environment Variable: pteCheckableColumnClickMode

Scope of Used Layers in EIP specifies the layers to display in the *Layers* panel when EIP mode is active. The default is *Current to bottom*; it displays layers used in the design at the current hierarchy level and below. When set to *Top to bottom*, all layers used in the design are listed in the *Layers* panel.

If different technology files are associated with the cellviews at different hierarchical levels in the design, only the layers defined in the technology file associated with the cellview being edited in place are displayed in the *Layers* panel.

© 2020 All Rights Reserved.

Environment Variable: pteUsedLayersEipScope

Depth for Visibility determines the number of layer-purpose pairs for which the visibility attribute is affected when you hold down `Ctrl` or both `Ctrl` and `Shift` and click in the *Layers* panel a layer-purpose pair with the middle mouse button.

Environment Variable: pteLayersVisibilityDepth

Depth for Selectability determines the number of layer-purpose pairs for which the selectability attribute is affected when you hold down `Ctrl` or both `Ctrl` and `Shift` and right-click a layer-purpose pair in the *Layers* panel.

Environment Variable: pteLayersSelectabilityDepth

Sync visibility/selectability by parent when selected propagates changes in the visibility and selectability status of parent objects and grids to child objects and grids.

Environment Variable: pteApplyParentState

Shape visibility by color visibility when selected makes colored shapes invisible if you deselect the Visibility (*V*) check box corresponding to the layer-purpose pairs on which they are drawn. If deselected, only the colored outline is hidden; the shapes remain visible.

Environment Variable: pteLinkColorVisibilityToShapeVisibility

Color mixed sort when selected sorts all layer-purpose pairs, colored and uncolored, together and lists them in the order defined by the specified attribute.

Environment Variable: pteColorMixedSort

Compact MPT mode when selected hides the Color (*C*) column and displays on the *Active* (*Layer*) toolbar a set of options that you can use to control the color and lock state combination for a layer-purpose pair and also the visibility of colored shapes based on their lock state.

Environment Variable: pteCompactMPT

Inverse color scheme specifies whether the color scheme of the Palette panels needs to be inversed.

Environment Variable: pteInverseColorScheme

Movable tools specifies whether the toolbars can be moved in the workspace.

Environment Variable: pteMovableTools

© 2020 All Rights Reserved.

Paste Form

Reference Points lets you select the reference point of the bounding box. You can select one of the ten reference points when pasting. The reference points are *lowerLeft*, *centerLeft*, *upperLeft*, *lowerCenter*, *centerCenter, upperCenter*, *lowerRight*, *centerRight*, *upperRight*, *origin*, *user specified*.

Environment Variable: pasteRef

Rotate turns the object 90 degrees counterclockwise. You can also click the middle mouse button to rotate or mirror objects you paste.

Sideways mirrors the object along the X axis.

Upside Down mirrors the object along the Y axis.

Related Topics

Yanking and Pasting Objects

leHiPaste

© 2020 All Rights Reserved.

Print Hierarchy Tree Form

Library lets you select the name of the library for which you want to print the hierarchy of cellview instances.

Cell lets you type, or select from the *Cells* list box, the name of the cell for which you want to print the hierarchy of cellview instances.

View lets you specify the name of the view for which you want to print the hierarchy of cellview instances.

Type displays the type of the view.

Browse opens the *Library Browser*.

Library path file displays the path to your library definitions file. If the whole path is too long to be displayed, you can put the pointer in this field and use the arrow keys to display the rest of the path. You cannot edit this field.

Cells displays the names of all the cells in the library shown in the *Library* drop-down.

OK opens the *Hierarchy Tree* window for the selected cellview and closes the form.

Cancel closes the form without doing anything.

Apply opens the *Hierarchy Tree* window for the selected cellview. The *Print Hierarchy Tree* form remains open so that you can select other library, cell, and view for which you want to print the hierarchy of cellview instances.

Help opens help information in Cadence Help.

Hierarchy Tree Window File Menu

The *File* menu contains commands that let you open, save, search, and close the text window.

Open loads the contents of a text window previously saved with *Save As*.

Save As saves the data in this window to a text file.

Auto Update On and shaded command names are not used by text windows opened with the *Print Hierarchy Tree* command.

Close Window closes this text window.

© 2020 All Rights Reserved.

Related Topics

Using the Print Hierarchy Tree Command

© 2020 All Rights Reserved.

Propagate Nets Form

Terminal Name is the terminal name. Several pins can be defined on one terminal.

Net Name is the top-level net name. If a net name field is blank, the corresponding terminal is not currently mapped to a top-level net. Clicking *Defaults* displays the current top-level net name of the terminal.

Set to Terminal Name sets the net name to the terminal name.

Clear All Net Names clears the text fields.

Set All Net Names to Terminal Name sets all net names to the terminal names.

Show All Must-Join Terminal Names displays the must-join terminal names that exist in the cellview of the instance.

Related Topics

Propagating Nets

© 2020 All Rights Reserved.

Quick Align Form

Copy reference enables you to copy an object during alignment.
Environment variable: quickAlignMode

Note: The *Copy* option will only copy fully selected objects. It will not copy partially selected objects.

Snap Target enables you to select method to enable the pointer to the reference set and target object.

> **Edges and Points** snaps the pointer to edges, centerlines, vertices, nexus, and end points in the reference set and target object.
>
> **Edges** snaps the pointer to only the edges and centerlines in the reference set and target object.
>
> **Points** snaps the pointer to vertices, midpoint of edges, nexus, end points, and origin point of objects in reference set and target.

Environment variable: quickAlignSnapTarget

Offset

No Spacing does not apply any spacing between the reference edge, centerline, or point and the target edge, centerline, or point.
Environment variable: quickAlignSpacingType

Min Spacing applies the minimum spacing value of the reference object layer, as specified in the technology file, while aligning the reference set to the target edge, centerline, or point.
Environment variable: quickAlignSpacingType

User Spacing applies the spacing you specify to the distance between the reference edge, centerline, or point and the target edge, centerline, or point. You can specify an integer.
Environment variable: quickAlignSpacingType, quickAlignUserSpacing

Snap Target Visualization Settings

Use this section to customize the appearance of the edge and point highlights that appear on the potential reference and target edges and points.

> **Width** enables you to specify the width of the highlight. The default is `0`.
> Environment variable: targetWidth
>
> **Color** enables you to select the color of the highlight. The default is `yellow`.
> Environment variable: targetColor

© 2020 All Rights Reserved.

Line Style enables you to select the type of highlight. The default is `solid`.
Environment variable: targetLineStyle

© 2020 All Rights Reserved.

Quick Align Context-Sensitive Menu

Target Axis sets the *Vertical* or *Horizontal* axis for aligning the reference set of objects. By default, the target axis is created at the point of click in the design.

Spacing enables you to specify the spacing between the reference set and target object after alignment. The options available on the *Spacing* menu map to the *No Spacing*, *Min Spacing*, and *User Spacing* options on the Quick Align Form.

Switch to copy mode or **Switch to move/stretch mode** enables you to switch between *Copy* and *Move/Stretch* mode of alignment. If the *Move/Stretch* mode is selected in the Quick Align Form, then the *Switch to copy mode* option is available in this menu. However, if the *Copy* mode is selected in the Quick Align Form, then the *Switch to move/stretch mode* option is available in this menu.

Snap to targets in reference layer enables you to snap the pointer to only those targets that are on the same layer as the object with the reference point, edge, or centerline. If selected, this option on the context menu toggles to *Snap to targets on all layers*.

- The *Snap to targets in reference layer* option maps to the selected *On Same Layer* check box on the Quick Align Form.
- The *Snap to targets on all layers* option maps to the unselected *On Same Layer* check box.

Snap to Edges and Points snaps the pointer to edges, centerlines, vertices, nexus, and end points in the reference set and target object.

Snap to Edges snaps the pointer to only the edges and centerlines in the reference set and target object.

Snap to Points snaps the pointer to vertices, midpoint of edges, nexus, end points, and origin point of objects in reference set and target.

Options opens the Quick Align Form.

De(Select) Under Cursor submenu provides a list of all the objects currently selected and enables you to select the one you want to deselect. This command is available on the *Quick Align* context-sensitive menu only when you right-click over an object.

Related Topics

By Using the Quick Align Command

leHiQuickAlign

© 2020 All Rights Reserved.

Refresh Cellviews Form

Refresh these cellviews? shows the library, cell, and view name of cells that have been edited since you opened this cellview. You can choose to refresh, in memory, your copies of any of the listed master cells.

All or **None** lets you choose to update memory for all or none of the listed cellviews.

Related Topics

Refreshing Memory after Editing

© 2020 All Rights Reserved.

Remaster Instances Form

Search for

Library the name of the library that contains the cellviews you want to re-master.
Environment variable: remasterInstSearchLibText

CellName the name of the cellview that contains the cellviews you want to re-master.
Environment variable: remasterInstUpdateCellText

ViewName the name of the view to be re-mastered.
Environment variable: remasterInstSearchViewText

Update to

Library the name of the library that contains the replacement master cellviews.
Environment variable: remasterInstUpdateLibText

CellName the name of the cellview that contains the replacement master cellviews.
Environment variable: remasterInstUpdateCellText

ViewName the name of the view to replace the view in the *Search* for *ViewName* field.
Environment variable: remasterInstUpdateViewText

Check Terminals checks all corresponding cellviews for matching or a subset of matching instTerm information.
Environment variable: remasterInstCheckTerms

Related Topics

Re-mastering Instance Views

leHiRemasterInstances

© 2020 All Rights Reserved.

Remove Shape From Net Form

Net Name shows the net of the selected shape. If more than one shape is selected, Net Name shows the net of the first shape. If the net name shown is not the net you want the shapes deleted from, type the correct net name.

Related Topics

Removing Shapes from a Net

leHiDeleteShapeFromNet

© 2020 All Rights Reserved.

Repeat Copy Form

Release Direction resets the paste direction and allows you to paste the copied object in any direction and at any distance with respect to the previous pasted object. When off, the direction of pasting the object is fixed, which is the direction in which the copied object was last pasted. The check box is available only after you have placed the first copy.

Snap Mode controls how the mouse pointer snaps to the grid as you place the copies.

> **anyAngle** enables you to move the pointer at any angle.
>
> **diagonal** enables you to move the pointer parallel to the X or Y axis or at a 45-degree angle to the axes.
>
> **orthogonal** enables you to move the pointer parallel to the X and Y axis.
>
> **horizontal** enables you to move the pointer parallel to the X axis.
>
> **vertical** enables you to move the pointer parallel to the Y axis.

Delta

> **X** and **Y** enables you to specify the exact distance from the object at which you want to place the copy.
>
> **Apply XY** places the copy at the specified distance from the object.

Copy Connectivity copies the connectivity information on path, pathseg, fluid guard ring, and via to the new object.

Environment variable: copyConnectivity

Snap To Grid snaps the pin edges to the manufacturing or the routing grid, depending on the selected block type.

- If the block type is custom, the pin edges are snapped to the manufacturing grid.
- If the block type is digital, the pin edges are snapped to the routing grid.

Create Synchronous Copy creates a synchronized copy of the selected layout objects. For more information, see Generating a Synchronous Copy.

Rotate turns the object 90 degrees counterclockwise each time it is pressed.

Sideways mirrors the object along the X axis.

Upside Down mirrors the object along the Y axis.

© 2020 All Rights Reserved.

Related Topics

Copying Objects in Repeat Mode

leHiRepeatCopy

© 2020 All Rights Reserved.

Reshape Form

Mode sets the geometry to use for reshaping a selected object.

rectangle lets you add or remove a rectangle shape.

line lets you add a polygon to a shape or reshape a path. Only *line* can be used to reshape a path.

Snap Mode controls the shape of line segments; applies only when *Reshape Type* is set to *line*.
Environment variable: snapMode

Snap

Pins and boundary to grid snaps the stretched object to a grid depending on the block type. If it is an *analog* block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a *digital* block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not snapped, even if they were touching the boundary before the reshape.

During the stretch operation, in case the soft block is stretched outside the top-level prBoundary, then all the instance pins on the soft block, with the placement status as PLACED, are snapped to the manufacturing grid. This is irrespective of the block type because the routing grid is not initialized outside the prBoundary but the manufacturing grid exists outside the prBoundary.

Pins to boundary automatically snaps pins to the place and route boundary.

Snap Blocks Pins to automatically snaps soft block pins either to the *Modified Edge* or to the *Closest Edge* after the reshape is complete.

Related Topics

Reshaping Objects

Editing Soft Block Objects

Reshaping an Object in Level-1 Editing Mode

© 2020 All Rights Reserved.

Restore View Form

The text box list the last three saved views.

Related Topics

Restoring a Zoom or Pan Image

© 2020 All Rights Reserved.

Return To Level Form

The text box lists the levels above the current one in the hierarchy and the cell associated with each level.

Related Topics

Returning to a Different Hierarchy Level

© 2020 All Rights Reserved.

Rotate Form

Angle of Rotation sets the angle to rotate the object you select. You can select *90*, *180*, or *270* as the angle of rotation. If you select *90*, it turns the object, instance, or label 90 degrees counterclockwise. If you select 180, it mirrors the object about the Y axis, or turns an instance or label 180 degrees counterclockwise. If you select 270, it mirrors the object about the X axis, or turns an instance or label 270 degrees counterclockwise.

Snap Mode selects snap mode to use when you rotate objects. Valid snap modes are anyAngle, diagonal, orthogonal, horizontal, or vertical.

Precision controls the precision of the angle of the snap.

> **1 Degree** lets you use whole numbers; for example, 10.
>
> **.1 Degree** lets you use one decimal place; for example, 12.5.
>
> **Any** lets you use three decimal places; for example, 15.125, assuming 1,000 data base units/user units.

Related Topics

Rotating Objects

leHiRotate

© 2020 All Rights Reserved.

Save a Copy Form

The list box shows the name of the cell and the current library, cell and view in which it exists.

The following three fields at the bottom of the form let you specify a different name and location to save the cellview:

The first cyclic field lists libraries. You can choose the one in which you want to save the cellview.

Environment variable: saveACopyLibName

The second text field lets you specify a different name for the cell. If you retain the name and also do not change the library and view locations, the existing cellview will be overridden.

The third and last cyclic field lets you select the view type for the cell you are saving.

Open Saved Copy if selected, lets you open a saved copy of the design. The **Open for** and **Open in** fields are enabled when this option is selected.

Environment variable: openSavedCellViewInWindow

Open for lets you open a saved copy of the design for reading or editing.

Open in lets you choose where you want to open the saved copy of the design, in a new tab, current tab, or new window.

Related Topics

Saving a Design

© 2020 All Rights Reserved.

Save All Mark Nets Form

Library lets you specify the library in which you want to save the mark nets.

Cell lets you specify the cell in which you want to save the mark nets in the specified library.

View lets you specify the view name in which you want to save the mark nets in the specified library and cell.

Browse opens the *Library Browser* to let you select the library, cell, and view in which to save the generated mark nets.

Related Topics

Saving Marked Nets

leHiSaveAllHighLightMarkNet

© 2020 All Rights Reserved.

Save/Restore Selection Set

Name allows you to enter a name to a selected set of objects in order for you to retrieve it later.

Store allows you to save this information to a file for later retrieval. If the Store button does not work the selection set is empty, or the name you are entering in the name field already exists. A warning is issued in the CIW.

Stored Selection Sets are entered into the field once you click the Store button.

Select retrieves the stored information and displays it on the originally selected set. You must click the stored selection set previous to clicking the Select button in order to bring up the stored selection set you want to display. Each selection you make adds the additional selection to the display unless you click the deselect button from the form before making another selection.

Deselect removes the selection from the stored set from the objects in the stored selection set you have selected to display in the cellview.

Delete removes the set you have selected in the Stored Selection Sets form window. If the deleted set is the current selection set the selection is removed from the form, but remains displayed until you change the selection.

Related Topics

Selecting Objects

© 2020 All Rights Reserved.

Save Configuration Form

Select configuration name: specifies the name of the configuration you are saving.

Select path: specifies the path for the configuration display file. The configuration is saved to the `odc.il` in the location of *`selectedPath`*`/dfII/ODC/odc.il`. The path that you choose to save the configuration to will determine the sequence in which a configuration is loaded and is derived from the `setup.loc` file.

Related Topics

Saving the Display Information Configuration

© 2020 All Rights Reserved.

Save Via Variant Form

Save to specifies where to save the via variant - in the technology library or in the design itself. If you select *Technology Library*, select the library from the drop-down list in which to save the via variant.

Save as specifies the name of the new via variant. The drop-down list shows the earlier saved, if any, via variants. A warning message is generated if you overwrite an existing via variant.

Summary list box is read-only and displays all the parameters, including the updated ones, for the new via variant. `viaDef` at the top indicates the parent viaDef used for the creation of the via variant.

Related Topics

Create Via Form

Creating Vias

© 2020 All Rights Reserved.

Save View Form

Name sets the name you want to assign this view. This name will appear in the Restore View form.

Related Topics

Saving or Restoring a View

© 2020 All Rights Reserved.

Save Multipart Path Templates Form

Template List list existing templates.

ASCII Fie saves the template definition to an ASCII file.

Note: The extension of the template file can be `.il` or you can use any other extension.

Technology Library saves the template definition to the selected technology library.

Related Topics

About Saving a Multipart Path as a Template

© 2020 All Rights Reserved.

Select Area Shape Type Form

Shape Type enables you to use a *rectangle* (selected by default) or a *polygon* to define an area for selecting objects.

Related Topics

Selecting Objects

© 2020 All Rights Reserved.

Select Array Member Form

Row indicates the row the array member is in.

Column indicates the column the array member is in.

Related Topics

Descending into a Cellview

© 2020 All Rights Reserved.

Selection Options Form

This form comprises the following sections:

- Selection Controls
- Point Selection Controls
- Area Selection Controls
- Auto Store Selection Controls

Selection Controls

Mode controls whether the selection is full or partial.

Full selects objects fully.

Partial selects objects fully if you completely enclose them within the area selection box, and partially selects objects if you partially enclose them within the area selection box.

Spine selects all the contiguous pathsegs that have the same width as the first-clicked pathseg. It excludes the pathsegs that branch off from the main spine at a T junction. This option is selected by default. If you do not select this option, then only the pathseg you click at gets selected.
Environment variable: spineSelection

Allow different width enables spine selection for pathsegs with different widths. This option is available only when *Spine* is selected.

Via Stack allows the selection of a via stack with a single click.
Environment variable: viaStackSelection

Vias with matching sizes between common layers lets you select vias of the same via stack when there are two or more super-imposed via stacks on adjacent layers with the same origin. Vias of the same via stack have the same size on common layers. If this option is off, multiple super-imposed via stacks are selected as a single via stack and are available for editing as a single via stack. This option is available only when *Via Stack* is selected.

For each pair of adjacent vias in a via stack, the bounding box of the top layer in the lower via is equal to the bounding box of the bottom layer in the higher via. This refers to the matching sizes of the vias between the common layers and is used for identifying vias of the same via stack. For example, the two vias M1M2 and M2M3 in the via stack M1M3 are on adjacent layers and share the same origin. In addition, the bounding box of Metal2 shape in M1M2 is equal to the bounding box of the

© 2020 All Rights Reserved.

Metal2 shape in M2M3. This testifies that the two vias belong to the same via stack. Environment variable: viaStackSelectionMatchingSizes

Mosaic Partial Selection allows partial selection of mosaics.

Via Partial Selection allows partial selection of vias. You can control the status of this option using the following SKILL command:

```
hiGetCurrentWindow()~>selectPartialVia
```

Net Based Selection allows selection of nets in the layout editor.
Environment variable: netBasedSelectionOn

Repeat Commands allows the commands to automatically repeat, if you first select the command and then the object. This option does not affect commands if you first select an object, then the command.

Point Selection Controls

Environment variable: pointSelectCycleMode

Overlap Mode

Toggle mode allows you to change selection between two objects.

- Click using the left mouse button on one object and the software toggles back and forth between the smaller object and the larger object.
- You can add an unlimited number of objects to the selected set by holding down the shift key and selecting additional objects.
- You can delete the selections by using the control key and clicking the left mouse button on the original selected object. The objects are deselected in the opposite order in which they were selected.
- Once you have selected the objects you want in the set you can also click the selected edge of one of the objects. This deletes all of the selections except the one you have clicked on. This does not delete the objects themselves.

Cycle mode allows you to select overlapped objects one by one by using the left mouse button to select any object and continuing to select them by clicking on the original object until you reach the largest overlapped object.

- The cycle mode is for use when three or more objects overlap at the same point.
- The cycle mode always cycles from the smallest to the largest object. If you continue to click the original object after reaching the largest object the cycle returns to the smallest object you first clicked on. The cycle repeats by re-selecting the smallest object you clicked on and selecting outwards again to the

© 2020 All Rights Reserved.

largest object. If you start by selecting the largest object in the overlapping stack the selection will not cycle back to the smaller objects. You must select a smaller object so that each selected point selects larger and larger objects.

- Multiple objects are not retained in the selected set. The final selected object is the selection that is retained. You can unselect by clicking outside of all of the overlapping objects.
- You can add an unlimited number of objects to the selected set by holding down the shift key and selecting additional objects.
- You can delete the one of the selections by using the control key and clicking the left mouse button on the selected object. The objects are deselected in the opposite order in which they were selected.
- Once you have selected the objects you want in the set you can also click the selected edge of one of the objects. This deletes all of the selections except the one you have clicked on. This does not delete the objects themselves.
- The objects are selected according to the selection filter set for the Layer Selection Window and the Object Selection Window.
- The Cycle mode is window based.

User Aperture Mode defines a box around the initial point. You can select multiple points within this aperture range and the software will treat it as one point. The default is five pixels. You can specify the size of the box in the **Size** field.
Environment variable: pointSelectAperture

off specifies that point selection does not use the specified aperture.

on specifies that point selection uses the specified aperture.

visible specifies that point selection uses the specified aperture, which displays on the canvas.

Select objects larger than the window enables you to control whether you can select an object that has bounding box or boundary greater than the size of the design canvas. When this option is not selected, you cannot select objects larger in size than the design canvas. This check box is selected by default.
Environment variable: selectObjectLargerThanWindow

Select objects of edit cellview first enables you to point-select objects belonging to the edit cellview in preference over objects from another cellview, such as, shapes of instances, even though objects of the edit cellview might be on a higher layer.
Environment variable: selectObjectInEditCvFirst

Area Selection Controls

© 2020 All Rights Reserved.

This section allows you to toggle between full and partial area selection modes.

Full Mode controls the area selection in *Full* selection mode.
Environment variable: fullSelectionMode

Enclosed Figures selects the objects you completely enclose with the area selection box.

Enclosed and Crossed Figures selects the objects you completely or partially enclose with the area selection box.

Crossed Figures selects the objects you partially enclose with the area selection box and does not select any fully-enclosed objects.

Partial Mode controls the area selection in *Partial* selection mode.
Environment variable: partialSelectionMode

Vertex selects the objects you completely enclose with the area selection box, the vertices of any objects you partially enclose with the area selection box, and the edges of any objects you enclose with the area selection box.

Enclosed Edges selects the objects you completely enclose with the area selection box and the edges of any objects you partially enclose with the area selection box.

Crossed Edges selects the edges of any objects you enclose with the area selection box and does not select any fully-enclosed objects.

Enclosed and Crossed Edges selects the objects you completely enclose with the area selection box, the vertices of any objects you partially enclose with the area selection box, and the edges of any objects you partially enclose with the area selection box.

Auto Store Selection Controls

This section stores multiple sets of selections you have made. You can reload into the cellview these selections without have to reselect the objects. The radio button controls whether or not you can store the selections.
Environment variable: autoStoreSelectionSetOn

- The *Auto Store* option is disabled by default.
- The selection set consists of full or partially selected figures.
- The selection set is a cellview property. The *Edit – Select – Save/Restore* is not cumulative. Once the cellview is closed the recorded information is not retained.

© 2020 All Rights Reserved.

- You can iterate through the sets by selecting the sets in the in the *Save/Restore Selection Set* form or by using the *Edit – Select – Next Saved Set* or *Edit – Select – Previous Saved Set* commands.
- Deleted or modified objects are removed from the saved selection set. If all objects are removed from the selection set, the set is removed from the Save/Restore Selection Set form.
- Restore is independent of the object selectability of the Layer Selection Window and the Object Selection Window.
- The command is not compatible with the Undo or Redo commands.
- If the subview is already open, the command does not work with Edit In Place nor the Descend command.

Max number of Stored sets limits the number of sets which can be retained by the Save/Restore Selection Set form.
Environment variable: maxAutoStoreSelectionSet

- The maximum number and the default of stored sets is ten, however you can set the number lower if ten is not required.
- The oldest saved selection set is removed, without generating a warning or error message to you.

Max number of selected elements within a set limits the number of objects you can enter into one selection set.
Environment variable: maxAutoStoreSelectionObj

- The maximum number and the default of stored sets is one hundred, however you can set the number lower if one hundred is not required.
- The oldest saved selection set is not recorded and generates a warning message in the CIW.

Related Topics

Selecting Objects

© 2020 All Rights Reserved.

Selection Protection Highlight Options Form

Options

Halo Mode determines the type of halo around a protected object. The *normal* mode does not generate any halo. The default mode is *fadeout*. The figures below show the effect of different modes on objects marked protected.

Environment variable: protectedObjHighlightHaloMode

normal *fadeout* *glow* *plain* *random*

Halo Color determines the color of the halo. The default halo color is yellow. When the *Halo Mode* is set to *glow*, the *Halo Color* appears on the outer edges of the halo. The figures below show examples of the halo in three colors.

yellow orange green

Level determines the placement of the halo relative to the object boundary box. The default level is *under*.

Environment variable: protectedObjHighlightHaloLevel

over *under*

Width specifies the thickness of the halo on a protected object. The default width is *normal*. The figures below show halos with different thickness.

© 2020 All Rights Reserved.

Environment variable: protectedObjHighlightHaloWidth

thick *normal* *thin*

Transparency controls the visibility of the halo. The visibility of the halo is the maximum at the lowest transparency setting. The default transparency value is `40`. The figures below show halos at different transparency settings.

Environment variable: protectedObjHighlightHaloTrans

10 *40* *70*

Related Topics

Making Objects Selectable and Unselectable

© 2020 All Rights Reserved.

Set Area View Level Form

Display Levels specifies the first (*From*) and last (*To*) levels in the design hierarchy that can be seen in detail. The hierarchy levels are numbered 0 to 32. Shapes in the current cellview are at level 0, shapes in the masters of instances inside of it are level 1, and so forth.

Related Topics

Setting an Area Display

leHiSetAreaViewLevel

© 2020 All Rights Reserved.

Set Default Application Form

Current application specifies the current application in which the cellview is opened.

Current cellview default specifies the default application setting for the current cellview.

Current default for all maskLayout cellviews specifies the default application setting for all cellview's with the view type of the current cellview.

Set default application for current cellview to sets the default application for the current open cellview.

> **As Is** specifies to retain the current application setting.
>
> **None** removes any default application settings.
>
> ***anyRegisteredApplication*** specifies to set the default application to the selected registered application.

Note: To apply this setting, the environment variable saveViewSubType must be set to nil, otherwise the Layout cellview automatically opens in the same application in which it was last saved.

Set default application for all maskLayout cellviews to sets the default application for all cellviews with the view type of the current cellview.
Environment variable: maskLayoutDefaultApp

> **As Is** specifies to retain the current application setting.
>
> **None** removes any default application settings.
>
> ***anyRegisteredApplication*** specifies to set the default application to the selected registered application.

Note: To apply this setting, the environment variable saveViewSubType must be set to nil, otherwise the Layout cellview automatically opens in the same application in which it was last saved.

Related Topics

Specifying Default Application in .cdsenv

Setting the Default Application for a Layout Cellview

© 2020 All Rights Reserved.

Set Pin Label Text Style Form

Height sets the height of the name in user units (usually microns).

Font sets the text style of the name.

Text Options

Drafting prevents the label from being rotated more than 90 degrees.

Overbar determines how text strings containing underscore characters are displayed.

Create As Label determines whether the pin is created as a label or text display. If this option is selected, a label is created, else a text display is created.

Environment variable: pinTextCreateAsLabel

Note: The label is displayed for the pin only if the *Pin Names* option in the Display Options form is enabled.

Layer Name sets the layer for the pin name.

Same as Pin puts the pin name on the same layer as the pin

Environment variable: pinTextSameLayer

Layer Purpose puts the pin name on the same layer as the pin being created.

Same as Pin puts the pin name on the same layer as the pin

Environment variable: pinTextSamePurpose

Justification sets the location of the name origin.

Orientation sets the orientation of the pin name.

Environment variable: pinTextOrientation

Related Topics

Setting Pin Label Text Style Options

© 2020 All Rights Reserved.

Show Selected Set Form

The *Show Selected Set* section shows information about the each type of object that is selected in the current cellview.

File

The *File* menu lets you manipulate this text window and its contents.

Open loads the contents of a text window you previously saved with *Save As*.

Save As saves the data in this window to a text file.

Auto Update On and shaded command names are not used by text windows opened with the *Show Selected Set* command.

Search searches through the text in this window for a text string.

Close Window closes this text window.

Related Topics

Viewing Selected Object Information

leHiShowSelSet

© 2020 All Rights Reserved.

Size Form

Increase Size By Value controls the amount by which you enlarge or reduce the object. Positive numbers enlarge, negative numbers reduce.

Increase Size By Direction controls the amount by which you enlarge or reduce the object in the *Left*, *Right*, *Top*, or *Bottom* direction. Positive numbers enlarge, negative numbers reduce.

Related Topics

Sizing Objects

leHiSize

© 2020 All Rights Reserved.

Split Form

Lock angles prevents you from changing the angle of a corner or edge as you stretch it.

Snap Mode controls the direction in which you can stretch an edge.
Environment variable: snapMode

Keep Wires Connected to splits paths, pathsegs, and via vertices coincident with a selected vertex. These are split as a set.

> **All wires and vias** maintains pathseg-to-pathseg and pathseg-to-via connections during split.
>
> **Spine wires only** maintains connections to spine pathsegs but not to connecting vias when splitting a spine. This option is selected by default.
>
> **No wires and vias** does not maintain pathseg-to-pathseg and pathseg-to-via connections during split.

Keep wires connected to shapes maintains connection with shapes, such as pin figures, rectangles, polygons, and MPP, coincident with the selected vertex during the split operation. This check box is selected by default.

Lock Unselected Vias locks the vias that are not selected while splitting a wire. By default, this option is disabled. It is enabled only when the *All wires and vias* option is selected from the *Keep Wires Connected To* field.

Related Topics

Splitting Objects

leHiSplit

© 2020 All Rights Reserved.

Stretch Form

Lock Angles prevents you from changing the angle of a corner or edge as you stretch it.

Snap Mode controls the direction in which you can stretch an edge.
Environment variable: snapMode

Delta X,**Y** lets you specify the number of user units the object will stretch on the X and/or Y axes relative to its point of origin.

Apply stretches the origin point of the selected object as specified in the Delta X and Delta Y fields.

Display draglines controls whether draglines are displayed during the *Stretch* command. By default, the draglines indicate connections from the pins of the object you are moving to pins of the nearest objects.

Environment variable: flightLineEnable

Tip

When constraint-aware editing is on, the *Stretch* command honors the following physical constraints: *Alignment*, *Fixed*, *Locked*, *Matched Parameters*, *Orientation*, *Matched Orientation*, and *Symmetry*.

- For more information on Constraint-Aware Editing mode, see Constraint-Aware Editing.
- For detailed definitions of the supported constraints, see Default Constraint Types in the *Virtuoso Unified Custom Constraints User Guide*.

Connections

Keep Wires Connected to stretches paths, pathsegs, and via vertices coincident with a selected vertex. These are stretched as a set.

All wires and vias maintains pathseg-to-pathseg and pathseg-to-via connections during stretch.

Spine wires only maintains connections to spine pathsegs but not to connecting vias when stretching a spine. This option is selected by default. For information about selecting a spine, see Selecting a Path Vertex.

No wires and vias does not maintain pathseg-to-pathseg and pathseg-to-via connections during stretch.

© 2020 All Rights Reserved.

Keep wires connected to shapes maintains connection with shapes, such as pin figures, rectangles, polygons, and MPP, coincident with the selected vertex during the stretch operation. This check box is selected by default.
Environment variable: keepWiresConnectedToNonWires

Lock Unselected Vias locks the vias that are not selected while splitting a wire. By default, this option is disabled. It is enabled only when the *All wires and vias* option is selected from the *Keep Wires Connected To* field.

Keep guard ring ends connected enables you to keep the two touching ends of a guard ring together while stretching the guard ring. If the check box is not selected, only the selected end of the guard ring is stretched and results in opening up the guard ring. This check box is not selected by default.

Note: This check box is common in the and the Quick Align Form. The setting in one form is preserved in the other form.

Chain Mode lets you move device chains. (You can only completely select instances, not parts of instances, so that the stretch command will only allow you to move them using the stretch command. To stretch parts of instances you must use the stretch handles.)

All moves all the instances in a chain.

Selected moves only selected instances.

Selected Plus Left moves the selected instance and all instances to the left of it in the chain.

Selected Plus Right moves the selected instance and all instances to the right it in the chain.

Special Objects

Via Mode defines the mode of stretching of the via.

Stretch Row/Col maintains the number of rows when stretching horizontally and the number of columns when stretching vertically.

Stretch Metal fills the metal in the via as per the dimensions of the new box created after the stretch operation.

Environment variable: stretchViaMode

Allow opposite metal extend ensures that the cut box stays at the same location even if the enclosure is increased after the stretch operation.The metal of the new via goes beyond the existing metal. This option is available only when the *Stretch Row/Col* via mode is selected.

© 2020 All Rights Reserved.

Snap

Pins and boundary to grid snaps the stretched object to a grid depending on the block type. If it is an *analog* block, the place and route boundary and pin edges are snapped to the manufacturing grid. If it is a *digital* block, the place and route boundary is snapped to the placement grid and the pin centers are snapped to the routing grid. Pins with placement status `unplaced`, `fixed`, `locked`, or `unknown` are not snapped, even if they were touching the boundary before the stretch.

Pins to boundary automatically snaps pins to the place and route boundary.

Soft Block Pins to automatically snaps soft block pins either to the *Modified Edge* or to the *Closest Edge* after the reshape is complete.

> **Note:** During the stretch operation, in case the soft block is stretched outside the top-level prboundary, then all the instance pins on the soft block, with the placement status as PLACED, are snapped to the manufacturing grid. This is irrespective of the block type because the routing grid is not initialized outside the prboundary but the manufacturing grid exists outside the prboundary.

Boundary

Constant area stretch maintains the area of a soft block constant while its place and route boundary is stretched. Additional options let you specify the adjustable edges, that is, the edges that can be moved automatically in order to maintain a constant area.

Left edge, **Right edge**, **Top edge**, **Bottom edge** automatically snaps soft block pins to the *left edge*, right edge, top edge, or bottom edge after the reshape is complete.

Stretch with P&R boundary stretches along with the P&R boundary, the shapes intersecting an edge of a P&R boundary or abutted to an edge of a P&R boundary. The supported shapes are orthogonal pathSegs, rectilinear pins, rectangles and row regions. Environment variable: stretchWithPRBoundary

Related Topics

Stretching Objects

Stretching an Object in Level-1 Editing Mode

Stretching Multipart Paths

Stretching Wires and Vias

Removing Loops from Wires

© 2020 All Rights Reserved.

leHiStretch

© 2020 All Rights Reserved.

Summary Window

The *File* menu on the *Summary* form lets you manipulate this text window and its contents.

- *Open* loads the contents of a text window previously saved with *Save As*.
- *Save As* saves the data in this window to a text file.
- *Auto Update On* and shaded command names are not used by text windows opened with the *Summary* command.
- *Find* searches through the text in this window for a text string.
- *Close Window* closes this text window.

Show Environment

The *Show Environment* section shows information about the cellview and its edit mode, display levels, and current entry layer (entry layer).

Layer Object Statistics

The *Layer Object Statistics* section lists the number of shapes created in the cellview and the layer on which each is present. The shape types listed are Arc, Bend, Donut, Dot, Eliipse, Label, Line, Path, PathSeg, Polygon, Rect, Trl, Taper, and TextDisplay.

Instance Statistics

The *Instance Statistics* section shows the names of the cell instances and arrays of instances (mosaics) that have been placed in this cellview, as well as the total number of unplaced instances.

Design Statistics

The *Design Statistics* section includes the Layer Summary that displays the total number of metal and routing layers. The Component Summary portion of this section reports the placement status of the components.

Routing Grid Statistics

The *Routing Grid Statistics* section displays the x and y steps, offsets and the manufacturing grid information of the valid routing layers.

© 2020 All Rights Reserved.

Note: Valid routing layers belong to a valid constraint group. The default constraint group for the design is considered to be the valid constraint group. The following constraint groups are considered in this order; if one does not exist, the next one in the order is considered: default constraint group, active constraint group, foundry constraint group. If the foundry constraint group also does not exist, then `Metal` and `Poly` are considered as valid routing layers.

Wire Statistics

The *Wire Statistics* section shows the total number of standard and custom vias, total number of path and path segments and total length of paths and path segments that have been placed in this cellview.

Marker Object Statistics

The *Marker Object Statistics* section shows the number of error markers that have flagged design violations in this cellview.

Row Object Statistics

The *Row Object Statistics* section shows the number of horizontal and vertical rows that have been placed in this cellview.

Track Pattern Statistics

The *Track Pattern Statistics* section shows the number of horizontal and vertical tracks that have been placed in this cellview.

Blockage Object Statistics

The *Blockage Object Statistics* section shows the number of blockage objects, placement area blockage objects, placement area halo objects and layer blockage objects that have been placed in this cellview. It displays the total number of blockage objects, the blockage count of each type of blockage in the cellview, and is sorted by layer

PR Boundary Object Statistics

The *PR Boundary Object Statistics* section shows whether or not a PR boundary object has been placed in this cellview, shows the origin point of the PR boundary and its area measurement in user units, and the shape of the PR boundary.

© 2020 All Rights Reserved.

Related Topics

Listing the Data In Cellviews

© 2020 All Rights Reserved.

Tap Form

Layer sets the tapped layer as the entry layer, and updates the *Layer* assistant. This selection includes paths, path segments, wires, rectangles, polygons, circles, ellipses, donuts, microwave objects, labels, and textDisplays.

Net captures net names from rectangles, polygons, and paths, wires, wire segments, labels, or text displays. You can capture the net names to automatically insert the net names in the Propagate Nets Form when applying connectivity to a via.

Attributes captures attributes from paths, instances, vias, rows and microwave objects.

Tapping blockages captures the blockage type and the maximum density of slot and fill blockages, and the effective width, if any. It captures directional offsets if the blockage has a halo. Tapping blockages does not change the active layer.

Tapping path, wire and wire segments captures the beginning and ending extension in user units, the path end type, and the path width.

Tapping instances captures the instance master's cell type, the list of halos on the instance, as well as any clusters associated with the instance.

Tapping rows captures the number of sites, whether the row is vertical or horizontal and the orientation of the row.

Trls, bends and tapers attributes capture the bend factor (maximum width before they are chamfered), taper style (linear or exponential), number of points to taper in user units, the bend style (bend chamfer or radial), the width of a chamfered bend in a transmission line as a multiple of its standard width in user units, the number of segments in a radial bend in a transmission line as a function of the number of segments in a 180-degree radial bend, the radius of a radial bend in a transmission line as a multiple of the transmission-line width in user units, the width of the first end of a transmission line bend in user units, the width of the second end of a transmission line bend, and the width of the second end of a transmission line bend in user units.

Command (shapes, wires) starts the create command for the tapped shape or wire. The following table lists the command that will be started for each object type when the *Command (Only for Shapes/Wires)* option is selected.

Object Type	Command
Rectangle	leHiCreateRect
Polygon	leHiCreatePolygon

© 2020 All Rights Reserved.

Object Type	Command
Path	leHiCreatePath
Ellipse	leHiCreateEllipse
Donut	leHiCreateDonut
PathSeg	leHiCreateGeometricWire

Note: The *Command (Only for Shapes/Wires)* option only works for shapes and wires. It does not work for other object types, such as instances, vias, and blockages.

Pop up selection window brings up the *Choose Object to* form. When you click an object with multiple layers, the form appears with a list of layers on which you can create the object.

Related Topics

Using the Tap Command

leHiLayerTap

© 2020 All Rights Reserved.

Track Pattern Editor Form

Filter by Routing Layer filters the routing layers for track patterns to be displayed in the form for the current design.

> **all** updates the form to show all routing layers to which tracks patterns are assigned.
>
> **layerName** updates the form to only show track patterns that are assigned to the specified routing layer.
>
> **Note:** Toggle selection of track patterns using the left and right mouse buttons.

Delete deletes the selected track pattern.

Layer specifies the routing layer to associate with the new track pattern.

Direction

> **Horizontal** defines horizontal tracks
>
> **Vertical** defines vertical tracks

Spacing specifies the spacing between tracks. Also know as the routing pitch.

Origin Offset offsets the tracks up and to the right of the origin point by the specified distance.

Number of Tracks specifies the number of tracks (horizontal or vertical) associated with the selected track pattern.

Color Pattern specifies the color pattern for track patterns. The color pattern is used when there is more than one mask for a layer.

Name assigns the track pattern as a constraint to a constraint group that can then be assigned to a net. This causes the net to be routed on the specified tracks.

Pattern Group Name is a constraint group that contains the `allowWidthRanges` constraint to determine the width associated with the track.

Width specifies the width associated with the track.

For more information about creating track patterns for Virtuoso® Multi-Patterning Technology using the fields *Color Pattern*, *Name*, *Pattern Group Name*, and *Width*, see Track-based Coloring in the *Virtuoso Multi-Patterning Technology User Guide*.

© 2020 All Rights Reserved.

Related Topics

Creating Track Patterns

leHiCellviewTrackPatterns

© 2020 All Rights Reserved.

Transmission Line Bend Form

Snap Mode controls the shape of the transmission line segments.

Bend Style sets the type of corner to use in the bend. You can choose from: *bend*, *chamfer*, or *radial*.

Side 1 Width and **Side 2 Width** set the width of the two segments in user units (typically microns).

Bend Factor sets the maximum width allowed in a standard bend before the bend is automatically chamfered. (Available only when *Bend Style* is *bend*.)

Chamfer Factor sets the fraction of the corner that is removed in a chamfer bend. (Available only when *Bend Style* is *chamfer*.)

Radius Factor sets the ratio of the centerline bend width to the radial bend width. (Available only when *Bend Style* is *radius*.)

Resolution sets the number of segments used in a 180-degree radial bend.
Environment variable: trlNPtsToPI

Related Topics

Creating Bends

leHiCreateBend

© 2020 All Rights Reserved.

Transmission Line Form

Mode controls how transmission line segments snap to the grid.

Bend Style sets the type of corner to use for the bends in the transmission line. You can choose from: *bend*, *chamfer*, or *radial*.

Width specifies the transmission line width in user units (typically microns).

Bend Factor sets the maximum width allowed in a standard bend before the bend is automatically chamfered. (Available only when *Bend Style* is *bend*.)

Chamfer Factor sets the fraction of the corner that is removed in a chamfer bend. (Available only when *Bend Style* is *chamfer*.)

Radius Factor sets the ratio of the centerline bend width to the radial bend width. (Available only when *Bend Style* is *radius*.)

Resolution sets the number of segments used in a 180-degree radial bend.
Environment variable: trlNPtsToPI

Related Topics

Creating Transmission Lines

leHiCreateTrl

© 2020 All Rights Reserved.

Transmission Line Taper Form

Snap Mode controls how the points of the taper snap to the grid.

Taper Style sets the taper style you create:

linear

exponential

Side 1 Width and **Side 2 Width** specify the width of the taper ends in user units (typically microns).

Resolution sets the number of segments used in an exponentially tapered line.
Environment variable: trlNPtsToPI

Related Topics

Creating Tapers

leHiCreateTaper

© 2020 All Rights Reserved.

Transparency Options Form

All Linked check box changes the *Outline* and *Fill* values of *Tracks*, *FinFET/WSP Shapes*, *Regions*, and *Wire Editor Tracks* together by tuning only a single slider.

Transparency Enabled check box enables the transparency options in the form.

Tracks

Linked check box lets you change the *Outline* and *Fill* values of *Tracks* by tuning only one slider at a time.

Outline lets you set the transparency for the outline of tracks by tuning the slider.

Fill lets you set the transparency for the file of tracks by tuning the slider.

FinFET/WSP Shapes

Linked check box lets you change the *Outline* and *Fill* values of *FinFET/WSP Shapes* by tuning only one slider at a time.

Outline lets you set the transparency for the outline of FinFET/WSP shapes by tuning the slider.

Fill lets you set the transparency for the file of FinFET/WSP shapes by tuning the slider.

Regions

Linked check box lets you change the *Outline* and *Fill* values of *Regions* by tuning only one slider at a time.

Outline lets you set the transparency for the outline of regions by tuning the slider.

Fill lets you set the transparency for the file of regions by tuning the slider.

Wire Editor Tracks

Linked check box lets you change the *Outline* and *Fill* values of *Wire Editor Tracks* by tuning only one slider at a time.

Outline lets you set the transparency for the outline of wire editor tracks by tuning the slider.

Fill lets you set the transparency for the file of wire editor tracks by tuning the slider.

© 2020 All Rights Reserved.

Related Topics

Display Enhancements for WSP

© 2020 All Rights Reserved.

Tree Form

Display sets how much of the hierarchy is listed.

Top to stop level lists the entire hierarchy of instances in the cellview, from the top level to the to the *Display Levels: Stop* field on the Display Options form.

Current to stop level lists the hierarchy of instances from the current level of hierarchy to the *Display Levels: Stop* field on the Display Options form.

Top to current lists the hierarchy of instances from the top level to the current level of hierarchy.

Current to user level lists the hierarchy of instances from the current level of hierarchy to the level specified by you in the user level spin box.

Tree Window File Menu

The *File* menu contains commands that let you open, save, search, and close the text window.

Open loads the contents of a text window previously saved with *Save As*.

Save As saves the data in this window to a text file.

Auto Update On / *Auto Update Off* and shaded command names are not used by text windows opened with the *Tree* command.

Find searches through the text in this window for a text string.

Close Window closes this text window.

Related Topics

Using the Tree Form

leHiTree

© 2020 All Rights Reserved.

Uniquify Form

Library lets you specify the name of the library that contains the cellviews you want to uniquify.

Cell lets you specify the name of the cell that contains the view to be uniquified.

View lets you specify the name of the view to be uniquified.

Related Topics

Uniquifying the Cellview

leUniquifyCellView

© 2020 All Rights Reserved.

Unmark Net Form

Unmark All removes the highlighting from all marked nets. If you want to delete the highlighting of a single marked net, click that net.

Related Topics

Unmarking Nets

leUnmarkNet

© 2020 All Rights Reserved.

Yank Form

Yank Shape sets the type of shape you draw around, or through, the objects you want to copy.

Yank Levels sets the number of levels of design hierarchy through which *Yank* can copy shapes. The current cellview is level 0, instances inside it are level 1, and so forth.

Snap Mode controls the shape of line segments when *Yank Shape* is set to *polygon*.

Related Topics

Yanking and Pasting Objects

leHiYank

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

B

Environment Variables

Environment Variable Functions

This appendix contains the Virtuoso® Layout Suite L layout editor and graphics editor environment variable names, descriptions, types, and values. Graphic environment variables control the characteristics of the window display and layout environment variables control how various layout editor commands work. Many graphic environment variables have duplicate layout environment variables. In these cases, the layout variable supersedes the graphic variable unless the graphic variable is stored in the cellview. You can set both types of environment variables: graphic and layout.

Note: Only the environment variables documented in this chapter are supported for public use. All other Layout editor environment variables, regardless of their name or prefix, and undocumented aspects of the environment variables described below, are private and are subject to change at any time.

Checking the Value of an Environment Variable

To determine the current value of an environment variable,

➤ Type in the CIW

```
envGetVal( "layout" "variable_name" )
```

or

```
envGetVal( "graphic" "variable_name" )
```

For example

```
envGetVal( "layout" "accessEdgesOn" )
```

or

```
envGetVal( "graphic" "accessEdgesOn" )
```


© 2020 All Rights Reserved.

How the System Evaluates Environment Variables

The system looks for the value of an environment variable as follows:

1. If no layout license is available the system looks for the system-defined default settings in the following file:

   `install_dir/etc/tools/graphic/.cdsenv`

2. If a layout license is available the system looks for the system-defined default settings in the following file:

   `install_dir/etc/tools/layout/.cdsenv`

3. Then the system loads any values in the local `.cdsenv` file in your home directory,

   `~/.cdsenv`

   The layout variables supersede any matching graphic variables, and environment variables defined in your `.cdsenv` supersede any default values in the graphic or layout `.cdsenv`.

Note: There is a precedence of window settings which might override environment variable settings. The order in which the settings are read is: cellview properties, library properties, technology library properties, and lastly, environment variable values. Therefore, any override window properties stamped on a cellview will take precedence over a setting in the `.cdsenv` file.

Graphic Environment Variables

Graphic environment variables control the window display. These variables can be stored in the cellview or the `.cdsenv` file. There are several ways to change the default settings of graphic environment variables:

- Set the variable in the Display Options Form.

  If you want to retain these settings for future sessions, save the settings using the *Save* button in the *Options – Save Defaults* form in the CIW.

- Copy the variables you want to change from the default `.cdsenv` file to the `.cdsenv` file in your home directory.

  It is there that you edit the default value. The default file is

  `your_install_dir/tools/dfII/samples/.cdsenv`

  Note: Do not use the `.cdsenv` system files located in the following directory:

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

© 2020 All Rights Reserved.

These files contain system default values for all applications and have extra arguments that do not work in your environment.

- Set the variable in the CIW using

  `envSetVal("graphic" "name" type value)`

  For example:

  `envSetVal("graphic" "iconsOn" 'boolean t)`

  To determine the current value of a graphic environment variable, type in the CIW

  `envGetVal("graphic" "name")`

  For example:

  `envGetVal("graphic" "iconsOn")`

Layout Editor Environment Variables

Layout environment variables control how various layout editor commands work. There are several ways to change the default settings of layout environment variables:

- Set the variable in the Layout Editor Options Form. If you want to retain these settings for future sessions, save the settings in the Save Defaults form in the *Options* menu in the CIW.
- Include the environment variables in your `.cdsenv` file in your home directory. The default file is

  your_install_dir/tools/dfII/samples/.cdsenv
- Include the `envSetVal()` function in your `.cdsinit` file.
- Include the `envSetVal()` function in any other SKILL file you load.
- Type the `envSetVal()` function in the CIW.

For example, to set the `gravityOn` variable, type the following in the CIW or include it in a file:

```
envSetVal("layout" "gravityOn" 't)
```

To determine the current value of any layout editor environment variable, type the `envGetVal()` function in the CIW as follows:

```
envGetVal("layout" "variable_name")
```


© 2020 All Rights Reserved.

Copying Environment Variables from the Samples Directory to ~/.cdsenv

Note: Do not copy lines from the `graphic/.cdsenv` or `layout/.cdsenv` files to the `.cdsenv` file in your home directory. The file formats are slightly different, so the lines that result from copying do not work.

Copy the default environment variables from the sample `.cdsenv` file located at:

your_install_dir/tools/dfII/samples/.cdsenv

Copy the variables you want to change from the sample `.cdsenv` file to the `.cdsenv` file in your home directory, and edit the default values in your `.cdsenv` file.

Setting the variables in the *.cdsenv* file in your home directory ensures that these environment variables remain set to your specification each time you open a layout editor session.

Changing Settings for Environment Variables in the CIW

Set the variable in the Command Interpreter Window with the command

envSetVal("graphic" *"name" type value*)

For example:

```
envSetVal("graphic" "iconsOn" 'boolean t)
```

or

envSetVal("layout" *"name" type value*)

For example:

```
envSetVal("layout" "iconsOn" 'boolean t)

envSetVal("layout" "chopShape" 'cyclic "rectangle")

envSetVal("layout" "alignSeparation" ' 2.0)

envSetVal("layout" "contactColumns" 'int 5)

envSetVal("layout" "filterSizeDrawingStyle" 'string "outlined")
```

If you set the environment variable by this method prior to opening a cellview window, the next window opened in the session will reflect the change. Any environment variable reset with this method takes effect after any open cellview window is closed and reopened, and in all subsequent windows that are opened in the current session. You must reset the environment variable in the current session to change it. Changes are effective for the current session only.

© 2020 All Rights Reserved.

Note: There are a number of environment variables that are no longer compatible with `leSetEnv` and `leGetEnv`. The setting methods for the environment variables are noted in this Appendix under the heading Setting Method, which is included with the description of each environment variable. Cadence recommends that you use only `envSetVal` and `envGetVal` to determine the setting of, set, or reset environment variables.

Changing Graphic Environment Variables That Are Cellview Properties

Some graphic environment variables are cellview properties. If you want the changes to the environment variable to be visible in the cellview window immediately, set these variables in the Command Interpreter Window using

```
window = hiGetCurrentWindow()
window~>environmentVariable = value
```

For example:

```
window = hiGetCurrentWindow()
window~>displayStretchHandles = t
```

This changes the environment variable for the current window only, and you must reset it if you open new windows. The change is effective for the current session only. The setting method for the window environment variables are noted in this Appendix under the heading Setting Method, using the word window, which is included with the description of each environment variable.

Environment Variables in Forms

Some of the environment variables correspond to fields in the layout editor's forms. The forms corresponding to the environment variables, if any, are noted in this Appendix under the heading GUI Equivalent, which is included with the description of each environment variable. The usual method of changing these options during an editing session is to change the options in the form. If you want to retain these settings for future sessions, save the settings in the *CIW – Options – Save Defaults* command.

Improving Performance Using Environment Variables

To enhance layout editor performance the following layout environment variables have new default values as of ICOA 5.0.33 and IC 5.0.33. Though performance is data and hardware dependent, these environment variables should reduce the time needed for redrawing the cellview window, panning, and zooming in and out, as compared to IC 4.4.6.

© 2020 All Rights Reserved.

The behavior of `filterSize` and `filterSizeDrawingStyle` are changed to display the largest and most essential data, while filtering out smaller shapes, and reducing overall display time.

The `filterSize` default is changed from `3.0` to `6.0`. The behavior of this environment variable is also changed:

- **Shapes:** a shape is filtered if two sides of its bounding box are smaller than the `filterSize` setting. In this case the shape is not displayed.
- **Instances**
  - An instance with one side of its bounding box smaller than the `filterSize` setting displays as an empty rectangle
  - An instance with two sides of its bounding box smaller than the `filterSize` setting displays according to the modified `filterSizeDrawingStyle` behavior (see below).

The `filterSizeDrawingStyle` default is changed from `outlined` to `empty`. The behavior of this environment variable is also changed:

- **Shapes:** the display of shapes is no longer altered by the `filterSizeDrawingStyle` setting
- **Instances:** with two sides of the bounding box smaller than the `filterSize` setting display according to the display stop level setting:
  - When instances are above stop level, the instance bounding box displays as a filled rectangle when `filterSizeDrawingStyle` is set to `filled`. The instance bounding box displays as a hollow rectangle when `filterSizeDrawingStyle` is set to `outlined`. The instance bounding box is not displayed when `filterSizeDrawingStyle` setting is set to `empty`.
  - When instances are at stop level the display of instances is no longer altered by the `filterSizeDrawingStyle` setting. An instance with two sides of its bounding box larger than the `filterSize` is drawn as an empty rectangle.

The `gravityOn` default is changed from `t` to `nil`. Gravity turned off will speed pointer response time during selection and movement.

The `openToStopLevel` is changed from `nil` to `t`. When set to `t` all data is loaded to the window stop level (corresponding to the `displayStopLevel` variable), then rendered. All data is loaded, even data too small to see. When set to `nil` data loads on demand during rendering. This will increase initial loading time, but usually speeds up subsequent zoom and pan operations.

© 2020 All Rights Reserved.

The `dotsOn` default is changed from `t` to `nil`. Neither the bounding box, nor the diamond origin for dots are displayed when `dotsOn` is set to `nil`. Dots are often used for pin shapes. This will shorten overall display time and will decrease screen clutter if you do not need to see dots or dot pins.

The `displayStretchHandles` default is changed from `t` to `nil`. If you do not need to interactively stretch Pcells, this will shorten the overall display time and will decrease screen clutter.

The `originMarkersOn` default is changed from `t` to `nil`. Origin markers of instances will not appear and this will shorten overall display time.

The `useTrueBBox` is changed from `t` to `nil`. Only the instance bounding box displays, which speeds up opening and redrawing times since only the top level data is considered. When set to `nil` the master bounding box is drawn, or the layer purpose "`instance`" "`drawing`" is used. When set to `t`, one more level of hierarchy is opened and evaluated, which can drastically effect loading and display time.

The following environment variables correspond to fields on the Display Options form under Display Controls: `dotsOn` (Dot Pins), `displayStretchHandles` (Stretch Handles), `openToStopLevel` (Open to Stop Level), `originMarkersOn` (Instance Origins), and `useTrueBBox` (Use True BBox). The following environment variables correspond to fields on the Display Options form under Filter: `filterSize` (Size), and `filterSizeDrawingStyle` (Style). They can be saved to, loaded from or deleted from a cellview, a library, a technology library or a `.cdsenv` using this form, however all the settings on this form will also be saved, loaded or deleted.

Note: The new default settings will not take effect if these environment variables are set to other than the default settings as properties of a cellview, a library, a technology library, or are set to values other than the default in the `.cdsinit` file or the `.cdsenv` files. The default value will be overwritten by these settings in the order in which they are listed. For example, any override window properties stamped on a cellview will take precedence over a setting in the `.cdsenv` file.

To find out if the settings of these environment variables are set as a cellview property,

1. From the cellview window, choose *File – Properties*.

   The Edit Cellview Properties form appears.

2. Click the *Property* radio button.

   All cellview properties are listed in this form.

   You can leave the properties as is, or add. delete, or modify the properties in this form.

3. Click *OK*.

© 2020 All Rights Reserved.

To find out if the settings of these environment variables are set as a library property,

1. From the CIW, window choose *Tools – Library Manager*

   The Library Manager: <WorkArea> form appears.

2. In the Library list box, select a library.

3. Choose *Edit – Properties.*

   If no properties are assigned to the library, a *Library Property Viewer* text window appears with the notation “No properties” at the bottom.

   If properties are assigned to the cellview the View Property Editor form appears.

4. Scroll down to viewProps.

5. Click EXPAND.

6. The viewProps properties form appears.

7. Choose layout or maskLayout and click EXPAND.

8. The layout properties form appears.

   You can leave the properties as is, or add. delete, or modify the properties in this form.

9. Click *OK*.

To find out if the settings of these environment variables are set as a technology library property,

1. From the CIW window choose *Tools – Library Manager*

   The Library Manager: <WorkArea> form appears.

2. In the Library list box, select a technology file.

3. Choose *Edit – Properties.*

   If no properties are assigned to the technology library a Library Property Viewer text window appears with the notation “No properties” at the bottom.

   If properties are assigned to the technology library the Library Property Editor form appears.

   You can leave the properties as is, or add. delete, or modify the properties in this form.

4. Click *OK*.

To find the settings if environment variables are set in the `.cdsinit` or `.cdsenv`,

© 2020 All Rights Reserved.

1. In a UNIX window, cd to *your_install_dir*.
2. Textedit or vi the `.cdsinit` or `.cdsenv`

   You can leave the properties as is, or add. delete, or modify the properties in these files.
3. Save the file, or quit without saving if you do not wish to change the settings.

The `gravityOn` environment variable corresponds to a field on the Layout Editor Options form under Gravity Controls. It can be can be saved to or loaded from a file, such as the `.cdsenv`. It will not be found in other locations, and will not be overwritten if it is in your local `.cdsenv`.

In summary:

Environment Variable	Old	New	Effect
`dotsOn`	`t`	`nil`	Dots are not displayed.
`displayStretchHandles`	`t`	`nil`	Stretch handles are not displayed.
`filterSize`	`3.0`	`6.0`	Objects with one side of less than 6 pixels are not displayed. Instances not at the stop level with two sides of less than 6 pixels are displayed according to the `filterSizeDrawingStyle` setting.
`filterSizeDrawingStyle`	`outlined`	`empty`	Now used only for instances which are not at the display stop level. When set to `empty`, instances are not displayed.
`gravityOn`	`t`	`nil`	Gravity is turned off.
`openToStopLevel`	`nil`	`t`	All data is loaded prior to rendering.
`originMarkersOn`	`t`	`nil`	Instance origin markers are not displayed.
`useTrueBBox`	`t`	`nil`	Displays only the instance master bounding box.

© 2020 All Rights Reserved.

List of Layout Editor Environment Variables

This section provides the description of the graphic and layout editor environment variables in Virtuoso® Layout Suite Editor.

- Graphic Environment Variables
- Layout Editor Environment Variables
- Environment Variables Used Internally

© 2020 All Rights Reserved.

Graphic Environment Variables

accessEdgesOn

addDesignLppsToTech

alphaBlendingBoundaryFillValue

alphaBlendingBoundaryHaloValue

alphaBlendingBoundaryOutLineValue

alphaBlendingFinFillValue

alphaBlendingFinHaloValue

alphaBlendingFinOutLineValue

alphaBlendingTrackFillValue

alphaBlendingTrackHaloValue

alphaBlendingTrackOutLineValue

arrayDisplay

askSaveOnReturn

autoStoreSelectionSetOn

cursorShape

defaultNewCellName

defaultNewLibName

defaultNewViewName

defaultOpenCellName

defaultOpenLibName

defaultOpenViewName

defaultToolName

© 2020 All Rights Reserved.

dimmingAutoEnabled

dimmingIntensity

dimmingOn

dimmingScope

discardCVEditOnWindowClose

displayFigGroupShapes

displayMissingMasterMarkers

displayPCellOrientation

displayPinNames

displayPRHighestResolution

displayResolution

displayRowName

displayRowSite

displayRulerMiddlePoint

displaySchDashText

displayStretchHandles

displayViaShapes

dotsOn

dotStyleMajorGrid

drawAxesOn

drawDottedGridOn

drawEngine

drawGridOn

drawInstancePinNames

© 2020 All Rights Reserved.

drawInstancePins

drawLabelsSurround

drawPRBoundary

drawSmallLabelOrigin

drawSurroundingOn

drawTraversedInstBbox

drawTraversedInstName

drfPath

dynamicHilightOn

filterSizeForResolution

fullSelectionMode

gridMultiple

gridSpacing

iconsOn

instanceDrawingMode

instLabel

leCheckMissingLayers

leCheckMissingMasters

leDispNetExpr

markerAccessMode

markerDeleteAccessModeIsEdit

markerDeleteConfirmModeChange

markerDeleteScope

markerDeleteSeverities

© 2020 All Rights Reserved.

markerDeleteSources

markerDeleteSourcesMode

markerSearchBy

markerSearchScope

markerSeverity

markerZoom

markerZoomScale

maskLayoutDefaultApp

maxAutoStoreSelectionObj

maxAutoStoreSelectionSet

maxPathSegInSpineAreaSelection

maxPathSegInSpineAreaSelection

mergeEnable

mergeScope

mergeVias

mergeXStippleShapes

meshNBThread

multiThreadInfo

netBasedSelectionNetName

netBasedSelectionOn

netBasedSelectionReverse

netExprDisplay

netNameDisplayDrawBackground

netNameDisplayDrawOnTop

© 2020 All Rights Reserved.

netNameDisplayEnabled

netNameDisplayFilter

netNameDisplayFilterDefaultList

netNameDisplayFilterLayers

netNameDisplayFilterMode

netNameDisplayFontStyle

netNameDisplayMaxLabelSize

netNameDisplayMaxLabelSizeFull

netNameDisplayNetShow

netNameDisplayScope

netNameDisplaySkipInstances

netNameDisplaySkipPcells

netNameDisplaySkipLayers

netNameDisplaySkipVias

netNameDisplayTermShow

netNameDisplayTermSkipInstances

netNameDisplayTermSkipPcells

netNameDisplayUserColor

netNameDisplayUseShapeColor

netsOn

openToStopLevel

originMarkersOn

overrideFontName

packetDialogBoxes

© 2020 All Rights Reserved.

partialSelect

partialSelectProximityMaxPixel

partialSelectionMode

pathCL

pathWidth

photonicDisplay

photonicPinFacetInPacket

photonicPinFacetOutPacket

photonicPinLabelPacket

photonicPinRadiusLinePacket

photonicPinWidthLinePacket

pointSelectAperture

pointSelectCycleMode

probeCycleBool

probeFilledInstance

probeFilledInstanceParam

probeHighlight

probeHiliteLppList

probeHiliteLppString

probeInstDisplayRadio

probeNetDisplayCheckInstance

probeNetDisplayCheckPins

probeNetDisplayFlightLines

probeNetDisplayHierarchyLevel

© 2020 All Rights Reserved.

probeNetNameFilterByName

probeNetNameFilterByNameDefaultList

probeNetNameFilterByNameMode

probeNetStopAtDisplayLevel

probePromptBusBit

probeShapeDisplay

probeTermDisplayRadio

probeVisibleLpp

protectedObjHighlightHaloLevel

protectedObjHighlightHaloMode

protectedObjHighlightHaloTrans

rulerColorPacketName

protectedObjHighlightHaloWidth

schematicDefaultApp

schematicDefaultTier

schematicSymbolDefaultApp

scrollPercent

segSnapMode

selectHilightPacket

selectionAperture

selectMarkersAtLast

selectObjectLargerThanWindow

selectObjectInEditCvFirst

selectProtectedItems

© 2020 All Rights Reserved.

selectUnderCursorMenuAlphaSorting

selectUnderCursorMenuLimit

slotEnable

snapMode

snapPatternFastDetailModeSnapping

snapPatternGlobalGridAnchoring

spineSelection

startLevel

statusBarXYdXdYLength

stopLevel

stretchHandlesLayer

textJustificationOn

trimEnable

tinyInstDetail

trueColorProbe

turnInfixOffWhenSmartSnapping

useAlphaBlending

useDynamicHilightPacket

useOverrideFontName

useTrueBBox

viaStackSelection

viaStackSelectionMatchingSizes

viaUseRowColDefault

viaValidateDRCStandalone

© 2020 All Rights Reserved.

viewNameToTypeList

xSnapSpacing

ySnapSpacing

© 2020 All Rights Reserved.

Layout Editor Environment Variables

abutServerOn

accessEdgesOn

acuteAnglePath

alignApplySpacing

alignDirection

alignLayer

alignMatchRefLayer

alignMatchRefSize

alignMethod

alignObjType

alignReference

alignRefMethod

alignRefObjType

alignReverseSort

alignSortMethod

alignSpacingType

alignSpacingValue

alignToolBarReferenceObject

alignUpdateLabelLayer

alignUsing

alignUsingInstanceDrawingMode

allowPartialDelete

© 2020 All Rights Reserved.

anchor

applyBounce

arrayDisplay

autoCopyTemplateCellViewList

autoCircleCreation

autoExtractSaveEnabled

autoLoadBindKeys

autoModePinSize

autoModePinSizeFromPath

autoNetName

autoPermutePins

autoPinLayerFunctionsList

autoPinRectHeight

autoPinRectWidth

autoRefresh

autoSetRefPoint

autoTapShape

autoTapType

autoViaOverrideVariantParameters

autoViaOverrideVariantParameters

autoZoomMode

autoZoomPan

autoZoomScale

balloonFadeTime

© 2020 All Rights Reserved.

balloonOn

balloonPauseTime

balloonTransparency

bannerFields

blockageAllowPGNet

blockageAttachOwner

blockageAttributeMode

blockageCreateMethod

blockageDerivePoints

blockageIsPushedDown

blockageKeepOwner

blockageMaterial

blockageMaxDensity

blockageSpacingType

blockageSpacingValue

blockageType

bondPadLPP

chainMode

checkAllSPGridsEnclosures

checkHalfWidthPathSeg

checkLargeCoordinates

chopArray

chopCircleJustify

chopCircleRadius

© 2020 All Rights Reserved.

chopLineJustify

chopLineType

chopLineUserWidth

chopPRBDialog

chopRectHeight

chopRectJustify

chopRectWidth

chopShape

chopSpecifySize

chopSpecifySpacing

circleRadius

circleShapeMode

clusterBoundaryAspectRatio

clusterBoundaryUtilization

confirmDelete

contactColumns

convertEachInstanceIntoMosaic

copyColumns

copyConnectivity

copyIncrementInstName

copyIncrementNetName

copyResetRowCol

copyRows

copyToLayer

© 2020 All Rights Reserved.

createAsLabel

createInstResetCDF

createPinPathSpineAware

createPinSchematicAware

createPinLabel

createRectilinearBoundary

createViaAlwaysUsesDefinedCalcMode

cursorShape

defCreatePinConnModel

delDetachedBlockages

deleteFluidShapes

deleteTarget

descendPromptForView

dimmingAutoEnabled

dimmingIntensity

dimmingOn

dimmingScope

disableLayerFuncSnapToMfgGrid

disableMissingLayerPopUp

disableRoundPathStyle

displayInstanceOrientation

displayMissingMasterMarkers

displayOptionsFile

displayPinNames

© 2020 All Rights Reserved.

displayPRHighestResolution

displayRefPoint

displayResolution

displayRowName

displayRowSite

displaySelectionHighlightWhenDimmed

displaySigDigits

displayStartLevel

displayStopLevel

displayStretchHandles

displayViaShapes

doNotShowChopPRBDialog

dotsOn

dotStyleMajorGrid

drawAxesOn

drawDottedGridOn

drawGridOn

drawInstancePins

drawSurroundingOn

drawTraversedInstBbox

drawTraversedInstName

drdEditSmartSnapAllowedWidthSnap

drfPath

dynamicHilightOn

© 2020 All Rights Reserved.

dynamicMeasurementBackgroundColor

dynamicMeasurementDrawBackground

dynamicMeasurementDisplay

dynamicMeasurementEdge

dynamicMeasurementFontSize

dynamicMeasurementFromCursor

dynamicMeasurementLayerScope

dynamicMeasurementTextColor

dynamicProbe

editInstanceBoundaryMode

editorOptionsFile

eipHighlightColor

enableAlignToolBarReferenceObject

enableCopyAndPaste

enableEIPEnhancements

enableSyncCloneEditing

expandPinBusExpression

extendPathLengthCheck

extractAfterGenerateAll

extractCellviewIllegalConnectionLimit

extractCellviewOpenViolationLimit

extractCellviewShortViolationLimit

extractCellviewUnnecessaryPseudoParallelConnectionLimit

extractConnectUnassignedInstTerms

© 2020 All Rights Reserved.

extractConsiderAllCutLayersInVias

extractEnabled

extractEnableVirtualConnections

extractHierExtractSearchDepth

extractKeepShapeAssignment

extractNetOpenViolationLimit

extractNetShortViolationLimit

extractReinitOnStopLevelChange

extractRemoveStickyProperty

extractRunOnInit

extractStopLevel

extractVerifyAnalogSigType

extractVerifyBigMPP

extractVerifyClockSigType

extractVerifyGroundSigType

extractVerifyIllegalHierConnections

extractVerifyIllegalOverlapsForStoppedColoredShapes

extractVerifyIncompleteMustConnectAllPins

extractVerifyInvalidConnections

extractVerifyMustConnectMarkers

extractVerifyOpenViolations

extractVerifyOpticalPortRadiusMatch

extractVerifyPowerSigType

extractVerifyResetSigType

© 2020 All Rights Reserved.

extractVerifyScanSigType

extractVerifySignalSigType

extractVerifyTieHighSigType

extractVerifyTieLowSigType

extractVerifyTieOffSigType

extractVerifyWellSubstrateConnections

extractVerifyUnimplementedInstTerms

extractVerifyUnnecessaryPseudoParallelConnections

extractVerifyWeakMarkers

extractVirtualConnectionSeparator

fitIntoClusterBoundary

fixedWidthPaths

flattenExcludePcellListFile

flattenMode

flattenPCells

flattenVias

flipAroundCombinedCenter

followRoutingDirection

ghostOfDraggedObjects

gravityDistance

gravityEditCellViewOnly

gravityOn

gravitySelectableOnly

gravityType

© 2020 All Rights Reserved.

gravityUse5PixelApertureWhenOff

grEnclosedBy

gridMultiple

haloBottomOffset

haloLeftOffset

haloRightOffset

haloTopOffset

highlightAlignedEdges

highlightAlignedEdgesColor

highlightAlignedEdgesWidth

iconsOn

ignoreBounce

ignoreParallelMinWidthForSquarePinResize

instanceDrawingMode

instanceSelectionMode

instanceVisibilityMode

instCellName

instColumns

instDelX

instDelY

instHalosEnabled

instLabel

instLibName

instMosaicCreation

© 2020 All Rights Reserved.

instName

instPhysicalOnly

instRows

instViewName

keepCopiedInsideGroup

keepFirstTermName

keepGuardRingEndsConnected

keepWiresConnectedToNonWires

labelAttach

labelAutoConn

labelAutoInst

labelDrafting

labelExpand

labelFontStyle

labelHeight

labelJustify

labelKeepLabel

labelLPP

labelMode

labelOverbar

labelPurpose

leChopPreservePathSeg

leDispNetExpr

leWindowBBox

© 2020 All Rights Reserved.

lockAngles

lppVisibilityMode

magAnchorPosition

magConfined

magCursorOffset

magDisplayMode

magFloatPosition

magOffsetCorner

magScale

magSize

maintainConnections

makeCellCellType

markerZoomScale

markNetAutoColorCycle

markNetConnectedLayer

markNetExcludePurposes

markNetHilightObj

markNetNumThreads

markNetOptionsFile

markNetRange

markNetSaveViaInfo

markNetSelectOverlapLayer

markNetStartLevel

markNetStopLayer

© 2020 All Rights Reserved.

markNetStopLevel

markNetThickline

markNetViaLayer

measurementDisplayOn

measurementTextHeight

mergeEnable

mergeScope

modalCommands

modCornerRadius

modCornerType

mppASCIIFileName

mppASCIIFileName

mppTemplate

mustConnectAllPins

muteUnstretchableOption

netBasedSelectionNetName

netBasedSelectionOn

netBasedSelectionReverse

netName

netsOn

nPtsToTaper

numLevels

numSides

openToStopLevel

© 2020 All Rights Reserved.

orientation

originMarkersOn

overrideFontName

padLabelLPP

partialSelect

pasteRef

pathBeginExt

pathCL

pathEndExt

pathJustify

pathOffset

pathStyle

pathWidth

photonicPinWidth

photonicPinAngle

photonicPinWidthLinePacket

pinAccessDir

pinDefaultObjectType

pinLabelPlacementMode

pinLayerPurpose

pinLPP

pinMode

pinSelectionMode

pinShape

© 2020 All Rights Reserved.

pinTextCreateAsLabel

pinTextLayer

pinTextLayerNames

pinTextPurpose

pinTextPurposeNames

pinTextSamePurpose

pinTextSameLayer

pinTextOrientation

pinWidth

pinXPitch

pinYPitch

polygonArc

preservePins

preservePinFigs

preserveRodObjects

preserveTerminalContacts

preserveTermName

preventDeletionOfLogicalInstance

probeNetNameFilterByNameMode

propEditHideTopSection

propEditorHeight

propEditorWidth

propertyEditorAutoResize

propEditStretchableTree

© 2020 All Rights Reserved.

pteAllLayersScope

pteApplyParentState

pteAutoRedrawMode

pteCheckableColumnClickMode

pteCollapseLayersInViewBy

pteColoredLppSortOrder

pteColorMixedSort

pteCompactMPT

pteDeferRedrawOnMouseMove

pteDisplayActiveLppInForeground

pteEnableMouseBindings

pteFindModeOn

pteIncludeCutLayers

pteInfraDefValueObjGrids

pteInfraObjSelFollowVis

pteInverseColorScheme

pteLayerSetActivationMode

pteLayerSetManagerLeftClickMode

pteLayersSelectabilityDepth

pteLayersVisibilityDepth

pteLinkColorVisibilityToShapeVisibility

pteLppGroup

pteLppsDisplayOrderControl

pteLSManagerAsAssistant

© 2020 All Rights Reserved.

pteLSManagerRespectFilters

pteMPTMode

pteMPTScope

pteMovableTools

pteMultiSelectionMode

pteRetainFilterSettings

pteRoutingOnlyScope

pteShowColorControlsForMPT

pteShowMPTScope

pteShowUsedSystemLpps

pteUsedLayersEipScope

pteWindowSynchronizationMode

QAHoldTimeMilliSec

QAMaxPixelsPerMilliSec

quickAlignMode

quickAlignSameLayerTarget

quickAlignSnapTarget

quickAlignSpacingType

quickAlignUserSpacing

quickAlignUserSpacing

realColorSelection

realignLabelToPinCenter

rectangularWSPPinAsSquare

rectBaseMode

© 2020 All Rights Reserved.

rectHeight

rectOnSnapPatternGrid

rectSmartSnapEnable

rectSmartSnapOnActiveLayer

rectWidth

recursionCheck

remasterInstCheckTerms

remasterInstSearchCellText

remasterInstSearchLibText

remasterInstSearchViewText

remasterInstUpdateCellText

remasterInstUpdateLibText

remasterInstUpdateViewText

removeChopShape

removeEmptyNetTerm

removeTerminals

replaceFigs

resetOrientation

reshapeType

retainSelectionSet

rodAutoName

rotateAroundCombinedCenter

rowAbutFirstUnit

rowAbutNumberOfUnits

© 2020 All Rights Reserved.

rowAbutUnit

rowBaseName

rowCreateMethod

rowChannelWidth

rowFirstUnitOrient

rowFlipFirstUnit

rowFlipUnit

rowNumberOfRows

rowPartitionBottomOffset

rowPartitionLeftOffset

rowPartitionRightOffset

rowPartitionTopOffset

rowNumberOfSites

rowSiteEnable

rowUnitOrient

rulerDisplayType

rulerEdgeMode

rulerFontBold

rulerFontSize

rulerLabelBackgroundColor

rulerSegmentModeType

saveRulers

scrollPercent

segSnapMode

© 2020 All Rights Reserved.

selectCreatedObj

selectionAperture

selectOnlyInstWithSelectableShapes

setDxDyToZero

setupConstraintGroup

showWhatsNew

slotEdgeToSlotOffset

slotEnable

slotEnableRect

slotEnablePath

slotInterSlotSpacing

slotLength

slotRowSpacing

slotShapeMode

slotSpecifyArea

slotStaggerOffset

slotWarnWhenMoreThanRows

slotWidth

snapDeviceInDetailMode

snapMode

snapPatternDrawingStyle

snapPatternSnapping

snapPatternSnappingMode

snapPatternSnapToXYGridFirst

© 2020 All Rights Reserved.

snapRectToWSPGrid

snapResizeWireObjects

snapToMfgGrid

softFence

startLevel

statusBarXYdXdYLength

statusToolbarFields

stopLevel

stretchPcellApplyToName

stretchViaMode

supressFreezePcellUI

tapConnectedFigsOnly

taperStyle

tapLayerFilter

tapPurposeList

tapType

targetColor

targetLineStyle

targetWidth

termDirection

termName

textJustificationOn

tinyInstDetail

transparentGroup

© 2020 All Rights Reserved.

trimEnable

trlBendFac

trlBendStyle

trlChamFac

trlNPtsToPI

trlRadFac

trlWidth1

trlWidth2

turnInfixOffWhenSmartSnapping

undoRedoHistoryLimit

updateLabelLPPWithPin

updatePCellIncrement

useDynamicHilightPacket

useOverrideFontName

useRelativeXY

useSeparateLAndXIForms

useTapPurposeList

useTrueBBox

validLppFilterOn

viaAlignCuts

viaAutoViaAlignment

viaAutoViaCutArrayDirection

viaAutoViaEnableMaxStackDepth

viaAutoViaEnclosureDirection

© 2020 All Rights Reserved.

viaAutoViaMaxStackDepth

viaAutoViaMinNumCuts

viaCheckBlockageTypes

viaCheckMinNumCut

viaCheckPlateMaxWidth

viaCheckPlateMinWidth

viaCheckShapeConn

viaColumns

viaCompuEnclosures

viaComputeFromMode

viaConstraintGroup

viaCreateInRoute

viaCutSpacingMethod

viaCutClassOrientation

viaEnableUserSkillCallbacks

viaEnableValidPurposes

viaExtendEnclosureBeyondOverlap

viaFillEnclosureOverlap

viaHideEnclosures

viaIgnorePurposes

viaMode

viaParamCalcMode

viaPreventDRCOnOverlap

viaPreventDRCWithNeighbors

© 2020 All Rights Reserved.

viaPrintStatisticsReport

viaPrintStatisticsReportByStackDepth

viaRows

viaSelectionMode

viaStackColumns

viaStackRows

viaUseEntireOverlap

viaUseRowColDefault

viaValidPurposes

viaWallFromSelectedShapesEnabled

viaWallKeepViasEnabled

viaWallPreviewEnabled

xSnapSpacing

ySnapSpacing

yankShape

zoomToFig

© 2020 All Rights Reserved.

abutServerOn

```
layout abutServerOn boolean { t | nil }
```

Description

Turns automatic abutment and automatic unabutment on and off.

When set to `t`, unabutment automatically restores abutted devices when moved apart. This ensures that dropped vias are restored when chained devices are moved apart. It also eliminates errors such as devices with no diffusion overlap on one side.

When set to `nil` in layout XL, automatic abutment and automatic unabutment are off.

When set to `nil` in the layout editor, unabutment retains the abutted device configuration even when the device is unabutted.

The default is `t`.

GUI Equivalent

Command	*Options – Editor*
Form Field	*Abut Server* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "abutServerOn")
envSetVal("layout" "abutServerOn" 'boolean t)
envSetVal("layout" "abutServerOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

accessEdgesOn

```
layout accessEdgesOn boolean { t | nil }
graphic accessEdgesOn boolean { t | nil }
```

Description

Determines whether you see the access edges of rectangular pins (`t`) or not. The default value is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Access Edges* (Display Options Form)

Examples

```
envGetVal("layout" "accessEdgesOn")
envSetVal("layout" "accessEdgesOn" 'boolean t)
envSetVal("layout" "accessEdgesOn" 'boolean nil)
envGetVal("graphic" "accessEdgesOn")
envSetVal("graphic" "accessEdgesOn" 'boolean t)
envSetVal("graphic" "accessEdgesOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

acuteAnglePath

```
layout acuteAnglePath boolean { t | nil }
```

Description

Allows creation of acute angle paths when set to `t`. Allows creation of paths from 90 degree to 270 degree angles when set to `nil`. You must set Snap Mode to anyAngle for this environment variable to take effect. The default is `t`.

GUI Equivalent

Command	*Create – Path*
Form Field	*Acute Angle* (Create Path Form)

Examples

```
envGetVal("layout" "acuteAnglePath")
envSetVal("layout" "acuteAnglePath" 'boolean t)
envSetVal("layout" "acuteAnglePath" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

addDesignLppsToTech

```
graphic addDesignLppsToTech boolean { t | nil }
```

Description

Creates a layer purpose pair (LPP) in the technology file when a shape is present in the design, however, its layer number and purpose number does not correspond to an existing LPP in the technology file. A default packet is used to display shapes on this new LPP.

If layer and purpose names associated with either the layer or purpose numbers, or both are not defined in technology file attached to sub cellviews of the hierarchy, then:

- Name of the new layer will be LXXX, where XXX is the layer number.
- Name of the new purpose will be PYYY, where YYY is the purpose number.

The feature is not activated by default and the default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "addDesignLppsToTech")
envSetVal("graphic" "addDesignLppsToTech" 'boolean t)
envSetVal("graphic" "addDesignLppsToTech" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignApplySpacing

```
layout alignApplySpacing boolean { t | nil }
```

Description

Sets the toggle button for the *Apply Spacing* field on the *Edit – Advanced – Align* form to on or off. When the value is on (`t`), the data entry area is enabled. The valid values are `t` and `nil`. The default value is `nil`. To set the value for the data entry area, use the alignSpacingValue environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Form Field	*Apply Spacing* (Align Form)

Examples

```
envGetVal("layout" "alignApplySpacing")
envSetVal("layout" "alignApplySpacing" 'boolean t)
envSetVal("layout" "alignApplySpacing" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignDirection

```
layout alignDirection string { horizontal | vertical }
```

Description

Sets the *Horizontal* and *Vertical* radio buttons for the *Align Direction* field in the *Reference* section of the *Edit – Advanced – Align* form. The valid values are `horizontal` and `vertical`. The default is `vertical`.

GUI Equivalent

Command	*Edit – Advanced – Align*
Form Field	*Align Direction* (Align Form)

Examples

```
envGetVal("layout" "alignDirection")
envSetVal("layout" "alignDirection" 'string "horizontal")
envSetVal("layout" "alignDirection" 'string "vertical")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignLayer

`layout alignLayer string` *`layerPurpose`*

Description

Enables you to set up the default layer-purpose pair for the *Layer* list in the *Reference* and *Align Using* sections of the *Edit – Advanced – Align* form. By default, no layer purpose is set up.

GUI Equivalent

Command	*Edit – Advanced – Align*
Form Field	*Layer* (Align Form)

Examples

```
envGetVal("layout" "alignLayer")
envSetVal("layout" "alignLayer" 'string "Metal1 drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignMatchRefLayer

```
layout alignMatchRefLayer boolean { t | nil }
```

Description

Lets you change the target pin layer purpose of the pin to the reference pin layer purpose. The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Edit – Advanced – Align*
Field	*Match layer/purpose with the reference pin* (Align Form)

Examples

```
envGetVal("layout" "alignMatchRefLayer")
envSetVal("layout" "alignMatchRefLayer" 'boolean t)
envSetVal("layout" "alignMatchRefLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignMatchRefSize

```
layout alignMatchRefSize boolean { t | nil }
```

Description

Lets you change the size of the target pin to that of the reference pin. The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Edit – Advanced – Align*
Field	*Match size with the reference pin* (Align Form)

Examples

```
envGetVal("layout" "alignMatchRefSize")
envSetVal("layout" "alignMatchRefSize" 'boolean t)
envSetVal("layout" "alignMatchRefSize" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignMethod

```
layout alignMethod cyclic { top | bottom | left | right | center | origin | upperLeft
     | upperRight | upperCenter | lowerLeft | lowerRight | lowerCenter | centerLeft
     | centerRight | centerCenter }
```

Description

In the *Align Using* section on the *Edit – Advanced – Align* form, sets the values in the cyclic *Use* field for *Object* or *Layer*, depending on how the *Reference* and *Align Direction* fields are set, as shown in the table below.

When Reference is set to:	And Align Direction is set to:	And Align Using is set to:	The Align Using - Use cyclic list contains:	The default value for Use is:
Point	*Horizontal* or *Vertical*	*Object*	`Upper Left`, `Upper Center`, `Upper Right`; `Center Left`, `Center Center`, `Center Right`; `Lower Left`, `Lower Center`, `Lower Right`; `Origin`	`Lower Left`
		Layer	`Upper Left`, `Upper Center`, `Upper Right`; `Center Left`, `Center Center`, `Center Right`; `Lower Left`, `Lower Center`, `Lower Right`	`Lower Left`
Object	*Horizontal*	*Object*	`Top Edge`, `Bottom Edge`, `Center`; `Origin`	`Center`
		Layer	`Top Edge`, `Bottom Edge`, `Center`	`Center`
	Vertical	*Object*	`Left Edge`, `Right Edge`, `Center`; `Origin`	`Center`
		Layer	`Left Edge`, `Right Edge`, `Center`	`Center`
Layer	*Horizontal*	*Object*	`Top Edge`, `Bottom Edge`, `Center`; `Origin`	`Center`
Layer	*Vertical*	*Object*	`Left Edge`, `Right Edge`, `Center`; `Origin`	`Center`

© 2020 All Rights Reserved.

When Reference is set to:	And Align Direction is set to:	And Align Using is set to:	The Align Using - Use cyclic list contains:	The default value for Use is:
	Horizontal	*Layer*	`Top Edge`, `Bottom Edge`, `Center`	`Center`
	Vertical	*Layer*	`Left Edge`, `Right Edge`, `Center`	`Center`

You can set the *Align Using* field with the alignUsing environment variable and the *Reference* field with the alignReference environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Align Using – Object – Use* *Align Using – Layer – Use* (Align Form)

Examples

```
envGetVal("layout" "alignMethod")
envSetVal("layout" "alignMethod" 'cyclic "top")
envSetVal("layout" "alignMethod" 'cyclic "bottom")
envSetVal("layout" "alignMethod" 'cyclic "left")
envSetVal("layout" "alignMethod" 'cyclic "right")
envSetVal("layout" "alignMethod" 'cyclic "center")
envSetVal("layout" "alignMethod" 'cyclic "origin")
envSetVal("layout" "alignMethod" 'cyclic "upperLeft")
envSetVal("layout" "alignMethod" 'cyclic "upperRight")
envSetVal("layout" "alignMethod" 'cyclic "upperCenter")
envSetVal("layout" "alignMethod" 'cyclic "lowerLeft")
envSetVal("layout" "alignMethod" 'cyclic "lowerRight")
envSetVal("layout" "alignMethod" 'cyclic "lowerCenter")
envSetVal("layout" "alignMethod" 'cyclic "centerLeft")
envSetVal("layout" "alignMethod" 'cyclic "centerRight")
envSetVal("layout" "alignMethod" 'cyclic "centerCenter")
```


© 2020 All Rights Reserved.

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignObjType

```
layout alignObjType string { instance | row | cluster boundary | any shape | pins }
```

Description

When the *Align Using* field on the *Edit – Advanced – Align* form is set to *Object*, sets the object type to *Instance*, *Any Shape, Row*, *Cluster Boundary*, *Group*, or for Layout GXL, *Pins*. The valid values are: `instance`, `row`, `cluster boundary`, `group`, `any shape` and for the floorplanning mode in Layout GXL, `pins`. You can set the *Align Using* field with the alignUsing environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Align Using – Object* (Align Form)

Examples

```
envGetVal("layout" "alignObjType")
envSetVal("layout" "alignObjType" 'cyclic "instance")
envSetVal("layout" "alignObjType" 'cyclic "row")
envSetVal("layout" "alignObjType" 'cyclic "cluster boundary")
envSetVal("layout" "alignObjType" 'cyclic "any shape")
envSetVal("layout" "alignObjType" 'cyclic "group")
envSetVal("layout" "alignObjType" 'cyclic "pins")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignReference

```
layout alignReference string { point | object | layer }
```

Description

Sets the *Reference* field on the *Edit – Advanced – Align* form to *Point*, *Object*, or *Layer*. The valid values are `point`, `object`, or `layer`. The default is `point`. For `object`, instances, rows, cluster boundaries, and groups are supported in layout editor; additionally, pins are supported in Layout EXL.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Reference* (Align Form)

Examples

```
envGetVal("layout" "alignReference")
envSetVal("layout" "alignReference" 'string "point")
envSetVal("layout" "alignReference" 'string "object")
envSetVal("layout" "alignReference" 'string "layer")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignRefMethod

```
layout alignRefMethod cyclic { center | top | bottom | left | right | origin }
```

Description

When the *Reference* field on the *Edit – Advanced – Align* form is set to *Object* or *Layer*, sets the values in the cyclic *Use* field, based on whether *Align Direction* is set to *Horizontal* or *Vertical*, as shown in the table below.

When Reference is set to:	And Align Direction is set to:	The Use Cyclic list contains:	The default value is:
Object	*Horizontal*	`Top Edge`, `Bottom Edge`, `Center`, `Origin`	`Center`
	Vertical	`Left Edge`, `Right Edge`, `Center, Origin`	`Center`
Layer	*Horizontal*	`Top Edge`, `Bottom Edge`, `Center`, `Origin`	`Center`
	Vertical	`Left Edge`, `Right Edge`, `Center, Origin`	`Center`

The valid values are: `top`, `bottom`, `center`, `left`, `right`, and `origin`. The default value is `center`. You can set the *Reference* field with the alignReference environment variable, and the *Align Direction* field with the alignDirection environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Reference – Object – Use* *Reference – Layer – Use* (Align Form)

Examples

```
envGetVal("layout" "alignRefMethod")
envSetVal("layout" "alignRefMethod" 'cyclic "center")
envSetVal("layout" "alignRefMethod" 'cyclic "top")
```


© 2020 All Rights Reserved.

```
envSetVal("layout" "alignRefMethod" 'cyclic "bottom")
envSetVal("layout" "alignRefMethod" 'cyclic "left")
envSetVal("layout" "alignRefMethod" 'cyclic "right")
envSetVal("layout" "alignRefMethod" 'cyclic "origin")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignRefObjType

```
layout alignRefObjType cyclic { instance | row | cluster boundary | group | pins}
```

Description

When the *Reference* field on the *Edit – Advanced – Align* form is set to *Object*, sets the object type to *Instance*, *Row*, *Cluster Boundary*, or *Group*. The valid values are: `instance`, `row`, `cluster boundary`, `group`, and when in floorplanning mode in Layout GXL, `pins`. You can set the *Reference* field with the alignReference environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Reference – Object* (Align Form)

Examples

```
envGetVal("layout" "alignRefObjType")
envSetVal("layout" "alignRefObjType" 'cyclic "instance")
envSetVal("layout" "alignRefObjType" 'cyclic "row")
envSetVal("layout" "alignRefObjType" 'cyclic "cluster boundary")
envSetVal("layout" "alignRefObjType" 'cyclic "group")
envSetVal("layout" "alignRefObjType" 'cyclic "pins")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignReverseSort

```
layout alignReverseSort boolean { t | nil }
```

Description

Sets the toggle button for the *Reverse Sort* field on the *Edit – Advanced – Align* form to on (`t`) or off (`nil`). The valid values are `t` and `nil`. The default value is `nil`.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Reverse Sort* (Align Form)

Examples

```
envGetVal("layout" "alignReverseSort")
envSetVal("layout" "alignReverseSort" 'boolean t)
envSetVal("layout" "alignReverseSort" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignSortMethod

```
layout alignSortMethod cyclic { alignDirection | horizontal | vertical |
     orderSelected | instanceName | netName}
```

Description

Sets the sort order for the *Sort Objects by* field on the *Edit – Advanced – Align* form to *Align Direction Order*, *Horizontal Order*, *Vertical Order*, *Order Selected*, *Instance Name*, or *Net Name*. The valid values are: `alignDirection`, `horizontal`, `vertical`, `orderSelected`, `instanceName`, `netName`. The default value is `alignDirection`.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Sort Objects By* (Align Form)

Examples

```
envGetVal("layout" "alignSortMethod")
envSetVal("layout" "alignSortMethod" 'cyclic "alignDirection")
envSetVal("layout" "alignSortMethod" 'cyclic "horizontal")
envSetVal("layout" "alignSortMethod" 'cyclic "vertical")
envSetVal("layout" "alignSortMethod" 'cyclic "orderSelected")
envSetVal("layout" "alignSortMethod" 'cyclic "instanceName")
envSetVal("layout" "alignSortMethod" 'cyclic "netName")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignSpacingType

```
layout alignSpacingType string { spacings | pitches }
```

Description

On the *Edit – Advanced – Align* form, when the *Reference* field is set to *Point*, and the *Apply Spacing* field is turned on, sets the *Between Objects and Pitches* radio buttons. The valid values are `spacings` (for the *Between Objects* radio button) or `pitches`. The default value is `spacings`. You can set the *Apply Spacing* field with the alignApplySpacing environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Apply Spacing* (Align Form)

Examples

```
envGetVal("layout" "alignSpacingType")
envSetVal("layout" "alignSpacingType" 'string "spacings")
envSetVal("layout" "alignSpacingType" 'string "pitches")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignSpacingValue

`layout alignSpacingValue` *`floating_point_number`*

Description

When the *Apply Spacing* field on the *Edit – Advanced – Align* form is set to on, sets its value. The value must be a integer point number. The default value is `1.0`. You can set the *Apply Spacing* field with the alignApplySpacing environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Apply Spacing* (Align Form)

Examples

```
envGetVal("layout" "alignSpacingValue")
envSetVal("layout" "alignSpacingValue" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignToolBarReferenceObject

```
layout alignToolBarReferenceObject cyclic { left | top | bottom | right | vertical
     | horizontal | click }
```

Description

Selects the reference object from the selection set when using the *Align* toolbar. The valid values are `left`, `top`, `bottom`, `right`, `vertical`, `horizontal`, or `click`.

- `left`: The leftmost object from the selection set is used as the reference object
- `right`: The rightmost object from the selection set is used as the reference object
- `top`: The topmost object from the selection set is used as the reference object
- `bottom`: The bottommost object from the selection set is used as the reference object
- `horizontal`: The object closest to the horizontal center of the selection set bounding box is used as the reference object
- `vertical`: The object closest to the vertical center of the selection set bounding box is used as the reference object
- `click`: The clicked object is used as the reference object

The default is `left`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "alignToolBarReferenceObject")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "left")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "top")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "bottom")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "right")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "vertical")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "horizontal")
envSetVal("layout" "alignToolBarReferenceObject" 'cyclic "click")
```


© 2020 All Rights Reserved.

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignUpdateLabelLayer

```
layout alignUpdateLabelLayer boolean { t | nil }
```

Description

Updates the layer of labels to that of the pin. This option can be used when the layers of the pin and label are different.The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Edit – Advanced – Align*
Field	*Update pin label layer* (Align Form)

Examples

```
envGetVal("layout" "alignUpdateLabelLayer")
envSetVal("layout" "alignUpdateLabelLayer" 'boolean t)
envSetVal("layout" "alignUpdateLabelLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignUsing

```
layout alignUsing string { object | layer }
```

Description

Sets the *Align Using* field on the *Edit – Advanced – Align* form to *Object* or *Layer*. The valid values are `object` or `layer`. The default value is `object`.

- Setting this variable to `object` means that target objects of the type specified by the cyclic field will align to the reference point, object, or layer specified in the *Reference* section of the form.
- Setting this variable to *layer* means that target objects on the selected layer will align with the reference point, object, or layer specified in the *Reference* section of the form.

You can set the *Reference* field with the alignReference environment variable, the *Align Using* field with the alignObjType environment variable, and the cyclic field for object type with the alignObjType environment variable.

GUI Equivalent

Command	*Edit – Advanced – Align*
Field	*Align Using* (Align Form)

Examples

```
envGetVal("layout" "alignUsing")
envSetVal("layout" "alignUsing" 'string "object")
envSetVal("layout" "alignUsing" 'string "layer")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alignUsingInstanceDrawingMode

```
layout alignUsingInstanceDrawingMode boolean { t | nil }
```

Description

Aligns the instances based on how they are rendered on the canvas. Instances can be rendered on the canvas as bounding box, true bounding box, PR boundary, or snap boundary. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "alignUsingInstanceDrawingMode")
envSetVal("layout" "alignUsingInstanceDrawingMode" 'boolean t)
envSetVal("layout" "alignUsingInstanceDrawingMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

allowPartialDelete

```
layout allowPartialDelete boolean { t | nil }
```

Description

Specifies if deletion of partially selected objects is allowed (`t`) or not. The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "allowPartialDelete")
envSetVal("layout" "allowPartialDelete" 'boolean t)
envSetVal("layout" "allowPartialDelete" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingBoundaryFillValue

`graphic alphaBlendingBoundaryFillValue int` *`any _integer`*

Description

Sets the fill value for the alpha blending boundary. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingBoundaryFillValue")
envSetVal("graphic" "alphaBlendingBoundaryFillValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingBoundaryHaloValue

`graphic alphaBlendingBoundaryHaloValue int` *`any _integer`*

Description

Sets the halo value for the alpha blending boundary. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingBoundaryHaloValue")
envSetVal("graphic" "alphaBlendingBoundaryHaloValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingBoundaryOutLineValue

`graphic alphaBlendingBoundaryOutLineValue int any _integer`

Description

Sets the outline value for the alpha blending boundary. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingBoundaryOutLineValue")
envSetVal("graphic" "alphaBlendingBoundaryOutLineValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingFinFillValue

`graphic alphaBlendingFinFillValue int` *`any _integer`*

Description

Sets the fill value for the alpha blending fin. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingFinFillValue")
envSetVal("graphic" "alphaBlendingFinFillValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingFinHaloValue

`graphic alphaBlendingFinHaloValue int` *`any _integer`*

Description

Sets the halo value for the alpha blending fin. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingFinHaloValue")
envSetVal("graphic" "alphaBlendingFinHaloValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingFinOutLineValue

`graphic alphaBlendingFinOutLineValue int` *`any _integer`*

Description

Sets the outline value for the alpha blending fin. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingFinOutLineValue")
envSetVal("graphic" "alphaBlendingFinOutLineValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingTrackFillValue

`graphic alphaBlendingTrackFillValue int` *`any _integer`*

Description

Sets the fill value for the alpha blending track. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingTrackFillValue")
envSetVal("graphic" "alphaBlendingTrackFillValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingTrackHaloValue

`graphic alphaBlendingTrackHaloValue int` *`any_integer`*

Description

Sets the halo value for the alpha blending track. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingTrackHaloValue")
envSetVal("graphic" "alphaBlendingTrackHaloValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

alphaBlendingTrackOutLineValue

`graphic alphaBlendingTrackOutLineValue int` *`any_integer`*

Description

Sets the outline value for the alpha blending track. The default is `1000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "alphaBlendingTrackOutLineValue")
envSetVal("graphic" "alphaBlendingTrackOutLineValue" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

anchor

```
layout anchor cyclic { unknown | left | right }
```

Description

Adjusts the cellview placement of a chain of instances after the Subtract Finger or Add Finger command is used. The default is `left`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "anchor")
envSetVal("layout" "anchor" 'cyclic "unknown")
envSetVal("layout" "anchor" 'cyclic "left")
envSetVal("layout" "anchor" 'cyclic "right")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

applyBounce

```
layout applyBounce boolean { t | nil }
```

Description

Specifies if gravity offset (bounce) values are used (`t`) or not. If applyBounce is set to t, gravityOn is automatically set to `t`. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "applyBounce")
envSetVal("layout" "applyBounce" 'boolean t)
envSetVal("layout" "applyBounce" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

arrayDisplay

```
layout arrayDisplay string { full | border | source }
graphic arrayDisplay string { full | border | source }
```

Description

Determines how arrays are displayed. The default is `full`.

GUI Equivalent

Command	*Options – Display*
Field	*Array Display* (Display Options Form)

Examples

```
envGetVal("layout" "arrayDisplay")
envSetVal("layout" "arrayDisplay" 'string "full")
envSetVal("layout" "arrayDisplay" 'string "border")
envSetVal("layout" "arrayDisplay" 'string "source")
envGetVal("graphic" "arrayDisplay")
envSetVal("graphic" "arrayDisplay" 'string "full")
envSetVal("graphic" "arrayDisplay" 'string "border")
envSetVal("graphic" "arrayDisplay" 'string "source")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

askSaveOnReturn

```
graphic askSaveOnReturn boolean { t | nil }
```

Description

Determines whether a dialog box appears after the *Edit In Place* and *Descend Edit* commands, while in edit mode, and when changes have been made. The dialog box asks if you want to save your changes to a cellview. If set to `nil`, the dialog box does not appear and the cellview stays open in edit mode. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "askSaveOnReturn")
envSetVal("graphic" "askSaveOnReturn" 'boolean t)
envSetVal("graphic" "askSaveOnReturn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoCopyTemplateCellViewList

`layout autoCopyTemplateCellViewList string` *`any_cellview`*

Description

Copies the WSP and row region templates from the specified cellview to the new cellview being opened in the layout window. The copy operation is not performed when opening an existing layout or creating the cellview using database SKILL functions.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "autoCopyTemplateCellViewList")
envSetVal("layout" "autoCopyTemplateCellViewList" 'string "(lib1 cell1 layout)")
```

In the above example, when you open a new cellview, the WSPs and row region specifications from lib1:cell1:layout will be copied to the cellview being created.

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoCircleCreation

```
layout autoCircleCreation boolean { t | nil }
```

Description

Sets the *Mode* field to *Click To Place* when `t`, and *Draw* when `nil`. The default value is `nil`.

GUI Equivalent

Command	*Create – Shape – Circle*
Field	*Mode* (Create Circle Form)

Examples

```
envGetVal("layout" "autoCircleCreation")
envSetVal("layout" "autoCircleCreation" 'boolean t)
envSetVal("layout" "autoCircleCreation" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoExtractSaveEnabled

```
layout autoExtractSaveEnabled boolean { t | nil }
```

Description

Automatically saves the hierarchical cellviews extracted by the Extract Layout command when the extraction scope is set to *Current Cellview and Cellviews in Hierarchy*. The default is `t`.

For more information, see Hierarchical Extraction in the *Virtuoso Layout Suite XL User Guide*.

GUI Equivalent

Command	In VLS XL,
	Connectivity – Update – Extract Layout
Field	*Save Extracted Cellviews*

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoLoadBindKeys

```
layout autoLoadBindKeys boolean { t | nil }
```

Description

Loads the `Layout.il` file automatically when you run `virtuoso`. The `Layout.il` file is searched for in the following locations:

- `<install_dir>/share/cdssetup/dfII/bindkeys/Layout.il`
- `~/.cadence/dfII/bindkeys/<customBindKey>.il`
- `./.cadence/dfII/bindkeys/<customBindKey>.il`

The `Layout.il` file, if found, is loaded from each location in the order specified above. Each `Layout.il` file is appended to the one that was loaded last. If the `Layout.il` files contain any overlapping bindkey settings, the bindkey settings defined in the `Layout.il` file loaded last take effect.

The default value of the environment variable is `t`. When set to `nil`, the `Layout.il` files are not automatically loaded from the above specified paths.

GUI Equivalent

None

Examples

```
envGetVal("layout" "autoLoadBindKeys")
envSetVal("layout" "autoLoadBindKeys" 'boolean t)
envSetVal("layout" "autoLoadBindKeys" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoModePinSize

```
layout autoModePinSize boolean { t | nil }
```

Description

Specifies whether pin size needs to be set for the rectangular pin created in auto mode (`t`) or not (`nil`). The default value is `nil`.

GUI Equivalent

Command	*Create – Pin – Auto*
Field	*Pin Size* (Create Pin Form)

Examples

```
envGetVal("layout" "autoModePinSize")
envSetVal("layout" "autoModePinSize" 'boolean t)
envSetVal("layout" "autoModePinSize" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoModePinSizeFromPath

```
layout autoModePinSizeFromPath boolean { t | nil }
```

Description

When set to (`t`), the shape of the pin is the same as that of underlying path or pathSeg. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "autoModePinSizeFromPath")
envSetVal("layout" "autoModePinSizeFromPath" 'boolean t)
envSetVal("layout" "autoModePinSizeFromPath" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoNetName

```
layout autoNetName boolean { t | nil }
```

Description

Determines whether shapes are added to nets attached to pins (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Connectivity – Nets – Add Shape*
Field	*Auto* (Add Shape to Net Form)

Examples

```
envGetVal("layout" "autoNetName")
envSetVal("layout" "autoNetName" 'boolean t)
envSetVal("layout" "autoNetName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoPermutePins

```
layout autoPermutePins boolean { t | nil }
```

Description

Specifies whether pins are permuted automatically during manual routing or editing (`t`) or not (`nil`). The default is `t`.

Important

Although `autoPermutePins` is defined as a layout environment variable, it has no effect in layout editor. It is used only by XL.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Generation* tab
Field	*Permute pins* (Layout XL Options form)

Examples

```
envGetVal("layout" "autoPermutePins")
envSetVal("layout" "autoPermutePins" 'boolean t)
envSetVal("layout" "autoPermutePins" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoPinLayerFunctionsList

```
layout autoPinLayerFunctionsList string { metal1 metal2 metal3 }
```

Description

Specifies the list of layers for auto pin. The default is an empty list.

GUI Equivalent

None

Examples

```
envGetVal("layout" "autoPinLayerFunctionsList")
envSetVal("layout" "autoPinLayerFunctionsList" 'string "metal1 metal2")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoPinRectHeight

`layout autoPinRectHeight float` *`float_number`*

Description

Specifies the height of the rectangle. The default value is `0.0`.

GUI Equivalent

Command	*Create – Pin – Auto – Pin Size*
Field	*Height* (Create Pin Form)

Examples

```
envGetVal("layout" "autoPinRectHeight")
envSetVal("layout" "autoPinRectHeight" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoPinRectWidth

`layout autoPinRectWidth float` *`float_number`*

Description

Specifies the width of the rectangle. The default value is `0.0`.

GUI Equivalent

Command	*Create – Pin – Auto – Pin Size*
Field	*Width* (Create Pin Form)

Examples

```
envGetVal("layout" "autoPinRectWidth")
envSetVal("layout" "autoPinRectWidth" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoRefresh

```
layout autoRefresh boolean { t | nil }
```

Description

Specifies whether automatic checking of cellview needing refresh on edit in place or descend is on (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "autoRefresh")
envSetVal("layout" "autoRefresh" 'boolean t)
envSetVal("layout" "autoRefresh" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoSaveMarkNetSetupFile

```
layout autoSaveMarkNetSetupFile boolean { t | nil }
```

Description

Controls the saving of the `markNetOptions` file. The default is `t`.

GUI Equivalent

<NA>

Examples

```
envGetVal("layout" "autoSaveMarkNetSetupFile")
envSetVal("layout" "autoSaveMarkNetSetupFile" 'boolean t)
envSetVal("layout" "autoSaveMarkNetSetupFile" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoSetRefPoint

```
layout autoSetRefPoint boolean { t | nil }
```

Description

Determines whether reference points are set automatically whenever a new point is entered (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Auto Set Reference Point* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "autoSetRefPoint")
envSetVal("layout" "autoSetRefPoint" 'boolean t)
envSetVal("layout" "autoSetRefPoint" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoStoreSelectionSetOn

```
graphic autoStoreSelectionSetOn boolean { t | nil }
```

Description

Turns *Auto Store* on or off for *Auto Store Section Controls* in the Selection Options form, accessed by choosing *Options – Selection*. The default is `nil`.

GUI Equivalent

Command	*Options – Selection*
Form Field	*Auto Store Section Controls – Auto Store (*Selection Options Form)

Examples

```
envGetVal("layout" "autoStoreSelectionSetOn")
envSetVal("layout" "autoStoreSelectionSetOn" 'boolean t)
envSetVal("layout" "autoStoreSelectionSetOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoTapShape

```
layout autoTapShape boolean { t | nil }
```

Description

Controls whether a design-tap occurs on the first entry point (`t`) or not (`nil`), when you create rectangles, polygons, paths, circles, donuts, and ellipses. During the create command, when shape auto tap is on, the current entry layer will seed the form and update attributes of the object you create based on the first point you enter. The default is `nil`.

GUI Equivalent

Command	*Options – Layout Editor*
Field	*Auto Tap: Shape*(Layout Editor Options Form)

Examples

```
envGetVal("layout" "autoTapShape")
envSetVal("layout" "autoTapShape" 'boolean t)
envSetVal("layout" "autoTapShape" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoTapType

```
layout autoTapType boolean { t | nil }
```

Description

Controls how the auto tap command behaves. autoTapType is used to set whether attributes of the tapped objects will be used, and whether to switch to a different layer when tapping on an object that is on a different layer than the current layer.

When set to "`attributes layer`", both attributes and layers of objects are tapped. When set to "`layer` ", only layers are tapped. When set to "`attributes`", the tap command captures the beginning and ending extensions, the end type, the width.For trls, bends and tapers the tap command captures the bend factor (maximum width before they are chamfered), taper style (linear or exponential), number of points to taper in user units, the bend style (bend chamfer or radial), the width of a chamfered bend in a transmission line as a multiple of its standard width in user units, the number of segments in a radial bend in a transmission line as a function of the number of segments in a 180-degree radial bend, the radius of a radial bend in a transmission line as a multiple of the transmission-line width in user units, the width of the first end of a transmission line bend in user units, the width of the second end of a transmission line bend, and the width of the second end of a transmission line bend in user units.

Default is `layer attributes`.

GUI Equivalent

Command	*Options – Editor*
Field	*Tap Layer* and *Tap Attributes* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "autoTapType")
envSetVal("layout" "autoTapType" 'boolean t)
envSetVal("layout" "autoTapType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoViaOverrideVariantParameters

```
layout autoViaOverrideVariantParameters boolean { t | nil }
```

Description

Enables the via variant parameter values to be updated in the *Single*, *Stack*, and *Auto* via creation modes. The default value is `nil`.

GUI Equivalent

Command	*Create – Via, Options* button in the *Create Via* form
Field	*Override Variant Parameters* (Create Via Options Form)

Examples

```
envGetVal("layout" "autoViaOverrideVariantParameters")
envSetVal("layout" "autoViaOverrideVariantParameters" 'boolean t)
envSetVal("layout" "autoViaOverrideVariantParameters" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoZoomMode

```
layout autoZoomMode cyclic { Pan | Zoom | ZoomToFit }
```

Description

Lets you specify the dynamic zoom setting to be used . You can specify `Pan`, `Zoom`, `ZoomToFit`. The default is `Pan`.

GUI Equivalent

Command	*Window – Assistants – Navigator* or *Options – Display*
Field	*Mode* (Display Options Form)

Examples

```
envGetVal("layout" "autoZoomMode")
envSetVal("layout" "autoZoomMode" 'cyclic "Pan")
envSetVal("layout" "autoZoomMode" 'cyclic "Zoom")
envSetVal("layout" "autoZoomMode" 'cyclic "ZoomToFit")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoZoomPan

```
layout autoZoomPan boolean { t | nil }
```

Description

Turns on or off the automatic zooming or panning to objects you select in the *Navigator* assistant. The default is `t`.

GUI Equivalent

Command	*Window – Assistants – Navigator* or *Options – Display*
Field	*Dynamic Zoom* (Display Options Form)

Examples

```
envGetVal("layout" "autoZoomPan")
envSetVal("layout" "autoZoomPan" 'boolean t)
envSetVal("layout" "autoZoomPan" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

autoZoomScale

```
layout autoZoomScale int integer
```

Description

Lets you specify the zoom scale in percentage when using the dynamic zoom feature. The default is `90%`.

GUI Equivalent

Command	*Options – Display*
Field	*Zoom Scale (%)* (Display Options Form)

Examples

```
envGetVal("layout" "autoZoomScale")
envSetVal("layout" "autoZoomScale" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

balloonFadeTime

`layout balloonFadeTime` *`ingPointNumber`*

`graphic balloonFadeTime float` *`float_number`*

Description

Determines the amount of time (in seconds) for the information balloons to fade away after the balloon appears. Valid values are between .001 and 100.

GUI Equivalent

Command	*Options – Dynamic Display – Info Balloon* tab
Field	*Fade Time* (Display Options Form)

Examples

```
envGetVal("layout" "balloonFadeTime")
envSetVal("layout" "balloonFadeTime" 'float 5.0)
envGetVal("graphic" "balloonFadeTime")
envSetVal("graphic" "balloonFadeTime" 'float 5.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

balloonOn

```
layout balloonOn boolean { t | nil }
```

```
graphic balloonOn boolean { t | nil }
```

Description

Determines whether Info Balloon is set to on (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Dynamic Display*
Field	*Show Info Balloon* (Dynamic Display Form)

Examples

```
envGetVal("layout" "balloonOn")
envSetVal("layout" "balloonOn" 'boolean t)
envSetVal("layout" "balloonOn" 'boolean nil)
envGetVal("graphic" "balloonOn")
envSetVal("graphic" "balloonOn" 'boolean t)
envSetVal("graphic" "balloonOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

balloonPauseTime

`layout balloonPauseTime float` *`float_number`*

Description

Determines the amount of time before the information balloon will appear when the pointer is over an object. Valid values are between 1 and 10.

GUI Equivalent

Command	*Options – Dynamic Display – Info Balloon* tab
Field	*Pause TIme* (Dynamic Display Form)

Examples

```
envGetVal("layout" "balloonPauseTime")
envSetVal("layout" "balloonPauseTime" 'float 1.0)
envSetVal("layout" "balloonPauseTime" 'float 5.0)
envSetVal("layout" "balloonPauseTime" 'float 10.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

balloonTransparency

`layout balloonTransparency int` *`any_integer`*

`graphic balloonTransparency int` *`any_integer`*

Description

Determines the transparency of the balloon information box. When set to 0, the box is completely transparent. When set to 100, objects behind the box are not visible. The default is 50.

GUI Equivalent

Command	*Options – Dynamic Display – Info Balloon* tab
Field	*Transparency* (Dynamic Display Form)

Examples

```
envGetVal("layout" "balloonTransparency")
envSetVal("layout" "balloonTransparency" 'int 70)
envSetVal("layout" "balloonTransparency" 'int 100)
envGetVal("graphic" "balloonTransparency")
envSetVal("graphic" "balloonTransparency" 'int 25)
envSetVal("graphic" "balloonTransparency" 'int 98)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

bannerFields

```
layout bannerFields string { X | Y | Mode | Select | SelI | SelO | SelN | Tpt |
     DRDEnforce | DRDNotify | DRDPost-Edit | CAE | dX | dY | Dist | Cmd }
```

Description

Enables you to configure the items that appear on the status banner. You can display all, some, or none of the items on the status banner. You can also change the order in which the items appear.

- You can set the *Mode* option to full select (F), full select with spine (FS), partial select (P), or partial select with spine (PS).
- The Select option displays a combination of instances and objects.
- SelI displays the number of instances selected. In SelI, the last character is the letter I in uppercase, not one. The number following the setting is the number of instances selected.
- SelO displays the number of objects other than the instances selected. In SelO, the last character is the letter O in uppercase, not zero. The number following the setting is the number of selected objects, other than instances.
- SelN displays the number of selected nets.
- Tpt displays (T) if either the *Transparent Group* icon on the *Options* toolbar or the *Transparent Group* option in the Display Options form is turned on.
- The DRDEnforce, DRDNotify, and DRDPost-Edit options display the status set for DRD modes.
- The option for constraint aware editing, CAE, is available only in VLS XL and VLS GXL. It can be set to ON or OFF. By default, CAE is OFF.

Note: *X*, *Y*, and *Mode* are coupled. If you set one, the others are automatically set. The same is true for *dX*, *dY* and *Dist*, and Select, SelI and SelO.

By default, this environment variable displays Cmd and SnapMode. Cmd is also available on the *Status Toolbar*.

Important

The string selections are case sensitive.

© 2020 All Rights Reserved.

GUI Equivalent

None

Examples

```
envGetVal("layout" "bannerFields")
envSetVal("layout" "bannerFields" 'string "X")
envSetVal("layout" "bannerFields" 'string "Y")
envSetVal("layout" "bannerFields" 'string "Mode")
envSetVal("layout" "bannerFields" 'string "Select")
envSetVal("layout" "bannerFields" 'string "SelI")
envSetVal("layout" "bannerFields" 'string "SelO")
envSetVal("layout" "bannerFields" 'string "SelN")
envSetVal("layout" "bannerFields" 'string "Tpt")
envSetVal("layout" "bannerFields" 'string "DRDEnforce")
envSetVal("layout" "bannerFields" 'string "DRDNotify")
envSetVal("layout" "bannerFields" 'string "DRDPost-Edit")
envSetVal("layout" "bannerFields" 'string "CAE")
envSetVal("layout" "bannerFields" 'string "dX")
envSetVal("layout" "bannerFields" 'string "dY")
envSetVal("layout" "bannerFields" 'string "Dist")
envSetVal("layout" "bannerFields" 'string "Cmd")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageAllowPGNet

```
layout blockageAllowPGNet boolean { t | nil }
```

Description

Enables you to place shapes with signal types power and ground within the blockage area. The default is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Allow PGNet* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageAllowPGNet")
envSetVal("layout" "blockageAllowPGNet" 'boolean t)
envSetVal("layout" "blockageAllowPGNet" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageAttachOwner

```
layout blockageAttachOwner string { none | instance | cluster }
```

Description

Assigns one instance or one cluster as the blockage's owner, depending upon the type of blockage. In the case of a placement halo blockage, the owner should be an instance or P&R boundary. An owner can have any number of blockages. Use the attach/detach command to add or remove an owner. Cyclical relationships are not allowed. For example, if object A is a parent to object B, then object B cannot be a parent for object A. The default is `none`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Attach to Owner* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageAttachOwner")
envSetVal("layout" "blockageAttachOwner" 'string "none")
envSetVal("layout" "blockageAttachOwner" 'string "instance")
envSetVal("layout" "blockageAttachOwner" 'string "cluster")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageAttributeMode

```
layout blockageAttributeMode cyclic { ignoreLPP | useLPP }
```

Description

Determines the selectability of blockages.

- If set to `ignoreLPP`, blockage visibility and selection is controlled by the status of the check box for the corresponding blockage object in the *Object* pane.
- If set to `useLPP`, blockage visibility and selection is controlled by the status of the check box for the corresponding blockage object in the *Object* pane and the status of the object's layer-purpose pair attribute in the *Layer* pane.

The default is `ignoreLPP`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "blockageAttributeMode")
envSetVal("layout" "blockageAttributeMode" 'cyclic "ignoreLPP")
envSetVal("layout" "blockageAttributeMode" 'cyclic "useLPP")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageCreateMethod

```
layout blockageCreateMethod string { manual | automatic }
```

Description

Sets the method used to create a blockage. Manual requires that you enter points to define the blockage. Automatic derives points from objects you select, then creates the blockage from these points. The default is `manual`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Create Method* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageCreateMethod")
envSetVal("layout" "blockageCreateMethod" 'string "manual")
envSetVal("layout" "blockageCreateMethod" 'string "automatic")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageDerivePoints

```
layout blockageDerivePoints string { Selected Set bBox | Instance Halo | Instance
     prBoundary| Instance bBox }
```

Description

Sets the type of object or objects from which you can derive the points to create a blockage. The default is `Selected Set bBox`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Derive From* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageDerivePoints")
envSetVal("layout" "blockageDerivePoints" 'string "Selected Set bBox")
envSetVal("layout" "blockageDerivePoints" 'string "Instance Halo")
envSetVal("layout" "blockageDerivePoints" 'string "Instance prBoundary")
envSetVal("layout" "blockageDerivePoints" 'string "Instance bBox")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageIsPushedDown

```
layout blockageIsPushedDown boolean { t | nil }
```

Description

When set to `t`, indicates that the blockage has been pushed down from a higher level block. The default is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Pushed Down* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageIsPushedDown")
envSetVal("layout" "blockageIsPushedDown" 'boolean t)
envSetVal("layout" "blockageIsPushedDown" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageKeepOwner

```
layout blockageKeepOwner boolean { t | nil }
```

Description

Retains owner information. When you create multiple blockages, the owner is automatically assigned to the additional blockages. The default is `nil`

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Keep Owner* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageKeepOwner")
envSetVal("layout" "blockageKeepOwner" 'boolean t)
envSetVal("layout" "blockageKeepOwner" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageMaterial

```
layout blockageMaterial string { all | other | nwell | pwell | nplus | pplus | ndiff
     | pdiff | poly | metal | diffusion | li | cut | passivationCut}
```

Description

Selects the material on which the blockage is enforced. The default is `all`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Material* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageMaterial")
envSetVal("layout" "blockageMaterial" 'string "all")
envSetVal("layout" "blockageMaterial" 'string "other")
envSetVal("layout" "blockageMaterial" 'string "nwell")
envSetVal("layout" "blockageMaterial" 'string "pwell")
envSetVal("layout" "blockageMaterial" 'string "nplus")
envSetVal("layout" "blockageMaterial" 'string "pplus")
envSetVal("layout" "blockageMaterial" 'string "ndiff")
envSetVal("layout" "blockageMaterial" 'string "pdiff")
envSetVal("layout" "blockageMaterial" 'string "poly")
envSetVal("layout" "blockageMaterial" 'string "metal")
envSetVal("layout" "blockageMaterial" 'string "diffusion")
envSetVal("layout" "blockageMaterial" 'string "li")
envSetVal("layout" "blockageMaterial" 'string "cut")
envSetVal("layout" "blockageMaterial" 'string "passivationCut")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageMaxDensity

`layout blockageMaxDensity` *`any_floating_point_number`*

Description

Sets the maximum percentage of the total design area the metal in the specified layer may occupy. The default is `100.0`

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Max Density* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageMaxDensity")
envSetVal("layout" "blockageMaxDensity" 'float 70.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageSpacingType

```
layout blockageSpacingType string { Blockage geometry | Effective width | Specify }
```

Description

Sets the spacing type for the blockage. The default is `Blockage geometry`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Compute spacing using* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageSpacingType")
envSetVal("layout" "blockageSpacingType" 'string "Blockage geometry")
envSetVal("layout" "blockageSpacingType" 'string "Effective width")
envSetVal("layout" "blockageSpacingType" 'string "Specify")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageSpacingValue

`layout blockageSpacingValue` *`any_floating_point_number`*

Description

Specifies the value for effective width or spacing for blockage. The default is `0.0`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Compute spacing using* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageSpacingValue")
envSetVal("layout" "blockageSpacingValue" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

blockageType

```
layout blockageType string { routing | placement | pin | feedthru | screen }
```

Description

Selects the type of blockage to create. The default is `routing`.

GUI Equivalent

Command	*Create – P&R Objects – Blockage*
Field	*Blockage Type* (Create Blockage Form)

Examples

```
envGetVal("layout" "blockageType")
envSetVal("layout" "blockageType" 'string "routing")
envSetVal("layout" "blockageType" 'string "placement")
envSetVal("layout" "blockageType" 'string "pin")
envSetVal("layout" "blockageType" 'string "feedthru")
envSetVal("layout" "blockageType" 'string "screen")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

bondPadLPP

`layout bondPadLPP string "layer_purpose_pair"`

Description

Specifies the layer purpose pair on which the *Pad Opening Info* command is to search for pad shapes. The default is `"Bondpad drawing"`.

GUI Equivalent

Command	*Tools – Pad Opening Info*
Field	*Layer/Purpose* drop-down in the *Bond Pad* section at the top of the form (Pad Opening Info Form)

Examples

```
envGetVal("layout" "bondPadLPP")
envSetVal("layout" "bondPadLPP" 'string "Metal1 drawing")
envSetVal("layout" "bondPadLPP" 'string "Via2 drawing")
envSetVal("layout" "bondPadLPP" 'string "Bondpad drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

boundarySnappingLayer

`layout boundarySnappingLayer string "layerName1 layerName2"`

Description

Overrides the grid for PR boundary snapping. If this environment variable is specified, the , PR boundary snaps to the active grids whose snapping layers are specified in this environment variable. If this environment variable is not specified, snapping is done based on the grids specified in `boundarySnapXGrids` and `boundarySnapYGrids` environment variables. In case `boundarySnapXGrids` and `boundarySnapYGrids` environment variables are not specified, snapping is done to the least common multiple of all active grids.

GUI Equivalent

None

Examples

```
envGetVal("layout" "boundarySnappingLayer")
envSetVal("layout" "boundarySnappingLayer" 'string "M1WSP")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

boundarySnapXGrids

```
layout boundarySnapXGrids string "grid"
```

Description

Snaps the PR boundary to the specified grid(s) in the X direction.

Note: The grids specified through this environment variable need not be active global grids.

GUI Equivalent

None

Examples

```
envGetVal("layout" "boundarySnapXGrids")
envSetVal("layout" "boundarySnapXGrids" 'string "M1WSP")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

boundarySnapYGrids

```
layout boundarySnapYGrids string "grid"
```

Description

Snaps the PR boundary to the specified grid(s) in the Y direction.

Note: The grids specified through this environment variable need not be active global grids.

GUI Equivalent

None

Examples

```
envGetVal("layout" "boundarySnapYGrids")
envSetVal("layout" "boundarySnapYGrids" 'string "M2WSP M4WSP")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chainMode

```
layout chainMode cyclic { unknown | all | selected | selectLeft | selectRight }
```

Description

Determines which instances in a chain of instances will move, copy, stretch or delete. `All` moves, copies, stretches, or deletes all instances in a chain together. `selected` moves, copies, stretches, or deletes only selected instances irrespective of the chain contents. `selectLeft` moves, copies, stretches, or deletes the instance selected and all instances to the left of the selected instance in the chain. `selectRight` moves, copies, stretches, or deletes the instance selected and all instances to the right of the selected instance in the chain. The default is `selected`.

GUI Equivalent

Command	*Edit – Move*, *Edit – Copy*, *Edit – Stretch*, *Edit – Delete*
Field	*Chain Mode* (Move Form, Copy Form, Stretch Form, Delete Form)

Examples

```
envGetVal("layout" "chainMode")
envSetVal("layout" "chainMode" 'cyclic "unknown")
envSetVal("layout" "chainMode" 'cyclic "all")
envSetVal("layout" "chainMode" 'cyclic "selected")
envSetVal("layout" "chainMode" 'cyclic "selectLeft")
envSetVal("layout" "chainMode" 'cyclic "selectRight")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

checkAllSPGridsEnclosures

```
layout checkAllSPGridsEnclosures boolean {t | nil}
```

Description

During snapping in detailed mode, if this environment variable is set to `t`, Virtuoso searches all enclosures to locate the one that centers the shape on the tracks.

The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "checkAllSPGridsEnclosures")
envSetVal("layout" "checkAllSPGridsEnclosures" 'boolean t)
envSetVal("layout" "checkAllSPGridsEnclosures" 'boolean nil)
```

When the `checkAllSPGridsEnclosures` environment variable is set to `t`, all enclosures values (`0.009`, `0.01`, and `0.012`) are evaluated one by one. When the enviroment variable is set to `nil`, only the first enclosure vlalue (`0.009`) is evaluated.

```
snapPatternDefs(
(GPG94        ("PPitch" "poly94")
     'step                 0.094
     'stepDirection        "horizontal"
    'snappingLayers      (('layer "Poly" 'enclosures (0.009 0.01 0.012) 'purposes
("drawing" "dummy")))
     'type                 "local"
  )
)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

checkHalfWidthPathSeg

```
layout checkHalfWidthPathSeg cyclic { off | notify | enforce | never }
```

Description

Provides control for checking the width of a path segment with respect to the width of the path. This environment variable applies to commands used for creating and editing paths, including ROD MPPs and their subpaths.

When `off`, which is the default value, no warning is issued when you create or edit a path that has a segment length less than half the path width, except if it is the first or last segment. If you create or edit the first or last segment of the path with a length less than half the path width, the endStyle extension is used to determine the segment length. If the environment variable is set to `notify` and you attempt to create or edit a path that has a segment length less than half the path width, a warning dialog box appears. If you confirm to create or edit the short segment as an intended action, the path is created or updated, else the last attempted edit is ignored. If set to `enforce`, you will not be allowed to create a path segment with length less than half the path width.

GUI Equivalent

Command	*Options – Editor*
Field	*Half Width Check* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "checkHalfWidthPathSeg")
envSetVal("layout" "checkHalfWidthPathSeg" 'cyclic "off")
envSetVal("layout" "checkHalfWidthPathSeg" 'cyclic "notify")
envSetVal("layout" "checkHalfWidthPathSeg" 'cyclic "enforce")
envSetVal("layout" "checkHalfWidthPathSeg" 'cyclic "never")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

checkLargeCoordinates

```
layout checkLargeCoordinates boolean {t | nil}
```

Description

Provides control for checking for large coordinates during the copy operation. If set to `t`, the *Copy* command does not allow objects to go out of the INT_MAX boundary and hence does not distort the copied image. On 64-bit system, INT_MAX = 2147483647.

GUI Equivalent

Command	*Options – Editor*
Field	*Check Large Coordinates* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "checkLargeCoordinates")
envSetVal("layout" "checkLargeCoordinates" 'boolean t)
envSetVal("layout" "checkLargeCoordinates" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopArray

```
layout chopArray boolean { t | nil }
```

Description

Enables you to specify whether or not to chop an array of instances (mosaic). If set to `t`, the array of instances is allowed to be chopped based on the shape (line, rectangle, polygon) selected for chopping. If set to `nil`, the array of instances is not chopped. The default value of the variable is `t`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Chop array* (Chop Form)

Examples

```
envGetVal("layout" "chopArray")
envSetVal("layout" "chopArray" 'boolean t)
envSetVal("layout" "chopArray" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopCircleJustify

```
layout chopCircleJustify cyclic { Center | Left | Right | Bottom | Top }
```

Description

Specifies the justification of circle chop shape. The default is `Center`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Chop Shape: circle: Justification* (Chop Form)

Examples

```
envGetVal("layout" "chopCircleJustify")
envSetVal("layout" "chopCircleJustify" 'cyclic "Center")
envSetVal("layout" "chopCircleJustify" 'cyclic "Left")
envSetVal("layout" "chopCircleJustify" 'cyclic "Right")
envSetVal("layout" "chopCircleJustify" 'cyclic "Bottom")
envSetVal("layout" "chopCircleJustify" 'cyclic "Top")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopCircleRadius

`layout chopCircleRadius float` *`float_number`*

Description

Specifies the radius of the circle chop shape. The layout default is `0.0` user units.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Radius* (Chop Form)

Examples

```
envGetVal("layout" "chopCircleRadius")
envSetVal("layout" "chopCircleRadius" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopLineJustify

```
layout chopLineJustify cyclic { Center | Left | Right }
```

Description

Specifies the justification of line chop shape. The default is `Center`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Chop Shape: line: Justification* (Chop Form)

Examples

```
envGetVal("layout" "chopLineJustify")
envSetVal("layout" "chopLineJustify" 'cyclic "Center")
envSetVal("layout" "chopLineJustify" 'cyclic "Left")
envSetVal("layout" "chopLineJustify" 'cyclic "Right")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopLineType

```
layout chopLineType cyclic { User Spacing | Min Spacing }
```

Description

Specifies the type of spacing for line chop shape, `User Spacing` or `Min Spacing`. The default value is `User Spacing`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Type* (Chop Form)

Examples

```
envGetVal("layout" "chopLineType")
envSetVal("layout" "chopLineType" 'cyclic "User Spacing")
envSetVal("layout" "chopLineType" 'cyclic "Min Spacing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopLineUserWidth

`layout chopLineUserWidth float` *`float_number`*

Description

Specifies the width of the user spacing for line chop shape. The layout default is `0.0` user units.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Type: User Spacing: Width* (Chop Form)

Examples

```
envGetVal("layout" "chopLineUserWidth")
envSetVal("layout" "chopLineUserWidth" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopPRBDialog

```
layout chopPRBDialog boolean { t | nil }
```

Description

Displays a warning dialog box when the `Chop` command is executed outside the PR boundary or there is no PR boundary in the design. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "chopPRBDialog")
envSetVal("layout" "chopPRBDialog" 'boolean t)
envSetVal("layout" "chopPRBDialog" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopRectHeight

`layout chopRectHeight float` *`float_number`*

Description

Specifies the height of the rectangle chop shape. The layout default is `0.0` user units.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Height* (Chop Form)

Examples

```
envGetVal("layout" "chopRectHeight")
envSetVal("layout" "chopRectHeight" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopRectJustify

```
layout chopRectJustify cyclic { Bottom left | Bottom right | Top left | Top right
     | Center }
```

Description

Specifies the justification of rectangle chop shape. The default is `Bottom Left`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Chop Shape: rectangle: Justification* (Chop Form)

Examples

```
envGetVal("layout" "chopRectJustify")
envSetVal("layout" "chopRectJustify" 'cyclic "Bottom left")
envSetVal("layout" "chopRectJustify" 'cyclic "Bottom right")
envSetVal("layout" "chopRectJustify" 'cyclic "Top left")
envSetVal("layout" "chopRectJustify" 'cyclic "Top right")
envSetVal("layout" "chopRectJustify" 'cyclic "Center")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopRectWidth

`layout chopRectWidth float` *`float_number`*

Description

Specifies the width of the rectangle chop shape. The layout default is `0.0` user units.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Width* (Chop Form)

Examples

```
envGetVal("layout" "chopRectWidth")
envSetVal("layout" "chopRectWidth" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopShape

```
layout chopShape cyclic { unknown | rectangle | polygon | line | circle | wire }
```

Description

Specifies the default shape of the cutter used by `leHiChop` or `leChopShape`. The default is `rectangle`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Chop Shape* (Chop Form)

Examples

```
envGetVal("layout" "chopShape")
envSetVal("layout" "chopShape" 'cyclic "rectangle")
envSetVal("layout" "chopShape" 'cyclic "polygon")
envSetVal("layout" "chopShape" 'cyclic "line")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopSpecifySize

```
layout chopSpecifySize boolean { t | nil }
```

Description

Lets you specify the size of the chop shape, rectangle or circle. The default is `nil`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Specify Size* (Chop Form)

Examples

```
envGetVal("layout" "chopSpecifySize")
envSetVal("layout" "chopSpecifySize" 'boolean t)
envSetVal("layout" "chopSpecifySize" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

chopSpecifySpacing

```
layout chopSpecifySpacing boolean { t | nil }
```

Description

Lets you specify the spacing of the line chop shape. The default is `nil`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Specify Spacing* (Chop Form)

Examples

```
envGetVal("layout" "chopSpecifySpacing")
envSetVal("layout" "chopSpecifySpacing" 'boolean t)
envSetVal("layout" "chopSpecifySpacing" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

circleRadius

`layout circleRadius float` *`float_number`*

Description

Specifies the radius of the circle. The default value is `0.0`.

GUI Equivalent

Command	*Create – Shape – Circle*
Field	*Radius* (Create Circle Form)

Examples

```
envGetVal("layout" "circleRadius")
envSetVal("layout" "circleRadius" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

circleShapeMode

```
layout circleShapeMode cyclic { normal | halfQ1 | halfQ2 | halfQ3 | halfQ4 |
     quarterQ1 | quarterQ2 | quarterQ3 |quarterQ4}
```

Description

Specifies the shape of circle that you want to create: full circle (`normal`), half circle (`halfQ1, halfQ2, halfQ3, halfQ4`), or quarter circle (`quarterQ1, quarterQ2, quarterQ3, quarterQ4`). The default value is `normal`.

GUI Equivalent

Command	*Create – Shape – Circle*
Field	*Shape Type* (Create Circle Form)

Examples

```
envGetVal("layout" "circleShapeMode")
envSetVal("layout" "circleShapeMode" 'cyclic "normal")
envSetVal("layout" "circleShapeMode" 'cyclic "halfQ1")
envSetVal("layout" "circleShapeMode" 'cyclic "halfQ2")
envSetVal("layout" "circleShapeMode" 'cyclic "halfQ3")
envSetVal("layout" "circleShapeMode" 'cyclic "halfQ4")
envSetVal("layout" "circleShapeMode" 'cyclic "quarterQ1")
envSetVal("layout" "circleShapeMode" 'cyclic "quarterQ2")
envSetVal("layout" "circleShapeMode" 'cyclic "quarterQ3")
envSetVal("layout" "circleShapeMode" 'cyclic "quarterQ4")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

clusterBoundaryAspectRatio

`layout clusterBoundaryAspectRatio` *`floating_point_number`*

Description

In Virtuoso XL, sets the aspect ratio of the cluster boundary during auto-creation. Aspect Ratio W/L is the width-to-length ratio of the cluster boundary. For example, a value of 1 specifies a square boundary, 2 specifies a boundary twice as high as it is long, and .5 specifies a boundary half as wide as it is long.The default is `1.0`.

GUI Equivalent

Command	*Create – P&R Objects – Cluster Boundary*
Field	*Aspect Ratio (W/H)* (Create Cluster Boundary Form)

Examples

```
envGetVal("layout" "clusterBoundaryAspectRatio")
envSetVal("layout" "clusterBoundaryAspectRatio" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

clusterBoundaryUtilization

`layout clusterBoundaryUtilization` *`floating_point_number`*

Description

In Virtuoso XL, specifies the percentage of area within the cluster boundary that you want to fill. The default is `80.0`.

GUI Equivalent

Command	*Create – P&R Objects – Cluster Boundary*
Field	*Utilization %* (Create Cluster Boundary Form)

Examples

```
envGetVal("layout" "clusterBoundaryUtilization")
envSetVal("layout" "clusterBoundaryUtilization" 'float 90.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

confirmDelete

```
layout confirmDelete boolean { t | nil }
```

Description

Turns on and off a dialog box which reads “Delete Figure?” to confirm an object deletion. You choose Yes or No on the form. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "confirmDelete")
envSetVal("layout" "confirmDelete" 'boolean t)
envSetVal("layout" "confirmDelete" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

contactColumns

`layout contactColumns int` *`any_integer_greater_than_zero`*

Description

Specifies the number of columns for a contact placement. The default is `1`.

GUI Equivalent

Command	*Create – Contact*
Field	*Columns*

Examples

```
envGetVal("layout" "contactColumns")
envSetVal("layout" "contactColumns" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

convertEachInstanceIntoMosaic

```
layout convertEachInstanceIntoMosaic 'boolean { t | nil }
```

Description

Specifies the mode for the *Convert to Mosaic* command.

- When set to `t`, the command converts the currently selected instances into the equivalent number of individual mosaics.
- When set to `nil`, the command converts the currently selected instances into a single mosaic.

GUI Equivalent

Command	*Edit – Convert – To Mosaic*
Fields	*Convert Instances into Mosaics* (`t`)
	Convert Instances into Single Mosaic (`nil`)

Examples

```
envGetVal("layout" "convertEachInstanceIntoMosaic")
envSetVal("layout" "convertEachInstanceIntoMosaic" 'boolean t)
envSetVal("layout" "convertEachInstanceIntoMosaic" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyColumns

`layout copyColumns int` *`any_integer_greater_than_zero`*

Description

Specifies the default number of columns to copy.

GUI Equivalent

Command	*Edit – Copy*
Field	*Columns* (Copy Form)

Examples

```
envGetVal("layout" "copyColumns")
envSetVal("layout" "copyColumns" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyConnectivity

```
layout copyConnectivity boolean { t | nil }
```

Description

When selected, the connectivity information on path, pathseg, fluid guard ring, and via is copied to the new object. The default is `nil`.

GUI Equivalent

Command	*Edit – Copy*
Field	*Copy Connectivity* (Copy Form)

Examples

```
envGetVal("layout" "copyConnectivity")
envSetVal("layout" "copyConnectivity" 'boolean t)
envSetVal("layout" "copyConnectivity" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyIncrementInstName

```
layout copyIncrementInstName boolean { t | nil }
```

Description

When selected, increments the instance name during the `Copy` command. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "copyIncrementInstName")
envSetVal("layout" "copyIncrementInstName" 'boolean t)
envSetVal("layout" "copyIncrementInstName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyIncrementNetName

```
layout copyIncrementNetName boolean { t | nil }
```

Description

When selected, increments the bus bit on the copied object by one if the selected object has a net name with a bus bit. The default is `nil`.

GUI Equivalent

Command	*Edit – Copy*
Field	*Increment net name* (Copy Form)

Examples

```
envGetVal("layout" "copyIncrementNetName")
envSetVal("layout" "copyIncrementNetName" 'boolean t)
envSetVal("layout" "copyIncrementNetName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyResetRowCol

```
layout copyResetRowCol boolean { t | nil }
```

Description

Sets the default location for the Copy form's *Rows* and *Columns* fields each time you repeat the Copy command. When `t`, the location is set to the `.cdsenv` file. When `nil`, the last value entered in the form fields is retained.

GUI Equivalent

None

Examples

```
envGetVal("layout" "copyResetRowCol")
envSetVal("layout" "copyResetRowCol" 'boolean t)
envSetVal("layout" "copyResetRowCol" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyRows

`layout copyRows int` *`any_integer_greater_than_zero`*

Description

Specifies the default number of rows to copy.

GUI Equivalent

Command	*Edit – Copy*
Field	*Rows* (Copy Form)

Examples

```
envGetVal("layout" "copyRows")
envSetVal("layout" "copyRows" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

copyToLayer

```
layout copyToLayer boolean { t | nil }
```

Description

Specifies whether shapes are copied to a different layer with `leHiCopy`. This does not change the form field in the form. The default is `nil`.

GUI Equivalent

Command	*Edit - Copy*
Field	*Change To Layer* (Copy Form)

Examples

```
envGetVal("layout" "copyToLayer")
envSetVal("layout" "copyToLayer" 'boolean t)
envSetVal("layout" "copyToLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

createAsLabel

```
layout createAsLabel boolean { t | nil }
```

Description

Specifies that a label is created as a label (`t`) or a text display (`nil`). The default is `nil`.

GUI Equivalent

Command	*Create – Label*
Field	*Mode: Auto: Create As Label* (Create Label Form)

Examples

```
envGetVal("layout" "createAsLabel")
envSetVal("layout" "createAsLabel" 'boolean t)
envSetVal("layout" "createAsLabel" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

createInstResetCDF

```
layout createInstResetCDF boolean { t | nil }
```

Description

Specifies whether the Create Instance command retains the current effective CDF values for a component when the same component is placed at a later time. If set to `t`, the editor CDF resets the CDF values to their default settings when you start the Create Instance command. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "createInstResetCDF")
envSetVal("layout" "createInstResetCDF" 'boolean t)
envSetVal("layout" "createInstResetCDF" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

createPinLabel

```
layout createPinLabel boolean { t | nil }
```

Description

Specifies that a pin label be created when using the Create Pin form (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Create – Pin*
Field	*Create Label (*Create Pin Form*)*

Examples

```
envGetVal("layout" "createPinLabel")
envSetVal("layout" "createPinLabel" 'boolean t)
envSetVal("layout" "createPinLabel" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

createPinPathSpineAware

```
layout createPinPathSpineAware boolean { t | nil }
```

Description

Determines whether pins created in the *Single* mode are path spine aware.

- When set to `t`, a single merged pin is created on the highest metal layer shape island. A metal layer shape island is made of overlapping shapes that belong to the same layer and net.
- When set to `nil`, a single pin is created on the highest metal layer shape island.

GUI Equivalent

None

Examples

```
envGetVal("layout" "createPinPathSpineAware")
envSetVal("layout" "createPinPathSpineAware" 'boolean t)
envSetVal("layout" "createPinPathSpineAware" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

Creating Pins Automatically

© 2020 All Rights Reserved.

createPinSchematicAware

```
layout createPinSchematicAware boolean { t | nil }
```

Description

Specifies whether pin creation must be schematic aware. When set to `t`, the tool uses its *I/O Type* and *Signal Type* values from corresponding schematic reference. When set to `nil` or if the corresponding schematic reference is not found, the values specified in the Create Pin form are used.

GUI Equivalent

Command	*Create – Pin*
Field	*Schematic Aware (*Create Pin Form*)*

```
envGetVal("layout" "createPinSchematicAware")
envSetVal("layout" "createPinSchematicAware" 'boolean t)
envSetVal("layout" "createPinSchematicAware" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

createRectilinearBoundary

```
layout createRectilinearBoundary boolean { t | nil }
```

Description

Specifies that a rectilinear PR boundary is created by merging the PR boundary of the overlapping instances in the selection set. In case, there is no overlap between the selected instances, a rectangular PR boundary is created instead of rectilinear PR boundary. If this environment variable is set to `nil`, a rectangular PR boundary is created from the bounding box of the selection set.The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "createRectilinearBoundary")
envSetVal("layout" "createRectilinearBoundary" 'boolean t)
envSetVal("layout" "createRectilinearBoundary" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

createViaAlwaysUsesDefinedCalcMode

```
layout createViaAlwaysUsesDefinedCalcMode boolean { t | nil}
```

Description

Enables the via engine to use the calculation mode defined in the *Layout Editor Options* form for creating vias. If the environment variable is set to `nil`, the via engine uses pre-defined values for creating vias in *Single*, *Stack*, or *Auto* mode. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "createViaAlwaysUsesDefinedCalcMode")
envSetVal("layout" "createViaAlwaysUsesDefinedCalcMode" 'boolean t)
envSetVal("layout" "createViaAlwaysUsesDefinedCalcMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

cursorShape

```
layout cursorShape boolean { t | nil }
graphic cursorShape boolean { t | nil }
```

Description

Determines whether the pointer is displayed as dot shaped (`nil`) or as dotted-line cross shaped (`t`). The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Cross Cursor* (Display Options Form)

Examples

```
envGetVal("layout" "cursorShape")
envSetVal("layout" "cursorShape" 'boolean t)
envSetVal("layout" "cursorShape" 'boolean nil)
envGetVal("graphic" "cursorShape")
envSetVal("graphic" "cursorShape" 'boolean t)
envSetVal("graphic" "cursorShape" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultNewCellName

`graphic defaultNewCellName string` *`any_cell_name`*

Description

Specifies the default name of the cell for the *Cell* field in the New File form, accessed by choosing *File – New*. The default is none.

GUI Equivalent

Command	*File – New*
Field	*Cell*

Examples

```
envGetVal("graphic" "defaultNewCellName")
envSetVal("graphic" "defaultNewCellName" 'string "MyCell")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultNewLibName

`graphic defaultNewLibName string` *`any_library_name`*

Description

Specifies the default name of the library for the *Library* field in the New File form, accessed by choosing *File – New*. The default is none.

GUI Equivalent

Command	*File – New*
Field	*Library*

Examples

```
envGetVal("graphic" "defaultNewLibName")
envSetVal("graphic" "defaultNewLibName" 'string "MyLib")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultNewViewName

`graphic defaultNewViewName string` *`any_valid_view_name`*

Description

Specifies the default name of the view for the *View* field in the New File form, accessed by choosing *File – New*. The default is none.

GUI Equivalent

Command	*File – New*
Field	*View*

Examples

```
envGetVal("graphic" "defaultNewViewName")
envSetVal("graphic" "defaultNewViewName" 'string "schematic")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultOpenCellName

`graphic defaultOpenCellName string` *`any_cell_name`*

Description

Specifies the default name of the cell for the *Cell* field in the *Open File* form, accessed by choosing *File – Open*. The default is none.

GUI Equivalent

Command	*File – Open*
Field	*Cell*

Examples

```
envGetVal("graphic" "defaultOpenCellName")
envSetVal("graphic" "defaultOpenCellName" 'string "MyCell")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultOpenLibName

`graphic defaultOpenLibName string` *`any_library_name`*

Description

Specifies the default name of the library for the *Library* field in the *Open File* form, accessed by choosing *File – Open*. The default is none.

GUI Equivalent

Command	*File – Open*
Field	*Library*

Examples

```
envGetVal("graphic" "defaultOpenLibName")
envSetVal("graphic" "defaultOpenLibName" 'string "MyLib")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultOpenViewName

`graphic defaultOpenViewName string` *`any_valid_view_name`*

Description

Specifies the default name of the view for the *View* field in the *Open File* form, accessed by choosing *File – Open*. The default is none.

GUI Equivalent

Command	*File – Open*
Field	*View*

Examples

```
envGetVal("graphic" "defaultOpenViewName")
envSetVal("graphic" "defaultOpenViewName" 'string "layout")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defaultToolName

`graphic defaultToolName string` *`any_valid_tool_name`*

Description

Specifies the default tool name in the *Application – Open with* field in the New File form, accessed by choosing *File – New*. The default is none.

GUI Equivalent

Command	*File – New*
Field	*Application – Open with*

Examples

```
envGetVal("graphic" "defaultToolName")
envSetVal("graphic" "defaultToolName" 'string "Layout XL")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

defCreatePinConnModel

```
layout defCreatePinConnModel cyclic { strong | weak }
```

Description

Determines the default connectivity model while creating pins. The default is `strong`.

GUI Equivalent

Command	*Create – Pin*
Field	*Connectivity* (Create Pin Form)

Examples

```
envGetVal("layout" "defCreatePinConnModel")
envSetVal("layout" "defCreatePinConnModel" 'cyclic "strong")
envSetVal("layout" "defCreatePinConnModel" 'cyclic "weak")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

delDetachedBlockages

```
layout delDetachedBlockages boolean { t | nil }
```

Description

Deletes blockages which would otherwise remain after an instance containing blockages is flattened. The default is `nil`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Detached blockages* (Flatten Form)

Examples

```
envGetVal("layout" "delDetachedBlockages")
envSetVal("layout" "delDetachedBlockages" 'boolean t)
envSetVal("layout" "delDetachedBlockages" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

deleteFluidShapes

```
layout deleteFluidShapes boolean { t | nil }
```

Description

Deletes the fluid shapes when flattening Pcell instances. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "deleteFluidShapes")
envSetVal("layout" "deleteFluidShapes" 'boolean t)
envSetVal("layout" "deleteFluidShapes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

deleteTarget

```
layout deleteTarget cyclic { unknown | selected | net | pathSegment }
```

Description

Determines which selected objects are deleted when you use the *Edit – Delete* command. The default is `selected`.

GUI Equivalent

Command	*Edit – Delete*
Field	*Net Interconnect, Selected, Segment of a Path* (Delete Form)

Examples

```
envGetVal("layout" "deleteTarget")
envSetVal("layout" "deleteTarget" 'cyclic "unknown")
envSetVal("layout" "deleteTarget" 'cyclic "selected")
envSetVal("layout" "deleteTarget" 'cyclic "net")
envSetVal("layout" "deleteTarget" 'cyclic "pathSegment")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

descendPromptForView

```
layout descendPromptForView boolean { t | nil }
```

Description

Determines whether to open the Descend (view name) form when you click an instance. In the Descend (view name) form, you can choose a different view to open. If there is only one view of the master cell, Descend does not prompt you for a view name. If set to `nil`, the default view is the view name of the selected object. The default is `nil`.

GUI Equivalent

Command	*Edit – Hierarchy – Descend Read*
Field	*Prompt For View Name* (Descend Options Form)

Examples

```
envGetVal("layout" "descendPromptForView")
envSetVal("layout" "descendPromptForView" 'boolean t)
envSetVal("layout" "descendPromptForView" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dimGridLines

```
layout dimGridLines boolean { t | nil }
```

```
graphic dimGridLines boolean { t | nil }
```

Description

Determines whether or not the grid lines appear dimmed on the canvas.

GUI Equivalent

Command	*Options – Display*
Field	*Dim Grid Lines* (Display Options Form)

Examples

```
envGetVal("layout" "dimGridLines")
envSetVal("layout" "dimGridLines" 'boolean t)
envSetVal("layout" "dimGridLines" 'boolean nil)
envGetVal("graphic" "dimGridLines")
envSetVal("graphic" "dimGridLines" 'boolean t)
envSetVal("graphic" "dimGridLines" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dimmingAutoEnabled

```
layout dimmingAutoEnabled boolean { t | nil }
graphic dimmingAutoEnabled boolean { t | nil }
```

Description

Determines whether or not to automatically dim the design when you select an object or when the design contains highlight sets or probes.

GUI Equivalent

Command	*Options – Display*
Field	*Automatic Dimming* (Display Options Form)

Examples

```
envGetVal("layout" "dimmingAutoEnabled")
envSetVal("layout" "dimmingAutoEnabled" 'boolean t)
envSetVal("layout" "dimmingAutoEnabled" 'boolean nil)
envGetVal("graphic" "dimmingAutoEnabled")
envSetVal("graphic" "dimmingAutoEnabled" 'boolean t)
envSetVal("graphic" "dimmingAutoEnabled" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dimmingIntensity

`layout dimmingIntensity int` *`integer_number`*

`graphic dimmingIntensity int` *`integer_number`*

Description

Determines the luminosity of the dimmed objects. When set to 0 nothing is dimmed. When set to 100 dimmed objects are not visible. The default is 50.

GUI Equivalent

Command	*Options – Display*
Field	*Dim Intensity* (Display Options Form)

Examples

```
envGetVal("layout" "dimmingIntensity")
envSetVal("layout" "dimmingIntensity" 'int 70)
envGetVal("graphic" "dimmingIntensity")
envSetVal("graphic" "dimmingIntensity" 'int 70)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dimmingOn

```
layout dimmingOn boolean { t | nil }
graphic dimmingOn boolean { t | nil }
```

Description

Specifies whether you can change the luminosity of the objects in the cellview. The default value is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Enable Dimming* (Display Options Form)

Examples

```
envGetVal("layout" "dimmingOn")
envSetVal("layout" "dimmingOn" 'boolean t)
envSetVal("layout" "dimmingOn" 'boolean nil)
envGetVal("graphic" "dimmingOn")
envSetVal("graphic" "dimmingOn" 'boolean t)
envSetVal("graphic" "dimmingOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dimmingScope

```
layout dimmingScope string { none | all | outside | eipSurround }
graphic dimmingScope string { none | all | outside | eipSurround }
```

Description

Determines how objects in the cellview are dimmed.

- none: no dimming is applied.
- outside: dims everything but the instance, figure group, or any object you are editing in place.
- all: the design is completely dimmed as soon as you select this option. If you select an object and the dimming is set to 100, only the selection box is displayed.
- eipSurround: dims all the objects surrounding the EIP hierarchy.

GUI Equivalent

Command	*Options – Display*
Field	*Scope* (Display Options Form)

Examples

```
envGetVal("layout" "dimmingScope")
envSetVal("layout" "dimmingScope" 'string "none")
envSetVal("layout" "dimmingScope" 'string "all")
envSetVal("layout" "dimmingScope" 'string "outside")
envSetVal("layout" "dimmingScope" 'string "eipSurround")
envGetVal("graphic" "dimmingScope")
envSetVal("graphic" "dimmingScope" 'string "none")
envSetVal("graphic" "dimmingScope" 'string "all")
envSetVal("graphic" "dimmingScope" 'string "outside")
envSetVal("graphic" "dimmingScope" 'string "eipSurround")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

disableLayerFuncSnapToMfgGrid

```
layout disableLayerFuncSnapToMfggrid boolean { t | nil }
```

Description

By default, the *leLayer**, `leMergeShapes` and `leSizeShape` functions do not put data on grid. These functions are effected by the environment variable `disableLayerFuncSnapToMfgGrid`. To force the *le* functions (and the corresponding functions from the menus) to put data results on grid, set the `disableLayerFuncSnapToMfgGrid` function to `nil`.

Note: The *dbLayer** functions do not put date results back onto the manufacturing grid and are unaffected by the `disableLayerFuncSnapToMfgGrid` environment variable.

GUI Equivalent

None

Examples

```
envGetVal("layout" "disableLayerFuncSnapToMfggrid")
envSetVal("layout" "disableLayerFuncSnapToMfggrid" 'boolean t)
envSetVal("layout" "disableLayerFuncSnapToMfggrid" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

disableMissingLayerPopUp

```
layout disableMissingLayerPopUp boolean { t | nil }
```

Description

Determines whether to disable the missing layer pop-up window (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "disableMissingLayerPopUp")
envSetVal("layout" "disableMissingLayerPopUp" 'boolean t)
envSetVal("layout" "disableMissingLayerPopUp" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

disableRoundPathStyle

```
layout disableRoundPathStyle boolean { t | nil }
```

Description

Determines whether to disable round path style (`t`) option from the *End Type* field of the Create Path form or not (`nil`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "disableRoundPathStyle")
envSetVal("layout" "disableRoundPathStyle" 'boolean t)
envSetVal("layout" "disableRoundPathStyle" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

discardCVEditOnWindowClose

```
graphic discardCVEditOnWindowClose boolean { t | nil }
```

Description

Controls whether the changes done in the cellview are saved when you click *No* in the *Save Changes* dialog box. The default is `t`. When set to `nil`, the changes done in the cellview are preserved in the memory when you click *No* in the *Save Changes* dialog box.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "discardCVEditOnWindowClose")
envSetVal("graphic" "discardCVEditOnWindowClose" 'boolean t)
envSetVal("graphic" "discardCVEditOnWindowClose" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayFigGroupName

```
layout displayFigGroupName boolean { t | nil }
```

Description

Determines whether to display only the group name for a figure group (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Group Name Only* (Display Options Form)

Examples

```
envGetVal("layout" "displayFigGroupName")
envSetVal("layout" "displayFigGroupName" 'boolean t)
envSetVal("layout" "displayFigGroupName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayFigGroupShapes

```
graphic displayFigGroupShapes boolean { t | nil }
```

Description

Determines whether to display the group shapes (`t`) or only the group name and the bounding box for a figure group (`nil`). The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "displayFigGroupShapes")
envSetVal("layout" "displayFigGroupShapes" 'boolean t)
envSetVal("layout" "displayFigGroupShapes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayInstanceOrientation

```
layout displayInstanceOrientation boolean { t | nil }
graphic displayInstanceOrientation boolean { t | nil }
```

Description

Determines whether to display instance orientation in the cellview (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Instance Orientation* (Display Options Form)

Examples

```
envGetVal("layout" "displayInstanceOrientation")
envSetVal("layout" "displayInstanceOrientation" 'boolean t)
envSetVal("layout" "displayInstanceOrientation" 'boolean nil)
envGetVal("graphic" "displayInstanceOrientation")
envSetVal("graphic" "displayInstanceOrientation" 'boolean t)
envSetVal("graphic" "displayInstanceOrientation" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayMissingMasterMarkers

```
graphic displayMissingMasterMarkers boolean { t | nil }
```

Description

Enables (`t`) or disables (`nil`) the display of missing master markers.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "displayMissingMasterMarkers")
envSetVal("graphic" "displayMissingMasterMarkers" 'boolean t)
envSetVal("graphic" "displayMissingMasterMarkers" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayOptionsFile

```
layout displayOptionsFile string "~/.cdsenv"
```

Description

Specifies the file to which you want to store or from which you want to load the settings.

GUI Equivalent

Command	*Options – Display*
Field	*File* (Display Options Form)

Examples

```
envGetVal("layout" "displayOptionsFile")
```

```
envSetVal("layout" "displayOptionsFile" 'string "~/.cdsenv")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayPCellOrientation

```
graphic displayPCellOrientation boolean { t | nil }
```

Description

Displays the orientation for Pcells. The environment variable, `displayInstanceOrientation`, should be enabled for the `displayPCellOrientation` environment variable to work. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "displayPCellOrientation ")
envSetVal("graphic" "displayPCellOrientation " 'boolean t)
envSetVal("graphic" "displayPCellOrientation " 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayPinNames

```
layout displayPinNames boolean { t | nil }
graphic displayPinNames boolean { t | nil }
```

Description

Determines whether to display pin name text displays or labels in the cellview (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Pin Names* (Display Options Form)

Examples

```
envGetVal("layout" "displayPinNames")
envSetVal("layout" "displayPinNames" 'boolean t)
envSetVal("layout" "displayPinNames" 'boolean nil)
envGetVal("graphic" "displayPinNames")
envSetVal("graphic" "displayPinNames" 'boolean t)
envSetVal("graphic" "displayPinNames" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayPRHighestResolution

```
layout displayPRHighestResolution boolean { t | nil }
graphic displayPRHighestResolution boolean { t | nil }
```

Description

Determines whether to use the filter size for objects that is bigger or equal to the filter size used for shapes.

GUI Equivalent

Command	*Options – Display*
Field	*Highest Resolution for P&R Objects* (Display Options Form)

Examples

```
envGetVal("layout" "displayPRHighestResolution")
envSetVal("layout" "displayPRHighestResolution" 'boolean t)
envSetVal("layout" "displayPRHighestResolution" 'boolean nil)
envGetVal("graphic" "displayPRHighestResolution")
envSetVal("graphic" "displayPRHighestResolution" 'boolean t)
envSetVal("graphic" "displayPRHighestResolution" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayRefPoint

```
layout displayRefPoint boolean { t | nil }
```

Description

Specifies if the reference point is displayed in the cellview (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Editor*
Field	*Display Reference Point* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "displayRefPoint")
envSetVal("layout" "displayRefPoint" 'boolean t)
envSetVal("layout" "displayRefPoint" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayResolution

```
graphic displayResolution string { Low | Medium | High | Very High }
```

```
layout displayResolution string { Low | Medium | High | Very High }
```

Description

Enables you to set the display resolution to `Low`, `Medium`, `High`, or `Very High`.

GUI Equivalent

Command	*Options – Display*
Field	*Resolution* (Display Options Form)

Examples

```
envGetVal("graphic" "displayResolution")
envSetVal{"graphic" "displayResolution" 'string "Low")
envSetVal{"graphic" "displayResolution" 'string "Medium")
envSetVal{"graphic" "displayResolution" 'string "High")
envSetVal{"graphic" "displayResolution" 'string "Very High")
envGetVal("layout" "displayResolution")
envSetVal{"layout" "displayResolution" 'string "Low")
envSetVal{"layout" "displayResolution" 'string "Medium")
envSetVal{"layout" "displayResolution" 'string "High")
envSetVal{"layout" "displayResolution" 'string "Very High")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayRowName

```
layout displayRowName boolean { t | nil }
graphic displayRowName boolean { t | nil }
```

Description

Displays the names of placed rows. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Row Name* (Display Options Form)

Examples

```
envGetVal("layout" "displayRowName")
envSetVal("layout" "displayRowName" 'boolean t)
envSetVal("layout" "displayRowName" 'boolean nil)
envGetVal("graphic" "displayRowName")
envSetVal("graphic" "displayRowName" 'boolean t)
envSetVal("graphic" "displayRowName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayRowSite

```
layout displayRowSite boolean { t | nil }
graphic displayRowSite boolean { t | nil }
```

Description

Displays the position of row sites, which are represented by a cross marker on the row. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Display Row Site* (Display Options Form)

Examples

```
envGetVal("layout" "displayRowSite")
envSetVal("layout" "displayRowSite" 'boolean t)
envSetVal("layout" "displayRowSite" 'boolean nil)
envGetVal("graphic" "displayRowSite")
envSetVal("graphic" "displayRowSite" 'boolean t)
envSetVal("graphic" "displayRowSite" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayRulerMiddlePoint

```
graphic displayRulerMiddlePoint boolean { t | nil }
```

Description

Displays a tick in the middle point of the ruler segment. You can extend the tick from both sides of the ruler.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "displayRulerMiddlePoint")
envSetVal("graphic" "displayRulerMiddlePoint" 'boolean t)
envSetVal("graphic" "displayRulerMiddlePoint" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displaySchDashText

```
graphic displaySchDashText boolean { t | nil }
```

Description

Specifies how to display text or labels. When set to `t`, text or labels are displayed as small lines when the schematic or symbol is zoomed out. When set to `nil`, the text or labels always display, even when the window is zoomed out. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "displaySchDashText")
envSetVal("graphic" "displaySchDashText" 'boolean t)
envSetVal("graphic" "displaySchDashText" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displaySelectionHighlightWhenDimmed

```
layout displaySelectionHighlightWhenDimmed boolean { t | nil }
```

```
graphic displaySelectionHighlightWhenDimmed boolean { t | nil }
```

Description

Specifies the display of selected objects in their original colors when the design or other objects in the design are set to be dimmed. The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*True Color Selection only* (Display Options Form)

Examples

```
envGetVal("layout" "displaySelectionHighlightWhenDimmed")
envSetVal("layout" "displaySelectionHighlightWhenDimmed" 'boolean t)
envSetVal("layout" "displaySelectionHighlightWhenDimmed" 'boolean nil)
envGetVal("graphic" "displaySelectionHighlightWhenDimmed")
envSetVal("graphic" "displaySelectionHighlightWhenDimmed" 'boolean t)
envSetVal("graphic" "displaySelectionHighlightWhenDimmed" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displaySigDigits

`layout displaySigDigits int` *`any_integer_between_0_and_10`*

Description

Controls the number of decimal digits that display during the dynamic display of measurements of shapes. The default is `4`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "displaySigDigits")
envSetVal("layout" "displaySigDigits" 'int 7)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayStartLevel

`layout displayStartLevel int` *`any_integer_between_0_and_32`*

Description

Specifies the start level for viewing the hierarchy in the Set Area View Level form. The default is `0`.

GUI Equivalent

Command	*View – Area Display – Set*
Field	*Display Levels – From* (Set Area View Level Form)

Examples

```
envGetVal("layout" "displayStartLevel")
envSetVal("layout" "displayStartLevel" 'int 15)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayStopLevel

`layout displayStopLevel int` *`any_integer_between_0_and_32`*

Description

Specifies the stop level for viewing the hierarchy in the Set Area View Level form. The default is `0`.

GUI Equivalent

Command	*View – Area Display – Set*
Field	*Display Levels – To* (Set Area View Level Form)

Examples

```
envGetVal("layout" "displayStopLevel")
envSetVal("layout" "displayStopLevel" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayStretchHandles

```
layout displayStretchHandles boolean { t | nil }
```

```
graphic displayStretchHandles boolean { t | nil }
```

Description

Specifies whether stretch handles are displayed for stretchable Pcells (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Stretch Handles* (Display Options Form)

Examples

```
envGetVal("layout" "displayStretchHandles")
envSetVal("layout" "displayStretchHandles" 'boolean t)
envSetVal("layout" "displayStretchHandles" 'boolean nil)
envGetVal("graphic" "displayStretchHandles")
envSetVal("graphic" "displayStretchHandles" 'boolean t)
envSetVal("graphic" "displayStretchHandles" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

displayViaShapes

```
layout displayViaShapes boolean { t | nil }
```

```
graphic displayViaShapes boolean { t | nil }
```

Description

Displays via attributes. When Via Shapes is set to `nil` (the default), vias are displayed as the bounding box of each via. When set to `t` via attributes are graphically displayed, resulting in detailed images. For large designs, setting via shapes to `t` will increase the amount of time it takes to render a design.

GUI Equivalent

Command	*Options – Display*
Field	*Via Shapes* (Display Options Form)

Examples

```
envGetVal("layout" "displayViaShapes")
envSetVal("layout" "displayViaShapes" 'boolean t)
envSetVal("layout" "displayViaShapes" 'boolean nil)
envGetVal("graphic" "displayViaShapes")
envSetVal("graphic" "displayViaShapes" 'boolean t)
envSetVal("graphic" "displayViaShapes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

doNotShowChopPRBDialog

```
layout doNotShowChopPRBDialog boolean { t | nil }
```

Description

Lets you select if you want to display the warning dialog box when the `Chop` command is executed outside the PR boundary or when there is no PR boundary in the design. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "doNotShowChopPRBDialog")
envSetVal("layout" "doNotShowChopPRBDialog" 'boolean t)
envSetVal("layout" "doNotShowChopPRBDialog" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dotsOn

```
layout dotsOn boolean { t | nil }
```

```
graphic dotsOn boolean { t | nil }
```

Description

Determines whether to display diamond markers at via origins in the cellview (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Dot Pins* (Display Options Form)

Examples

```
envGetVal("layout" "dotsOn")
envSetVal("layout" "dotsOn" 'boolean t)
envSetVal("layout" "dotsOn" 'boolean nil)
envGetVal("graphic" "dotsOn")
envSetVal("graphic" "dotsOn" 'boolean t)
envSetVal("graphic" "dotsOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dotStyleMajorGrid

```
layout dotStyleMajorGrid boolean { t | nil }
graphic dotStyleMajorGrid boolean { t | nil }
```

Description

Dims the dots of the major grid. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Dim Major Dots* (Display Options Form)

Examples

```
envGetVal("layout" "dotStyleMajorGrid")
envSetVal("layout" "dotStyleMajorGrid" 'boolean t)
envSetVal("layout" "dotStyleMajorGrid" 'boolean nil)
envGetVal("graphic" "dotStyleMajorGrid")
envSetVal("graphic" "dotStyleMajorGrid" 'boolean t)
envSetVal("graphic" "dotStyleMajorGrid" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawAxesOn

```
layout drawAxesOn boolean { t | nil }
graphic drawAxesOn boolean { t | nil }
```

Description

Determines whether you see the cellview axes (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Axes* (Display Options Form)

Examples

```
envGetVal("layout" "drawAxesOn")
envSetVal("layout" "drawAxesOn" 'boolean t)
envSetVal("layout" "drawAxesOn" 'boolean nil)
envGetVal("graphic" "drawAxesOn")
envSetVal("graphic" "drawAxesOn" 'boolean t)
envSetVal("graphic" "drawAxesOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawDottedGridOn

```
layout drawDottedGridOn boolean { t | nil }
```

```
graphic drawDottedGridOn boolean { t | nil }
```

Description

Determines whether the grid is displayed as dots (`t`) or lines (`nil`), when `drawGridOn` is `t`. The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Grid Controls – Type – dots* and *lines* (Display Options Form)

Examples

```
envGetVal("layout" "drawDottedGridOn")
envSetVal("layout" "drawDottedGridOn" 'boolean t)
envSetVal("layout" "drawDottedGridOn" 'boolean nil)
envGetVal("graphic" "drawDottedGridOn")
envSetVal("graphic" "drawDottedGridOn" 'boolean t)
envSetVal("graphic" "drawDottedGridOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawEngine

```
graphic drawEngine int { 0 | 1 }
```

Description

Determines whether or not all data in a cellview is rendered. If `0`, all data is drawn including that which will be obscured by data on overlying layers. If `1`, only data which will ultimately be viewable is drawn. This function works best on large designs with closely placed data. It is slower on small designs, as the pruning of overlapping data takes longer than redrawing a small amount of data. The default is `0`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "drawEngine")
envSetVal("graphic" "drawEngine" 'int 1)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawGridOn

```
layout drawGridOn boolean { t | nil }
graphic drawGridOn boolean { t | nil }
```

Description

Determines whether the grid is displayed (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Grid Controls – Type – none* (Display Options Form)

Examples

```
envGetVal("layout" "drawGridOn")
envSetVal("layout" "drawGridOn" 'boolean t)
envSetVal("layout" "drawGridOn" 'boolean nil)
envGetVal("graphic" "drawGridOn")
envSetVal("graphic" "drawGridOn" 'boolean t)
envSetVal("graphic" "drawGridOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawInstancePinNames

```
layout drawInstancePinNames boolean { t | nil }
graphic drawInstancePinNames boolean { t | nil }
```

Description

Determines whether pin names are displayed in instances (`t`) or not (`nil`). The layout default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Instance Pin Names* (Display Options Form)

Examples

```
envGetVal("layout" "drawInstancePinNames")
envSetVal("layout" "drawInstancePinNames" 'boolean t)
envSetVal("layout" "drawInstancePinNames" 'boolean nil)
envGetVal("graphic" "drawInstancePinNames")
envSetVal("graphic" "drawInstancePinNames" 'boolean t)
envSetVal("graphic" "drawInstancePinNames" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawInstancePins

```
layout drawInstancePins boolean { t | nil }
graphic drawInstancePins boolean { t | nil }
```

Description

Determines whether non-maskable pins are displayed (`t`) or not (`nil`). The layout default is `nil`, the graphic default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Instance Pins* (Display Options Form)

Examples

```
envGetVal("layout" "drawInstancePins")
envSetVal("layout" "drawInstancePins" 'boolean t)
envSetVal("layout" "drawInstancePins" 'boolean nil)
envGetVal("graphic" "drawInstancePins")
envSetVal("graphic" "drawInstancePins" 'boolean t)
envSetVal("graphic" "drawInstancePins" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawLabelsSurround

```
graphic drawLabelsSurround boolean { t | nil }
```

Description

Draws a halo around label and text displays. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "drawLabelsSurround")
envSetVal("graphic" "drawLabelsSurround" 'boolean t)
envSetVal("graphic" "drawLabelsSurround" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawPRBoundary

```
graphic drawPRBoundary boolean { t | nil }
```

Description

Determines whether the place and route boundary of instances is displayed (`t`) or the instance bounding box is displayed (`nil`). The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "drawPRBoundary")
envSetVal("graphic" "drawPRBoundary" 'boolean t)
envSetVal("graphic" "drawPRBoundary" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawSmallLabelOrigin

```
graphic drawSmallLabelOrigin boolean { t | nil }
```

Description

Determines whether the small labels should be displayed in the design (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "drawSmallLabelOrigin")
envSetVal("graphic" "drawSmallLabelOrigin" 'boolean t)
envSetVal("graphic" "drawSmallLabelOrigin" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawSurroundingOn

```
layout drawSurroundingOn boolean { t | nil }
graphic drawSurroundingOn boolean { t | nil }
```

Description

Determines whether the surrounding design is displayed when you edit in place (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*EIP Surround* (Display Options Form)

Examples

```
envGetVal("layout" "drawSurroundingOn")
envSetVal("layout" "drawSurroundingOn" 'boolean t)
envSetVal("layout" "drawSurroundingOn" 'boolean nil)
envGetVal("graphic" "drawSurroundingOn")
envSetVal("graphic" "drawSurroundingOn" 'boolean t)
envSetVal("graphic" "drawSurroundingOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawTraversedInstBbox

```
layout drawTraversedInstBbox boolean { t | nil }
graphic drawTraversedInstBbox boolean { t | nil }
```

Description

Draws a bounding box around an instance that traverses to a size smaller than the fitIntoClusterBoundary. The traversing usually occurs when you zoom out of the design. The default value of the environment variable is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Traversed Instance Bbox* (Display Options Form)

Examples

```
envGetVal("layout" "drawTraversedInstBbox")
envSetVal("layout" "drawTraversedInstBbox" 'boolean t)
envSetVal("layout" "drawTraversedInstBbox" 'boolean nil)
envGetVal("graphic" "drawTraversedInstBbox")
envSetVal("graphic" "drawTraversedInstBbox" 'boolean t)
envSetVal("graphic" "drawTraversedInstBbox" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drawTraversedInstName

```
layout drawTraversedInstName boolean { t | nil }
graphic drawTraversedInstName boolean { t | nil }
```

Description

Displays the name of the instance for the traversed instance. The default value of the environment variable is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Traversed Instance Name* (Display Options Form)

Examples

```
envGetVal("layout" "drawTraversedInstName")
envSetVal("layout" "drawTraversedInstName" 'boolean t)
envSetVal("layout" "drawTraversedInstName" 'boolean nil)
envGetVal("graphic" "drawTraversedInstName")
envSetVal("graphic" "drawTraversedInstName" 'boolean t)
envSetVal("graphic" "drawTraversedInstName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drdEditSmartSnapAllowedWidthSnap

```
layout drdEditSmartSnapAllowedWidthSnap boolean { t | nil }
```

Description

Determines whether discrete dimension is enabled (`t`) or not (`nil`). The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drdEditSmartSnapAllowedWidthSnap")
envSetVal("layout" "drdEditSmartSnapAllowedWidthSnap" 'boolean t)
envSetVal("layout" "drdEditSmartSnapAllowedWidthSnap" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

drfPath

`grahic drfPath string` *`path`*

Description

Specifies the path to the Display Resource File. The default is none.

GUI Equivalent

None

Examples

```
envGetVal("layout" "drfPath")
```

```
envSetVal("layout" "drfPath" 'string "~/.display.drf")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicHilightOn

```
layout dynamicHilightOn boolean { t | nil }
graphic dynamicHilightOn boolean { t | nil }
```

Description

Determines whether dynamic highlighting is on (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Dynamic Highlight* (Display Options Form)

Examples

```
envGetVal("layout" "dynamicHilightOn")
envSetVal("layout" "dynamicHilightOn" 'boolean t)
envSetVal("layout" "dynamicHilightOn" 'boolean nil)
envGetVal("graphic" "dynamicHilightOn")
envSetVal("graphic" "dynamicHilightOn" 'boolean t)
envSetVal("graphic" "dynamicHilightOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementBackgroundColor

`layout dynamicMeasurementBackgroundColor string` *`color`*

Description

Lets you select the background color for the dynamic measurement labels. The default is `black`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "dynamicMeasurementBackgroundColor")
envSetVal("layout" "dynamicMeasurementBackgroundColor" 'string "black")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementDrawBackground

```
layout dynamicMeasurementDrawBackground boolean { t | nil }
```

Description

Turns off the background color. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "dynamicMeasurementDrawBackground")
envSetVal("layout" "dynamicMeasurementDrawBackground" 'boolean t)
envSetVal("layout" "dynamicMeasurementDrawBackground" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementDisplay

```
layout dynamicMeasurementDisplay boolean { t | nil }
```

Description

Enables you to dynamically display measurement between the pointer and the closest edges. The default is `nil`.

GUI Equivalent

Command	*Tools – Create Measurement*
Form Field	*Dynamic Measurement* (Create Measurement Form)

Examples

```
envGetVal("layout" "dynamicMeasurementDisplay")
envSetVal("layout" "dynamicMeasurementDisplay" 'boolean t)
envSetVal("layout" "dynamicMeasurementDisplay" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementEdge

```
layout dynamicMeasurementEdge boolean { t | nil }
```

Description

Lets you dynamically display measurement between the two vertices of an edge, the edge closest to the pointer. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "dynamicMeasurementEdge")
envSetVal("layout" "dynamicMeasurementEdge" 'boolean t)
envSetVal("layout" "dynamicMeasurementEdge" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementFontSize

`layout dynamicMeasurementFontSize int` *`any_positive_whole_number`*

Description

Specifies the font size for dynamic measurement labels. The default is `12`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "dynamicMeasurementFontSize")
envSetVal("layout" "dynamicMeasurementFontSize" 'int 14)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementFromCursor

```
layout dynamicMeasurementFromCursor boolean { t | nil }
```

Description

Displays measurement between the pointer and closest edge. It also displays the distance between the closest edges. When set to `nil`, the dynamic measurement displayed is the distance between closest edges. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "dynamicMeasurementFromCursor")
envSetVal("layout" "dynamicMeasurementFromCursor" 'boolean t)
envSetVal("layout" "dynamicMeasurementFromCursor" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementLayerScope

`layout dynamicMeasurementLayerScope string` *`layerScope`*

Description

Lets you select the layers to be considered for dynamic measurement. The options are `All`, `Active`, `Edge`, `Palette`, or `Selectable layers`. The default is `All`.

GUI Equivalent

Command	*Tools – Create Measurement*
Form Field	*Layer Scope* (Create Measurement Form)

Examples

```
envGetVal("layout" "dynamicMeasurementBackgroundColor")
envSetVal("layout" "dynamicMeasurementBackgroundColor" 'string "Active")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicMeasurementTextColor

```
layout dynamicMeasurementTextColor string { y0 | y1 | y2 | y3 | y4 | y5 | y6 | y7
| y8 | y9 }
```

Description

Lets you set and dynamically change the color of the measurement information. The default is `y0`.

GUI Equivalent

Command	*Options – Dynamic Display – Dynamic Measurement*
Field	*Color* (Dynamic Display Form)

Examples

```
envGetVal("layout" "dynamicMeasurementTextColor")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y0")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y1")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y2")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y3")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y4")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y5")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y6")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y7")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y8")
envSetVal("layout" "dynamicMeasurementTextColor" 'string "y9")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

dynamicProbe

```
layout dynamicProbe boolean { t | nil }
```

Description

Dynamically probes any object over which the mouse pointer hovers.

GUI Equivalent

Command	*Options – Highlight*
Field	*Dynamic Highlight* (Highlight Options Form)

Examples

```
envGetVal("layout" "dynamicProbe")
envSetVal("layout" "dynamicProbe" 'boolean t)
envSetVal("layout" "dynamicProbe" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

editInstanceBoundaryMode

```
layout editInstanceBoundaryMode cyclic { BBox | prBoundary | snapBoundary }
```

Description

Specifies whether the *Flip* and *Rotate* commands use the instance bounding box (`BBox`), the place and route boundary (`prBoundary`), or the snap boundary (`snapBoundary`) as the instance border when operating on instances and mosaics.

For example, if you have a standard cell with a large label that extends well beyond the PR boundary, the bounding box of that cell will be much larger than its place and route boundary. Set this environment variable to `prBoundary` to have *Flip* and *Rotate* ignore the bounding box and consider only the PR boundary when operating on the cell.

Note: The default is `BBox`. If you set this environment variable to `prBoundary` or `snapBoundary` and the instance or mosaic does not have the corresponding boundary, then `BBox` is used instead.

GUI Equivalent

Command	*Options – Editor*
Field	*Instance Editing Mode* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "editInstanceBoundaryMode")
envSetVal("layout" "editInstanceBoundaryMode" 'cyclic "BBox")
envSetVal("layout" "editInstanceBoundaryMode" 'cyclic "prBoundary")
envSetVal("layout" "editInstanceBoundaryMode" 'cyclic "snapBoundary")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

editorOptionsFile

```
layout editorOptionsFile string "~/.cdsenv"
```

Description

Specifies the file to which you want to store or from which you want to load the settings.

GUI Equivalent

Command	*Options – Editor*
Field	*File* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "editorOptionsFile")
envSetVal("layout" "editorOptionsFile" 'string "~/.cdsenv")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

eipHighlightColor

```
layout eipHighlightColor string haloColor
```

Description

Specifies the color of the halo around a figure group and instance.

GUI Equivalent

None

Examples

```
envGetVal("layout" "eipHighlightColor")
envSetVal("layout" "eipHighlightColor" 'string "magenta")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

enableAlignToolBarReferenceObject

```
layout enableAlignToolBarReferenceObject boolean { t | nil }
```

Description

When set to (`t`), enables the reference object on the *Align* toolbar. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "enableAlignToolBarReferenceObject")
envSetVal("layout" "enableAlignToolBarReferenceObject" 'boolean t)
envSetVal("layout" "enableAlignToolBarReferenceObject" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

enableCopyAndPaste

```
layout enableCopyAndPaste boolean { t | nil }
```

Description

When set to on (`t`), in the `Copy` command, the selected objects and the reference point are automatically stored in the buffer after the reference point for the copy command has been specified. You can use the `Paste` command to paste the objects stored in the buffer. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "enableCopyAndPaste")
envSetVal("layout" "enableCopyAndPaste" 'boolean t)
envSetVal("layout" "enableCopyAndPaste" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

enableEIPEnhancements

```
layout enableEIPEnhancements boolean { t | nil }
```

Description

When set to on (`t`), the following edit in place enhancements are available:

- When editing in place in a hierarchy, clicking a shape at the top level performs return to top.
- For figures inside a group inside an instance, by default, the EIP target is the figure (descend into instance and descend into the group). You can use the `Space` key to cycle through the instance and group or just the instance.
- When you return from instance and group, you can specify the options in the G*roup Return* form.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "enableEIPEnhancements")
envSetVal("layout" "enableEIPEnhancements" 'boolean t)
envSetVal("layout" "enableEIPEnhancements" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

enableSyncCloneEditing

```
layout enableSyncCloneEditing cyclic { Ask | Never | Always }
```

Description

Controls the display of the Edit Synchronous Clone form.

- `Ask`: The Edit Synchronous Clone form is displayed when you edit a clone.
- `Never`: The Edit Synchronous Clone form is not displayed when you edit a clone and the clone is not edited.
- `Always`: The Edit Synchronous Clone form is not displayed when you edit a clone and the clone is edited.

The default is `Ask`.

GUI Equivalent

Command	*Edit Synchronous Clone*
Field	*Apply same action to all other clones* (Edit Synchronous Clone Form)

Examples

```
envGetVal("layout" "enableSyncCloneEditing")
envSetVal("layout" "enableSyncCloneEditing" 'cyclic "Ask")
envSetVal("layout" "enableSyncCloneEditing" 'cyclic "Never")
envSetVal("layout" "enableSyncCloneEditing" 'cyclic "Always")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

expandPinBusExpression

```
layout expandPinBusExpression boolean { t | nil }
```

Description

When set to (`t`), expands the pin bus expression. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "expandPinBusExpression")
envSetVal("layout" "expandPinBusExpression" 'boolean t)
envSetVal("layout" "expandPinBusExpression" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extendPathLengthCheck

```
layout extendPathLengthCheck boolean { t | nil }
```

Description

When set to on (`t`), paths or path segments created with a segment of less than 1/2 the path width is checked and a warning is issued. When set to off (`nil`), no check is made and no warning is issued. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "extendPathLengthCheck")
envSetVal("layout" "extendPathLengthCheck" 'boolean t)
envSetVal("layout" "extendPathLengthCheck" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractAfterGenerateAll

```
layout extractAfterGenerateAll boolean { t | nil }
```

Description

Runs connectivity extraction as part of the *Generate All From Source* command. Switch on this environment variable to see the incomplete nets in the design immediately after layout generation has run.

Note: This option has no effect on either the *Generate Selected From Source* or *Update Components And Nets* commands. Connectivity extraction for these commands is controlled using the extractEnabled environment variable.

GUI Equivalent

Command	(In Layout XL) *Connectivity – Generate – All From Source – Generate*
Field	*Extract Connectivity after Generation(Generate Layout Form)*

Examples

```
envGetVal("layout" "extractAfterGenerateAll")
envSetVal("layout" "extractAfterGenerateAll" 'boolean t)
envSetVal("layout" "extractAfterGenerateAll" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractCellviewIllegalConnectionLimit

`layout extractCellviewIllegalConnectionLimit int integer`

Description

Specifies the maximum number of illegal hierarchical connection violations generated by the extractor for a particular cellview.

The default is `1000`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Limits*
Field	*Maximum number of illegal connections: By cellview* (Layout XL Options form)

Examples

```
envGetVal("layout" "extractCellviewIllegalConnectionLimit")
envSetVal("layout" "extractCellviewIllegalConnectionLimit" 'int 750)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractCellviewOpenViolationLimit

```
layout extractCellviewOpenViolationLimit int integer
```

Description

Specifies the maximum number of open violations generated by the extractor for a particular cellview. Limiting the number can improve performance when you are editing either a large design or a net with a large number of instance terminals and terminals in an unrouted design.

The default is 5000.

Important

This environment variable is considered only when extractEnabled and extractVerifyOpenViolations are switched on.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Limits*
Field	*Maximum number of opens: By cellview*

Examples

```
envGetVal("layout" "extractCellviewOpenViolationLimit")
envSetVal("layout" "extractCellviewOpenViolationLimit" 'int 3000)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractCellviewShortViolationLimit

```
layout extractCellviewShortViolationLimit int integer
```

Description

Specifies the maximum number of short violations generated by the extractor for a particular cellview.

The default is `1000`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Limits*
Field	*Maximum number of shorts: By cellview*

Examples

```
envGetVal("layout" "extractCellviewShortViolationLimit")
envSetVal("layout" "extractCellviewShortViolationLimit" 'int 750)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractCellviewUnnecessaryPseudoParallelConnectionLimit

`layoutXL extractCellviewUnnecessaryPseudoParallelConnectionLimit int integer`

Description

Specifies the maximum number of unnecessary pseudo-parallel connection violations generated by the extractor for a particular cellview.

The default is `1000`.

GUI Equivalent

Command:	*Options - Connectivity - Connectivity - Limits*
Field:	*Maximum number of unnecessary pseudo-parallel connections*
	(Limits tab)

Examples

```
envGetVal("layoutXL" "extractCellviewUnnecessaryPseudoParallelConnectionLimit")
envSetVal("layoutXL" "extractCellviewUnnecessaryPseudoParallelConnectionLimit"
'int 500)
envSetVal("layoutXL" "extractCellviewUnnecessaryPseudoParallelConnectionLimit"
'int 1000)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractConnectUnassignedInstTerms

```
layout extractConnectUnassignedInstTerms boolean { t | nil }
```

Description

Ensures that an instance terminal is automatically connected to the net assigned to the overlapping shape.

The default is `t`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Connect unassigned instance pins*

Examples

```
envGetVal("layout" "extractConnectUnassignedInstTerms")
envSetVal("layout" "extractConnectUnassignedInstTerms" 'boolean t)
envSetVal("layout" "extractConnectUnassignedInstTerms" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractConsiderAllCutLayersInVias

```
layout extractConsiderAllCutLayersInVias boolean { t | nil }
```

Description

Considers all cut layers inside vias as extractable, even if the cut layer is not listed in the `validLayers` constraint.

The default is `t`.

Important

Cadence recommends that you list all the extractable layers in the `validLayers` constraint. If there are no extractable layers defined in the constraint group, Layout XL is launched with the connectivity extractor disabled.

GUI Equivalent

None

Examples

```
envGetVal("layout" "extractConsiderAllCutLayersInVias")
envSetVal("layout" "extractConsiderAllCutLayersInVias" 'boolean t)
envSetVal("layout" "extractConsiderAllCutLayersInVias" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

`validLayers`

`viaDef`

© 2020 All Rights Reserved.

extractEnabled

```
layout extractEnabled boolean { t | nil }
```

Description

Enables incremental connectivity extraction during interactive editing, meaning that the connectivity model is updated automatically as you edit your design.

You specify how the extraction is performed and what is displayed in the layout window using the following environment variables. Indentation is used to show the dependencies between the environment variables.

- extractStopLevel
- extractVerifyMustConnectMarkers
- extractVerifyOpenViolations
  - extractCellviewOpenViolationLimit
  - extractNetOpenViolationLimit
  - extractVerifyUnimplementedInstTerms
- extractVerifyWeakMarkers

The default is `t`.

When set to nil, the connectivity model is not updated during interactive editing and no new short or open violation markers are generated in the layout window.

Tip

When the incremental extractor is switched off, you can still run connectivity extraction at any time using the *Connectivity – Extract Layout* command.

© 2020 All Rights Reserved.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Update connectivity information when design is modified*

Examples

```
envGetVal("layout" "extractEnabled")
envSetVal("layout" "extractEnabled" 'boolean t)
envSetVal("layout" "extractEnabled" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

Connectivity Extraction

© 2020 All Rights Reserved.

extractEnableVirtualConnections

```
layout extractEnableVirtualConnections boolean { t | nil}
```

Description

Enables connectivity extraction from virtual-connect labels that are either attached or stamped to a shape.

The default is `t`, which means that sets of physically connected shapes that are on the same net and tagged by a label comprising the net name followed by a virtual-connect separator are considered as connected.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Enable virtual connections*

Examples

```
envGetVal("layout" "extractEnableVirtualConnections")
envSetVal("layout" "extractEnableVirtualConnections" 'boolean t)
envSetVal("layout" "extractEnableVirtualConnections" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractHierExtractSearchDepth

```
layout extractHierExtractSearchDepth 'int { 1 | 2 ... 32 }
```

Description

Specifies how many levels down the hierarchy to search for cells to extract during hierarchical extraction. Specify an integer value between `1` and `32`. The default is `1`.

GUI Equivalent

Command	(In Layout XL) *Connectivity – Extract Layout*
Field	*Search Depth* (Extract Layout form)

Examples

```
envGetVal("layout" "extractHierExtractSearchDepth")
envSetVal("layout" "extractHierExtractSearchDepth" 'int 3)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractKeepShapeAssignment

```
layout extractKeepShapeAssignment boolean { t | nil }
```

Description

Ensures that the connectivity of a design is preserved by retaining the assignment of shapes on a net.

The default is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Keep shape assignment* (Layout XL Options form)

Examples

```
envGetVal("layout" "extractKeepShapeAssignment")
envSetVal("layout" "extractKeepShapeAssignment" 'boolean  t)
envSetVal("layout" "extractKeepShapeAssignment" 'boolean  nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractNetOpenViolationLimit

```
layout extractNetOpenViolationLimit int integer
```

Description

Specifies the maximum number of open violations generated by the extractor for a particular net. This improves interactive performance when you are editing either a large design or a net with a large number of instance terminals and terminals in an unrouted design.

The default is 100.

Important

This environment variable is considered only when extractEnabled and extractVerifyOpenViolations are switched on.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Limits*
Field	*Maximum number of opens: By net* (Layout XL Options form)

Examples

```
envGetVal("layout" "extractNetOpenViolationLimit")
envSetVal("layout" "extractNetOpenViolationLimit" 'int 70)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractNetShortViolationLimit

```
layout extractNetShortViolationLimit int integer
```

Description

Specifies the maximum number of short violations generated by the extractor for a particular net.

The default is `100`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Limits*
Field	*Maximum number of shorts: By net* (Layout XL Options form)

Examples

```
envGetVal("layout" "extractNetShortViolationLimit")
envSetVal("layout" "extractNetShortViolationLimit" 'int 75)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractReinitOnStopLevelChange

```
layout extractReinitOnStopLevelChange 'boolean { t | nil }
```

Description

Specifies if the extractor needs to be reinitialized when the extract stop level is changed.

The default is `nil`.

GUI Equivalents

None

Examples

```
envGetVal("layout" "extractReinitOnStopLevelChange")
envSetVal("layout" "extractReinitOnStopLevelChange" 'boolean t)
envSetVal("layout" "extractReinitOnStopLevelChange" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractRemoveStickyProperty

```
layout extractRemoveStickyProperty boolean { t | nil }
```

Description

Removes the lxStickyNet property on shapes that are connected to a connectivity source, such as a pin, instance terminal, or a label.

The default is `nil`.

GUI Equivalent

Command:	*Options – LayoutXL – Connectivity (tab)*
Field:	*Remove lxStickyNet property on non-floating shapes* (Connectivity)

Examples

```
envGetVal("layout" "extractRemoveStickyProperty")
envSetVal("layout" "extractRemoveStickyProperty" 'boolean t)
envSetVal("layout" "extractRemoveStickyProperty" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

Layout XL Options Form

© 2020 All Rights Reserved.

extractRunOnInit

```
layout extractRunOnInit boolean { t | nil }
```

Description

Runs connectivity extraction automatically when you launch Layout XL (or GXL) from Layout L.

The default is `nil`, meaning that connectivity extraction is not run automatically when you launch either application.

Caution

Enabling this environment variable means that connectivity extraction is run even if it not required and adversely impacts the performance of Layout XL during initialization. Cadence recommends that you do not enable this environment variable by default, but that you use the Connectivity – Extract Layout command when you know it is required.

GUI Equivalent

None

Examples

```
envGetVal("layout" "extractRunOnInit")
envSetVal("layout" "extractRunOnInit" 'boolean t)
envSetVal("layout" "extractRunOnInit" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractStopLevel

```
layout extractStopLevel int { 0 | 1 | 2 | ... | 32 }
```

Description

Specifies how much of the design hierarchy the extractor considers when extracting the design. The default is `0`, which means that the extractor extracts only

- Top-level objects; that is, shapes, vias, and instances used to implement pins
- Instance pins of top-level instances. These instance pins represent the interface of top-level instances to the lower levels in the design and are themselves considered to be at the top level.

Increasing the `extractStopLevel` increases the extractor's visibility of the shapes inside the hierarchy. For example,

extractStopLevel	Means...
`1`	extractStopLevel 0 + objects at level 1 + instance pins at level 2
`2`	extractStopLevel 1 + objects at level 2 + instance pins at level 3

Whatever the `extractStopLevel`, the extractor only ever extracts the top level; i.e., it only ever changes the connectivity of objects or creates markers at the top level.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Extract connectivity to level* (Layout XL Options form)

Examples

```
envGetVal("layout" "extractStopLevel")
envSetVal("layout" "extractStopLevel" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyAnalogSigType

```
layout extractVerifyAnalogSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Analog`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Analog*

Examples

```
envGetVal("layout" "extractVerifyAnalogSigType")
envSetVal("layout" "extractVerifyAnalogSigType" 'boolean t)
envSetVal("layout" "extractVerifyAnalogSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyBigMPP

```
layout extractVerifyBigMPP boolean { t | nil }
```

Description

Specifies that the connectivity extractor is to extract through large multipart paths. A large multipart path is one that contains more than 1000 elements. These are typically used to implement physical-only structures such as isolation rings, and therefore do not need to be extracted.

The default is `nil`, which means that the extractor ignores them.

Caution

Setting this environment variable to `t` can adversely affect the performance of the extractor.

GUI Equivalent

None

Examples

```
envGetVal("layout" "extractVerifyBigMPP")
envSetVal("layout" "extractVerifyBigMPP" 'boolean t)
envSetVal("layout" "extractVerifyBigMPP" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyClockSigType

```
layout extractVerifyClockSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Clock`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Clock*

Examples

```
envGetVal("layout" "extractVerifyClockSigType")
envSetVal("layout" "extractVerifyClockSigType" 'boolean t)
envSetVal("layout" "extractVerifyClockSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyGroundSigType

```
layout extractVerifyGroundSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Ground`. These markers can be reviewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Ground*

Examples

```
envGetVal("layout" "extractVerifyGroundSigType")
envSetVal("layout" "extractVerifyGroundSigType" 'boolean t)
envSetVal("layout" "extractVerifyGroundSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyIllegalHierConnections

```
layout extractVerifyIllegalHierConnections boolean { t | nil }
```

Description

Reports invalid hierarchical connections in the current cellview.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Illegal hierarchical connections*

Examples

```
envGetVal("layout" "extractVerifyIllegalHierConnections")
envSetVal("layout" "extractVerifyIllegalHierConnections" 'boolean t)
envSetVal("layout" "extractVerifyIllegalHierConnections" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyIllegalOverlapsForStoppedColoredShapes

```
layout extractVerifyIllegalOverlapsForStoppedColoredShapes boolean { t | nil }
```

Description

Reports invalid overlap violations for color-dependent stopping of shapes.

The default is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Illegal overlaps for stopped colored shapes*

Examples

```
envGetVal("layout" "extractVerifyIllegalOverlapsForStoppedColoredShapes")
envSetVal("layout" "extractVerifyIllegalOverlapsForStoppedColoredShapes" 'boolean
t)
envSetVal("layout" "extractVerifyIllegalOverlapsForStoppedColoredShapes" 'boolean
nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyIncompleteMustConnectAllPins

```
layout extractVerifyIncompleteMustConnectAllPins boolean { t | nil }
```

Description

Controls if incomplete `mustConnectAllPins` must be verified or not.

The default value is `t`.

GUI Equivalent

None.

Examples

```
envGetVal("layout" "extractVerifyIncompleteMustConnectAllPins")
envSetVal("layout" "extractVerifyIncompleteMustConnectAllPins" 'boolean t)
envSetVal("layout" "extractVerifyIncompleteMustConnectAllPins" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyInvalidConnections

```
layout extractVerifyInvalidConnections boolean { t | nil }
```

Description

Controls if invalid connections must be verified or not.

The default value is `t`.

GUI Equivalent

None.

Examples

```
envGetVal("layout" "extractVerifyInvalidConnections")
envSetVal("layout" "extractVerifyInvalidConnections" 'boolean t)
envSetVal("layout" "extractVerifyInvalidConnections" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyMustConnectMarkers

```
layout extractVerifyMustConnectMarkers boolean { t | nil }
```

Description

Displays the must-connect violation markers generated by the connectivity extractor. When set to `t`, the extractor shows top-level must-connect violations in the layout window; when set to `nil`, it does not.

The default is `t`.

Important

This environment variable is considered only when extractEnabled is switched on. It shows only violations at the current top level. Adjusting the `extractStopLevel` has no effect on the violations shown. To show markers at a lower level, you must edit the lower level cellview in place or run hierarchical extraction.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Verify must-connect violations*

Examples

```
envGetVal("layout" "extractVerifyMustConnectMarkers")
envSetVal("layout" "extractVerifyMustConnectMarkers" 'boolean t)
envSetVal("layout" "extractVerifyMustConnectMarkers" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyOpenViolations

```
layout extractVerifyOpenViolations boolean { t | nil }
```

Description

Toggles the verification and display of open violations in the design.

Important

This environment variable is considered only when extractEnabled is switched on.

Tip

To improve interactive performance for large designs or unrouted designs containing nets with a large number of instance terminals and terminals, use the extractCellviewOpenViolationLimit and extractNetOpenViolationLimit environment variables. They are considered only when `extractVerifyOpenViolations` is set to `t`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Verify open violations*

Examples

```
envGetVal("layout" "extractVerifyOpenViolations")
envSetVal("layout" "extractVerifyOpenViolations" 'boolean t)
envSetVal("layout" "extractVerifyOpenViolations" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyOpticalPortRadiusMatch

```
layout extractVerifyOpticalPortRadiusMatch boolean { t | nil }
```

Description

Checks if the radius of the overlapping optical ports match to form a valid connection.

The default value is `nil`.

GUI Equivalent

None.

Examples

```
envGetVal("layout" "extractVerifyOpticalPortRadiusMatch")
envSetVal("layout" "extractVerifyOpticalPortRadiusMatch" 'boolean t)
envSetVal("layout" "extractVerifyOpticalPortRadiusMatch" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyPowerSigType

```
layout extractVerifyPowerSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Power`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Power*

Examples

```
envGetVal("layout" "extractVerifyPowerSigType")
envSetVal("layout" "extractVerifyPowerSigType" 'boolean t)
envSetVal("layout" "extractVerifyPowerSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyResetSigType

```
layout extractVerifyResetSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Reset`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Reset*

Examples

```
envGetVal("layout" "extractVerifyResetSigType")
envSetVal("layout" "extractVerifyResetSigType" 'boolean t)
envSetVal("layout" "extractVerifyResetSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyScanSigType

```
layout extractVerifyScanSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Scan`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Scan*

Examples

```
envGetVal("layout" "extractVerifyScanSigType")
envSetVal("layout" "extractVerifyScanSigType" 'boolean t)
envSetVal("layout" "extractVerifyScanSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifySignalSigType

```
layout extractVerifySignalSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `Signal`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – Signal*

Examples

```
envGetVal("layout" "extractVerifySignalSigType")
envSetVal("layout" "extractVerifySignalSigType" 'boolean t)
envSetVal("layout" "extractVerifySignalSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyTieHighSigType

```
layout extractVerifyTieHighSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `TieHigh`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – TieHigh*

Examples

```
envGetVal("layout" "extractVerifyTieHighSigType")
envSetVal("layout" "extractVerifyTieHighSigType" 'boolean t)
envSetVal("layout" "extractVerifyTieHighSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyTieLowSigType

```
layout extractVerifyTieLowSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `TieLow`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – TieLow*

Examples

```
envGetVal("layout" "extractVerifyTieLowSigType")
envSetVal("layout" "extractVerifyTieLowSigType" 'boolean t)
envSetVal("layout" "extractVerifyTieLowSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyTieOffSigType

```
layout extractVerifyTieOffSigType boolean { t | nil }
```

Description

Creates markers for nets that have their *Signal type* set as `TieOff`. These markers can be viewed in the Annotation Browser.

The default value is `t`.

GUI Equivalent

Command:	*Connectivity – Connectivity – Annotations*
Field:	*Signal types – TieOff*

Examples

```
envGetVal("layout" "extractVerifyTieOffSigType")
envSetVal("layout" "extractVerifyTieOffSigType" 'boolean t)
envSetVal("layout" "extractVerifyTieOffSigType" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyWellSubstrateConnections

```
layout extractVerifyWellSubstrateConnections boolean { t | nil }
```

Description

Enables or disables the verification of the substrate in terms of open and short violations. The tool only supports a single, global substrate for the design being edited. Therefore, this variable should not be turned on if the design has local substrates. The default is `nil`.

Note: The tool does not support deep well layers either.

GUI Equivalent

None.

Examples

```
envSetVal("layout" "extractVerifyWellSubstrateConnections" 'boolean t)
```

Example 1: No open markers are created between two disjoint substrate vias on the same net because the vias short to each other through the substrate.

Example 2: A substrate short marker is created if a substrate via is created on a given net and placed over the substrate while the substrate is already polarized on another net by another substrate via.

```
envGetVal("layout" "extractVerifyWellSubstrateConnections")
envSetVal("layout" "extractVerifyWellSubstrateConnections" 'boolean t)
envSetVal("layout" "extractVerifyWellSubstrateConnections" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyUnimplementedInstTerms

```
layout extractVerifyUnimplementedInstTerms boolean { t | nil }
```

Description

Specifies that the connectivity extractor should consider instance terminals with pin shapes that are not extractable (including instance terminals with no pin shapes at all).

The default is `t`.

When set to `nil`, the extractor and the functions it serves ignore any instance terminal which has no pin shape or which has a pin shape on a layer that is not extractable. This allows Layout XL to extract based only on the extract layers defined in the technology file.

Important

This environment variable is considered only when extractEnabled and extractVerifyOpenViolations are switched on.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Verify unimplemented instance terminall*

Examples

```
envGetVal("layout" "extractVerifyUnimplementedInstTerms")
envSetVal("layout" "extractVerifyUnimplementedInstTerms" 'boolean t)
envSetVal("layout" "extractVerifyUnimplementedInstTerms" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyUnnecessaryPseudoParallelConnections

```
layoutXL extractVerifyUnnecessaryPseudoParallelConnections boolean { t | nil }
```

Description

Reports the overlap between two sub-nets of a pseudo-parallel net.

The default is `nil`.

GUI Equivalent

Command:	*Options - Connectivity - Connectivity - Annotation*
Field:	*Unnecessary pseudo-parallel connections* (Annotations tab)

Examples

```
envGetVal("layoutXL" "extractVerifyUnnecessaryPseudoParallelConnections")
envSetVal("layoutXL" "extractVerifyUnnecessaryPseudoParallelConnections" 'boolean
t)
envSetVal("layoutXL" "extractVerifyUnnecessaryPseudoParallelConnections" 'boolean
nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVerifyWeakMarkers

```
layout extractVerifyWeakMarkers boolean { t | nil }
```

Description

Displays the weak-connect violation markers generated by the connectivity extractor. When set to `t`, the extractor shows top-level weak-connect violations in the layout window; when set to `nil`, it does not.

The default is `t`.

Important

This environment variable is considered only when extractEnabled is switched on. It shows only violations at the current top level. Adjusting the `extractStopLevel` has no effect on the violations shown. To show markers at a lower level, you must edit the lower level cellview in place or run hierarchical extraction.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Verify weak-connect violations*

Examples

```
envGetVal("layout" "extractVerifyWeakMarkers")
envSetVal("layout" "extractVerifyWeakMarkers" 'boolean t)
envSetVal("layout" "extractVerifyWeakMarkers" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

extractVirtualConnectionSeparator

```
layout extractVirtualConnectionSeparator string "virtualConnectSeparator"
```

Description

Specifies the separator that the connectivity extractor uses to identify virtual-connect labels.

By default, the virtual-connect separator is colon (:). But, you can change the separator, as appropriate, using the *Use (identifier) to define nets as virtually connected* field in the Layout XL Options form.

For more information on virtual-connect label separators, see Virtual Connections. For information on connectivity extraction from virtually connected shapes, see Enabling Virtual Connections.

The default is `(":")`.

GUI Equivalent

Command	(In Layout XL) *Options – Layout XL – Connectivity – Controls*
Field	*Use (identifier) to define nets as virtually connected* (Layout XL Options form)

Examples

```
envGetVal("layout" "extractVirtualConnectionSeparator")
envSetVal("layout" "extractVirtualConnectionSeparator" 'string ":")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

filterSizeForResolution

```
graphic filterSizeForResolution string {(High highVal) | (Medium medVal)| (Low
     lowVal)}
```

Description

Enables you to customize the level of details, filter size applied to resolutions High, and/or Medium, and/or Low.

By default, this environment variable is defined as below:

```
graphic filterSizeForResolution ‘string "(High 2.0) (Medium 6.0) (Low
15.0)”
```

The table below lists the different filter sizes used and the `lowVal`, `MedVal`, and `HighVal` values defined in the `filterSizeForResolution` environment variable.

	Low	Medium	High	Very High
Shape filterSize	`lowVal` (def 15.0)	`medVal` (def 6.0)	`highVal` (def 2.0)	`0.0`
Highest Resolution for P&R Objects option deselected	`15.0`	`6.0`	`6.0`	`1.0`
Highest Resolution for P&R Objects option selected	`15.0`	`6.0`	`highVal`	`0.0`

GUI Equivalent

Command	*Options – Display*
Field	*Resolution* (Display Options Form)

Examples

```
envGetVal("graphic" "filterSizeForResolution")
```


© 2020 All Rights Reserved.

```
envSetVal{"graphic" "filterSizeForResolution" 'string {(High 3.0) | (Medium 7.0)|
(Low 18.0)}
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

fitIntoClusterBoundary

```
layout fitIntoClusterBoundary boolean { t | nil }
```

Description

In Virtuoso XL, places all objects in a cluster into the cluster boundary. Default is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Cluster Boundary*
Field	*Fit Into Boundary*

Examples

```
envGetVal("layout" "fitIntoClusterBoundary")
envSetVal("layout" "fitIntoClusterBoundary" 'boolean t)
envSetVal("layout" "fitIntoClusterBoundary" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

fixedWidthPaths

```
layout fixedWidthPaths boolean { t | nil }
```

Description

Specifies whether to override the technology file and use a fixed width by default when creating a path (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Create – Shape – Path*
Field	*Fixed Width* (Create Path Form)

Examples

```
envGetVal("layout" "fixedWidthPaths")
envSetVal("layout" "fixedWidthPaths" 'boolean t)
envSetVal("layout" "fixedWidthPaths" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

flattenExcludePcellListFile

`layout flattenExcludePcellListFile string` *`pathAndFileName`*

Description

Specifies the name of the file that contains the list of Pcells that should not be flattened. This file has a list of library, cell and view of Pcells. In case the file is not found, the Pcells are flattened and a warning message is displayed in the CIW. If the `flattenPCells` environment variable is set to `nil`, no Pcells are flattened and the no warning message is displayed in the CIW. If the `flattenPCells` environment variable is set to `t`, all Pcells except those specified in the file are flattened. A warning message is displayed for the Pcells specified in the file.

GUI Equivalent

None

Examples

```
envGetVal("layout" "flattenExcludePcellListFile")
```

```
envSetVal("layout" "flattenExcludePcellListFile" 'string "/project1/technology10/
technology10.excludePcellListFile")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

flattenMode

```
layout flattenMode cyclic { one level | displayed levels | user level}
```

Description

Specifies the default number of levels of hierarchy of a cell to flatten in `leHiFlatten()` or `leFlattenInst()`. The default is `one level`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Hierarchy Level* (Flatten Form)

Examples

```
envGetVal("layout" "flattenMode")
envSetVal("layout" "flattenMode" 'cyclic "one level")
envSetVal("layout" "flattenMode" 'cyclic "displayed levels")
envSetVal("layout" "flattenMode" 'cyclic "user level")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

flattenPCells

```
layout flattenPCells boolean { t | nil }
```

Description

Specifies whether Pcells are flattened by default by `leHiFlatten()` or `leFlattenInst()` (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Pcells* (Flatten Form)

Examples

```
envGetVal("layout" "flattenPCells")
envSetVal("layout" "flattenPCells" 'boolean t)
envSetVal("layout" "flattenPCells" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

flattenVias

```
layout flattenVias boolean { t | nil }
```

Description

Specifies whether geometries in the selected vias are flattened (`t`) or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Vias* (Flatten Form)

Examples

```
envGetVal("layout" "flattenVias")
envSetVal("layout" "flattenVias" 'boolean t)
envSetVal("layout" "flattenVias" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

flipAroundCombinedCenter

```
layout flipAroundCombinedCenter boolean { t | nil }
```

Description

Enables the multiple selected objects to be flipped around the center of the imaginary bounding box around the selected objects. The default value is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Flip Around Combined Center* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "flipAroundCombinedCenter")
envSetVal("layout" "flipAroundCombinedCenter" 'boolean t)
envSetVal("layout" "flipAroundCombinedCenter" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

followRoutingDirection

```
layout followRoutingDirection boolean { t | nil }
```

Description

Specifies whether the `Convert To PathSeg` command follows the routing direction. The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "followRoutingDirection")
envSetVal("layout" "followRoutingDirection" 'boolean t)
envSetVal("layout" "followRoutingDirection" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

fullSelectionMode

```
graphic fullSelectionMode string { enclosed | crossed | enclosedAndCrossed }
```

Description

Specifies whether the full selection mode is set to crossed (`crossed`), enclosed (`enclosed`) or enclosed and crossed (`enclosedAndCrossed`). Crossed Figures fully selects objects you partially enclose with the area selection box, and does not select fully enclosed objects. Enclosed Figures fully selects objects you completely enclose with the area selection box. Enclosed and Crossed Figures fully selects objects you completely or partially enclose with the area selection box. The default is `enclosed`.

GUI Equivalent

Command	*Options – Selection*
Field	*Full Mode* (Selection Options Form)

Examples

```
envGetVal("graphic" "fullSelectionMode")
envSetVal("graphic" "fullSelectionMode" 'string "enclosed")
envSetVal("graphic" "fullSelectionMode" 'string "crossed")
envSetVal("graphic" "fullSelectionMode" 'string "enclosedAndCrossed")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

ghostOfDraggedObjects

```
layout ghostOfDraggedObjects boolean { t | nil }
```

```
graphic ghostOfDraggedObjects boolean { t | nil }
```

Description

Allows viewing the ghost of the original object when it is being moved or stretched. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Dragged Object Ghost* (Display Options Form)

Examples

```
envGetVal("layout" "ghostOfDraggedObjects")
envSetVal("layout" "ghostOfDraggedObjects" 'boolean t)
envSetVal("layout" "ghostOfDraggedObjects" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravityApertureInPixels

`layout gravityApertureInPixels int` *`any_integer_greater_than_zero`*

Description

Specifies the aperture value when the snap distance between the pointer and an object (gravityDistance) is set to `Medium`. The default is `20`.

Note: The `gravityApertureInPixels` environment variable is applied only when the value of the gravityDistance environment variable set to `Medium`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "gravityApertureInPixels")
envSetVal("layout" "gravityApertureInPixels" 'int 15)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravityDistance

```
layout gravityDistance string { Small | Medium | Large | Very Large }
```

Description

Specifies the snap distance between the pointer and an object. You can choose from `Small`, `Medium`, `Large`, or `Very Large`. The default is `Medium`.

GUI Equivalent

Command	*Options – Editor*
Field	*Distance* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "gravityDistance")
envSetVal("layout" "gravityDistance" 'string "Small")
envSetVal("layout" "gravityDistance" 'string "Medium")
envSetVal("layout" "gravityDistance" 'string "Large")
envSetVal("layout" "gravityDistance" 'string "Very Large")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravityEditCellViewOnly

```
layout gravityEditCellViewOnly boolean { t | nil }
```

Description

Setting this environment variable to `t` implies gravity will work within current cellview only. Seting it to `nil` implies gravity will work outside the current cellview also. This environment variable is useful in EIP mode, when the current cellview is not the same as the window cellview.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "gravityEditCellViewOnly")
envSetVal("layout" "gravityEditCellViewOnly" 'boolean t)
envSetVal("layout" "gravityEditCellViewOnly" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravityOn

```
layout gravityOn boolean { t | nil }
```

Description

Specifies whether gravity mode is on. Gravity mode causes the pointer to snap to objects in the cellview. The default is `nil`.

GUI Equivalent

Command	*Options – Editor*
Field	*Gravity On* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "gravityOn")
envSetVal("layout" "gravityOn" 'boolean t)
envSetVal("layout" "gravityOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravitySelectableOnly

```
layout gravitySelectableOnly boolean { t | nil }
```

Description

Sets the drawing pointer to snap only to the selected objects in the cellview when set to `t`. The default is `nil`.

GUI Equivalent

Command	*Options – Editor*
Field	*Selectable Objects Only* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "gravitySelectableOnly")
envSetVal("layout" "gravitySelectableOnly" 'boolean t)
envSetVal("layout" "gravitySelectableOnly" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravityType

```
layout gravityType string { pin | edge | midpoint | nexus | vertex | end | centerline
     | origin | snapPatternTrack }
```

Description

Sets the drawing pointer to snap to the specified target type. The default value is "`pin edge midpoint nexus vertex end centerline origin snapPatternTrack`".

GUI Equivalent

None

Examples

```
envGetVal("layout" "gravityType")
envSetVal("layout" "gravityType" 'string pin)
envSetVal("layout" "gravityType" 'string edge)
envSetVal("layout" "gravityType" 'string midpoint)
envSetVal("layout" "gravityType" 'string nexus)
envSetVal("layout" "gravityType" 'string vertex)
envSetVal("layout" "gravityType" 'string end)
envSetVal("layout" "gravityType" 'string centerline)
envSetVal("layout" "gravityType" 'string origin)
envSetVal("layout" "gravityType" 'string snapPatternTrack)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gravityUse5PixelApertureWhenOff

```
layout gravityUse5PixelApertureWhenOff boolean { t | nil }
```

Description

Setting this environment variable to `t` implies gravity remains active within a small aperture of five pixels even when it is turned off.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "gravityUse5PixelApertureWhenOff")
envSetVal("layout" "gravityUse5PixelApertureWhenOff" 'boolean t)
envSetVal("layout" "gravityUse5PixelApertureWhenOff" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

grEnclosedBy

`layout grEnclosedBy float` *`float_number`*

Description

Initializes the *Enclose by* field on the *Guard Ring* form. The default value is `0.0`. You can replace the default value with another integer point number by updating this environment variable in your `.cdsenv` file. You can change the value for the *Enclose by* field by typing in the field on the *Guard Ring* form. The software verifies that minSpacing is correct. It also verifies that it is minimum DRC compliant, based on process and context. If necessary, the software overrides the user value in the form.

GUI Equivalent

Command	*Create – MPP Guard Ring*
Field	*Enclose by* (Guard Ring Form)

Examples

```
envGetVal("layout" "grEnclosedBy")
envSetVal("layout" "grEnclosedBy" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gridMultiple

`layout gridMultiple int` *`any_positive_whole_number`*

`graphic gridMultiple int` *`any_positive_whole_number`*

Description

Number of minor grid lines or dots between major grid lines or dots. The default is `5`.

GUI Equivalent

Command	*Options – Display*
Field	*Major Spacing* (Display Options Form)

Examples

```
envGetVal("layout" "gridMultiple")
envSetVal("layout" "gridMultiple" 'int 7)
envGetVal("graphic" "gridMultiple")
envSetVal("graphic" "gridMultiple" 'int 7)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

gridSpacing

`layout gridSpacing float` *`float_number`*

`graphic gridSpacing float` *`float_number`*

Description

Number between each minor grid line or dot. The default is `1.0` user units.

GUI Equivalent

Command	*Options – Display*
Field	*Minor Spacing* (Display Options Form)

Examples

```
envGetVal("layout" "gridSpacing")
envSetVal("layout" "gridSpacing" 'float 2.0)
envGetVal("graphic" "gridSpacing")
envSetVal("graphic" "gridSpacing" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

highlightSelProtectedItems

```
graphic highlightSelProtectedItems boolean { t | nil }
```

Description

Determines whether protected items should be highlighted or not. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "highlightSelProtectedItems")
envSetVal("graphic" "highlightSelProtectedItems" 'boolean t)
envSetVal("graphic" "highlightSelProtectedItems" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

haloBottomOffset

`layout haloBottomOffset` *`any_floating_point_number`*

Description

Specifies the number of user units the placement halo is extended from the bottom of the instance bBox. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Instance – Define Halo* and *Create – P&R Objects – P&R Boundary – Define Halo*
Field	*Offsets: Bottom* (Halo Editor Form)

Examples

```
envGetVal("layout" "haloBottomOffset")
envSetVal("layout" "haloBottomOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

haloLeftOffset

`layout haloLeftOffset float` *`any_floating_point_number`*

Description

Specifies the number of user units the placement halo is extended from the left of the instance bBox. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Instance – Define Halo* and *Create – P&R Objects – P&R Boundary – Define Halo*
Field	*Offsets: Left* (Halo Editor Form)

Examples

```
envGetVal("layout" "haloLeftOffset")
envSetVal("layout" "haloLeftOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

haloRightOffset

`layout haloRightOffset float` *`any_floating_point_number`*

Description

Specifies the number of user units the placement halo is extended from the right of the instance bBox. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Instance – Define Halo* and *Create – P&R Objects – P&R Boundary – Define Halo*
Field	*Offsets: Right* (Halo Editor Form)

Examples

```
envGetVal("layout" "haloRightOffset")
envSetVal("layout" "haloRightOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

haloTopOffset

`layout haloTopOffset float` *`any_floating_point_number`*

Description

Specifies the number of user units the placement halo is extended from the top of the instance bBox. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Instance – Define Halo* and *Create – P&R Objects – P&R Boundary – Define Halo*
Field	*Offsets: Top* (Halo Editor Form)

Examples

```
envGetVal("layout" "haloTopOffset")
envSetVal("layout" "haloTopOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

highlightAlignedEdges

```
layout highlightAlignedEdges boolean { t | nil }
```

Description

Determines whether highlight aligned edges is set to on (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Dynamic Display*
Field	*Highlight Aligned Edges On* (Dynamic Display Form)

Examples

```
envGetVal("layout" "highlightAlignedEdges")
envSetVal("layout" "highlightAlignedEdges" 'boolean t)
envSetVal("layout" "highlightAlignedEdges" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

highlightAlignedEdgesColor

`layout highlightAlignedEdgesColor string` *`string_value`*

Description

Specifies the color of the highlight displayed on the edges of aligned shapes. The default value is `green`.

GUI Equivalent

Command	*Options – Dynamic Display – Highlight Aligned Edges*
Field	*Color* (Halo Editor Form)

Examples

```
envGetVal("layout" "highlightAlignedEdgesColor")
envSetVal("layout" "highlightAlignedEdgesColor" 'string "blue")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

highlightAlignedEdgesWidth

`layout highlightAlignedEdgesWidth int` *`any_integer_greater_than_zero`*

Description

Specifies width of the highlight that is displayed in the edges of the shapes aligned. The default value is `3`.

GUI Equivalent

Command	*Options – Dynamic Display – Highlight Aligned Edges*
Field	*Width* (Dynamic Display Form)

Examples

```
envGetVal("layout" "highlightAlignedEdgesWidth")
envSetVal("layout" "highlightAlignedEdgesWidth" 'int 4)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

iconsOn

```
layout iconsOn boolean { t | nil }
```

```
graphic iconsOn boolean { t | nil }
```

Description

Determines whether to display the elements in an un-expanded array as rectangles with the symbol for the master cell for each element. The variable applies to an instance or contact array. In the Display Options form set the display start and stop levels to 0. When iconsOn is `t`, selecting Show Name Of instance displays the instance name. When iconsOn is `t`, selecting Show Name Of master displays the master name.The layout default is `nil`, the graphic default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Array Icons* (Display Options Form)

Examples

```
envGetVal("layout" "iconsOn")
envSetVal("layout" "iconsOn" 'boolean t)
envSetVal("layout" "iconsOn" 'boolean nil)
envGetVal("graphic" "iconsOn")
envSetVal("graphic" "iconsOn" 'boolean t)
envSetVal("graphic" "iconsOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

ignoreBounce

```
layout ignoreBounce boolean { t | nil }
```

Description

Determines whether you use the gravity offset (bounce) values from the Layout Editor Options form (`nil`) or the `minSpacing` value from the design rules in the technology file (`t`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "ignoreBounce")
envSetVal("layout" "ignoreBounce" 'boolean t)
envSetVal("layout" "ignoreBounce" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

ignoreParallelMinWidthForSquarePinResize

```
layout ignoreParallelMinWidthForSquarePinResize boolean { t | nil }
```

Description

For square pins, determines whether to ignore the `minWidth` constraint along the direction perpendicular to the WSP track.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "ignoreParallelMinWidthForSquarePinResize")
envSetVal("layout" "ignoreParallelMinWidthForSquarePinResize" 'boolean t)
envSetVal("layout" "ignoreParallelMinWidthForSquarePinResize" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instanceDrawingMode

```
graphic instanceDrawingMode string { BBox | prBoundary | snapBoundary }
```

```
layout instanceDrawingMode string { BBox | prBoundary | snapBoundary }
```

Description

Determines whether the boundary of instances is displayed as the instance bounding box (`BBox`), the place and route boundary (`prBoundary`), or the snap boundary (`snapBoundary`). The default is `BBox`. This environment variable replaces the drawPRBoundary environment variable.

GUI Equivalent

Command	*Options – Display*
Field	*Instance Drawing Mode* (Display Options Form)

Examples

```
envGetVal("graphic" "instanceDrawingMode")
envSetVal("graphic" "instanceDrawingMode" 'string "BBox")
envSetVal("graphic" "instanceDrawingMode" 'string "prBoundary")
envSetVal("graphic" "instanceDrawingMode" 'string "snapBoundary")
envGetVal("layout" "instanceDrawingMode")
envSetVal("layout" "instanceDrawingMode" 'string "BBox")
envSetVal("layout" "instanceDrawingMode" 'string "prBoundary")
envSetVal("layout" "instanceDrawingMode" 'string "snapBoundary")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instanceSelectionMode

```
layout instanceSelectionMode cyclic { ignoreLPP | useLPP }
```

Description

Controls the selection of instances in the layout editor. You can set the environment variable to one of the following values:

- `ignoreLPP`: Ignores the *Selectability* of the LPP `instance drawing`.
- `useLPP`: The instance is selectable on the canvas if the LPP `instance drawing` has *Selectability* enabled in the Layers pane.

The default is `ignoreLPP`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "instanceSelectionMode")
envSetVal("layout" "instanceSelectionMode" 'cyclic "ignoreLPP")
envSetVal("layout" "instanceSelectionMode" 'cyclic "useLPP")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instanceVisibilityMode

```
layout instanceVisibilityMode cyclic { ignoreLPP | useLPP }
```

Description

Controls the visibility of instances in the layout editor. You can set the environment variable to one of the following values:

- `ignoreLPP`: Ignores the *Visibility* of the LPP `instance drawing`.
- `useLPP`: The instance is visible on the canvas if the LPP `instance drawing` has *Visibility* enabled in the *Layers* pane of the Palette Assistant.

The default is `useLPP`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "instanceVisibilityMode")
envSetVal("layout" "instanceVisibilityMode" 'cyclic "ignoreLPP")
envSetVal("layout" "instanceVisibilityMode" 'cyclic "useLPP")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instCellName

`layout instCellName string` *`any_cell_name`*

Description

Specifies the default name of the cell for `leHiCreateInst`. The default is none.

GUI Equivalent

Command	*Create – Instance*
Field	*Cell* (Create Instance Form)

Examples

```
envGetVal("layout" "instCellName")
envSetVal("layout" "instCellName" 'string "MyCell1")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instColumns

`layout instColumns int` *`any_integer_greater_than_zero`*

Description

Specifies the number of columns for an arrayed instance.The default is `1`.

GUI Equivalent

Command	*Create – Instance*
Field	*Mosaic: Columns* (Create Instance Form)

Examples

```
envGetVal("layout" "instColumns")
envSetVal("layout" "instColumns" 'int 4)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instDelX

`layout instDelX float` *`any_number_greater_than_zero`*

Description

Specifies the distance between the columns of an arrayed instance or contact measured from edge to edge. This field is grayed out if the number of columns specified is `1`. The default is `0`.

GUI Equivalent

Command	*Create – Instance*
Field	*Mosaic: Delta X* (Create Instance Form)

Examples

```
envGetVal("layout" "instDelX")
envSetVal("layout" "instDelX" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instDelY

`layout instDelY float` *`any_number_greater_than_zero`*

Description

Specifies the distance between the rows of an arrayed instance or contact measured from edge to edge. This field is grayed out if the number of columns specified is `1`. The default is `0`.

GUI Equivalent

Command	*Create – Instance*
Field	*Mosaic: Delta Y* (Create Instance Form)

Examples

```
envGetVal("layout" "instDelY")
envSetVal("layout" "instDelY" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instHalosEnabled

```
layout instHalosEnabled boolean { t | nil }
```

Description

Determines whether a halo is defined when creating an instance. The default is `t`.

GUI Equivalent

Command	*Create – Instance*
Field	*Halo* (Create Instance Form)

Examples

```
envGetVal("layout" "instHalosEnabled")
envSetVal("layout" "instHalosEnabled" 'boolean t)
envSetVal("layout" "instHalosEnabled" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instLabel

```
layout instLabel string { instance | master | both }
graphic instLabel string { instance | master | both }
```

Description

Determines whether the instance name, the master name, or both the instance and master names are to be displayed. The default is `master`.

GUI Equivalent

Command	*Options – Display*
Field	*Show Name Of* (Display Options Form)

Examples

```
envGetVal("layout" "instLabel")
envSetVal("layout" "instLabel" 'string "instance")
envSetVal("layout" "instLabel" 'string "master")
envSetVal("layout" "instLabel" 'string "both")
envGetVal("graphic" "instLabel")
envSetVal("graphic" "instLabel" 'string "instance")
envSetVal("graphic" "instLabel" 'string "master")
envSetVal("graphic" "instLabel" 'string "both")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instLibName

`layout instLibName string` *`any_library_name`*

Description

Specifies the default name of the library for `leHiCreateInst`. The default is none.

GUI Equivalent

Command	*Create – Instance*
Field	*Library* (Create Instance Form)

Examples

```
envGetVal("layout" "instLibName")
envSetVal("layout" "instLibName" 'string "MyLib")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instMosaicCreation

```
layout instMosaicCreation boolean { t | nil }
```

Description

Creates an instance array. The default is `nil`.

GUI Equivalent

Command	*Create – Instance*
Field	*Create as Mosaic* (Create Instance Form)

Examples

```
envGetVal("layout" "instMosaicCreation")
envSetVal("layout" "instMosaicCreation" 'boolean t)
envSetVal("layout" "instMosaicCreation" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instName

`layout instName string` *`any_cell_name`*

Description

Specifies one or more names for the instances you create. You can specify as many names as you want (separated by spaces). As each name is used, it is removed from the list. If all names are used, the function automatically generates a name based on the current master's `instNamePrefix` property and puts it in the form field. The default is none.

GUI Equivalent

Command	*Create – Instance*
Field	*Names* (Create Instance Form)

Examples

```
envGetVal("layout" "instName")
envSetVal("layout" "instName" 'string "Inst1")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instPhysicalOnly

```
layout instPhysicalOnly boolean { t | nil }
```

Description

Creates an instance in such a way such that it exists only in the block domain and not in the module domain. The default is `nil`.

GUI Equivalent

Command	*Create – Instance*
Field	*Physical Only* (Create Instance Form)

Examples

```
envGetVal("layout" "instPhysicalOnly")
envSetVal("layout" "instPhysicalOnly" 'boolean t)
envSetVal("layout" "instPhysicalOnly" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instRows

`layout instRows int` *`any_integer_greater_than_zero`*

Description

Specifies the number of rows for an arrayed instance. The default is `1`.

GUI Equivalent

Command	*Create – Instance*
Field	*Mosaic: Rows* (Create Instance Form)

Examples

```
envGetVal("layout" "instRows")
envSetVal("layout" "instRows" 'int 7)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

instViewName

```
layout instViewName string { layout | schematic | symbolic }
```

Description

Specifies the name of the view for `leHiCreateInst`. The default is `layout`.

GUI Equivalent

Command	*Create – Instance*
Field	*View* (Create Instance Form)

Examples

```
envGetVal("layout" "instViewName")
envSetVal("layout" "instViewName" 'string "layout")
envSetVal("layout" "instViewName" 'string "schematic")
envSetVal("layout" "instViewName" 'string "symbolic")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

keepCopiedInsideGroup

```
layout keepCopiedInsideGroup boolean { t | nil }
```

Description

When the `transparentGroup` environment variable is set to `t`, the copied figure group members of different groups are added to their respective source figure groups. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "keepCopiedInsideGroup")
envSetVal("layout" "keepCopiedInsideGroup" 'boolean t)
envSetVal("layout" "keepCopiedInsideGroup" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

keepFirstTermName

```
layout keepFirstTermName boolean { t | nil }
```

Description

Specifies if the first terminal name is not removed on the Create Pin form when the pin is created. The default is `nil`.

GUI Equivalent

Command	*Create – Pin*
Field	*Keep First Name* (Create Pin Form)

Examples

```
envGetVal("layout" "keepFirstTermName")
envSetVal("layout" "keepFirstTermName" 'boolean t)
envSetVal("layout" "keepFirstTermName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

keepGuardRingEndsConnected

```
layout keepGuardRingEndsConnected boolean { t | nil }
```

Description

Controls whether a guard ring with touching ends will stay connected (`t`) or will result in opening up the guard ring (`nil`) during the *Stretch* command. The default is `nil`.

GUI Equivalent

Command	*Edit – Stretch*
Field	*Keep guard ring ends connected* (Stretch Form)

Examples

```
envGetVal("layout" "keepGuardRingEndsConnected")
envSetVal("layout" "keepGuardRingEndsConnected" 'boolean t)
envSetVal("layout" "keepGuardRingEndsConnected" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

keepWiresConnectedToNonWires

```
layout keepWiresConnectedToNonWires boolean { t | nil }
```

Description

Specifies if the connection with shapes, such as pin figures, rectangles, polygons, and MPP, coincident with the selected vertex during the stretch operation should be maintained. The default is `nil`.

GUI Equivalent

Command	*Edit – Stretch*
Field	*Keep wires connected to shapes* (Stretch Form)

Examples

```
envGetVal("layout" "keepWiresConnectedToNonWires")
envSetVal("layout" "keepWiresConnectedToNonWires" 'boolean t)
envSetVal("layout" "keepWiresConnectedToNonWires" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelAttach

```
layout labelAttach boolean { t | nil }
```

Description

Specifies whether the system prompts you to attach labels to an object. The default is `t`.

GUI Equivalent

Command	*Create – Label*
Field	*Attach* (Create Label Form)

Examples

```
envGetVal("layout" "labelAttach")
envSetVal("layout" "labelAttach" 'boolean t)
envSetVal("layout" "labelAttach" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelAutoConn

```
layout labelAutoConn boolean { t | nil }
```

Description

Specifies whether the method for creating labels is automatic using the connectivity information of objects to be labeled.

GUI Equivalent

Command	*Create – Label*
Field	*Mode – Auto – From Interconnect* (Create Label Form)

Examples

```
envGetVal("layout" "labelAutoConn")
envSetVal("layout" "labelAutoConn" 'boolean t)
envSetVal("layout" "labelAutoConn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelAutoInst

```
layout labelAutoInst boolean { t | nil }
```

Description

Specifies whether the method for creating labels is automatic using instances.

GUI Equivalent

Command	*Create – Label*
Field	*Mode – Auto – From Instance* (Create Label Form)

Examples

```
envGetVal("layout" "labelAutoInst")
envSetVal("layout" "labelAutoInst" 'boolean t)
envSetVal("layout" "labelAutoInst" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelDrafting

```
layout labelDrafting boolean { t | nil }
```

Description

Specifies whether labels are restricted to rotate no more than 90 degrees.The default is `t`.

GUI Equivalent

Command	*Create – Label*
Field	*Drafting* (Create Label Form)

Examples

```
envGetVal("layout" "labelDrafting")
envSetVal("layout" "labelDrafting" 'boolean t)
envSetVal("layout" "labelDrafting" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelExpand

```
layout labelExpand boolean { t | nil }
```

Description

Specifies whether or not the specified label string or pattern is to be expanded to separate label names in the sequence. In `manual` mode, the default is `t`.

GUI Equivalent

Command	*Create – Label*
Field	*Expand* (Create Label Form)

Examples

```
envGetVal("layout" "labelExpand")
envSetVal("layout" "labelExpand" 'boolean t)
envSetVal("layout" "labelExpand" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelFontStyle

```
layout labelFontStyle cyclic { stick | gothic | roman | math | euroStyle | swedish
     | script | fixed | milspec }
```

Description

Specifies the default font of labels. The default is `stick`.

GUI Equivalent

Command	*Create – Label*
Field	*Font* (Create Label Form)

Examples

```
envGetVal("layout" "labelFontStyle")
envSetVal("layout" "labelFontStyle" 'cyclic "stick")
envSetVal("layout" "labelFontStyle" 'cyclic "gothic")
envSetVal("layout" "labelFontStyle" 'cyclic "roman")
envSetVal("layout" "labelFontStyle" 'cyclic "math")
envSetVal("layout" "labelFontStyle" 'cyclic "euroStyle")
envSetVal("layout" "labelFontStyle" 'cyclic "swedish")
envSetVal("layout" "labelFontStyle" 'cyclic "script")
envSetVal("layout" "labelFontStyle" 'cyclic "fixed")
envSetVal("layout" "labelFontStyle" 'cyclic "milspec")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelHeight

`layout labelHeight float` *`any_number_greater_than_zero`*

Description

Specifies the height of labels in user units. The default is `1.0` user units.

GUI Equivalent

Command	*Create – Label*
Field	*Height* (Create Label Form)

Examples

```
envGetVal("layout" "labelHeight")
envSetVal("layout" "labelHeight" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelJustify

```
layout labelJustify cyclic { lowerLeft | centerLeft | upperLeft | lowerCenter |
     centerCenter | upperCenter | lowerRight | centerRight | upperRight }
```

Description

Specifies the default justification of a label with respect to its origin. The default is `centerCenter`.

GUI Equivalent

Command	*Create – Label*
Field	*Justification* (Create Label Form)

Examples

```
envGetVal("layout" "labelJustify")
envSetVal("layout" "labelJustify" 'cyclic "lowerLeft")
envSetVal("layout" "labelJustify" 'cyclic "centerLeft")
envSetVal("layout" "labelJustify" 'cyclic "upperLeft")
envSetVal("layout" "labelJustify" 'cyclic "lowerCenter")
envSetVal("layout" "labelJustify" 'cyclic "centerCenter")
envSetVal("layout" "labelJustify" 'cyclic "upperCenter")
envSetVal("layout" "labelJustify" 'cyclic "lowerRight")
envSetVal("layout" "labelJustify" 'cyclic "centerRight")
envSetVal("layout" "labelJustify" 'cyclic "upperRight")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelKeepLabel

```
layout labelKeepLabel boolean { t | nil }
```

Description

Specifies whether or not the specified label string or pattern should be retained in the *Label (Pattern)* field after all the generated labels have been placed. In `manual` mode, the default is `t`.

GUI Equivalent

Command	*Create – Label*
Field	*Keep Label* (Create Label Form)

Examples

```
envGetVal("layout" "labelKeepLabel")
envSetVal("layout" "labelKeepLabel" 'boolean t)
envSetVal("layout" "labelKeepLabel" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelLPP

```
layout labelLPP cyclic { auto | currentLayer | sameLayer | selectLayer }
```

Description

Specifies the layer/purpose to use for creating labels. The default is `currentLayer`. This determines the layer to place the label from the `stampLabelLayers` section of the technology file.

`currentLayer` uses the layer/purpose set as the current entry layer in the *Layers* assistant.

`sameLayer` uses the layer of the object to be labeled.

`selectLayer` allows you to specify the layer/purpose from the drop-down list in the *Create Label* form.

GUI Equivalent

Command	*Create – Label*
Field	*Label Layer/Purpose* (Create Label Form)

Examples

```
envGetVal("layout" "labelLPP")
envSetVal("layout" "labelLPP" 'cyclic "auto")
envSetVal("layout" "labelLPP" 'cyclic "currentLayer")
envSetVal("layout" "labelLPP" 'cyclic "sameLayer")
envSetVal("layout" "labelLPP" 'cyclic "selectLayer")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelMode

```
layout labelMode cyclic { manual | auto }
```

Description

Determines the method for creating labels - manually, using the specified label text or pattern or automatically, using connectivity information or instances or both. The default is `manual`.

GUI Equivalent

Command	*Create – Label*
Field	*Mode* (Create Label Form)

Examples

```
envGetVal("layout" "labelMode")
envSetVal("layout" "labelMode" 'cyclic "manual")
envSetVal("layout" "labelMode" 'cyclic "auto")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelOverbar

```
layout labelOverbar boolean { t | nil }
```

Description

Determines how text strings containing underscore characters are displayed. The default is `nil`.

GUI Equivalent

Command	*Create – Label*
Field	*Overbar* (Create Label Form)

Examples

```
envGetVal("layout" "labelOverbar")
envSetVal("layout" "labelOverbar" 'boolean t)
envSetVal("layout" "labelOverbar" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelPurpose

`layout labelPurpose string` *`any_valid_purpose_name`*

Description

Enables you to fix the purpose for labels in the *Use same layer as shape, select purpose* field of the *Create Label* form. If you are trying to fix an invalid value for the purpose, then the previously set purpose value will be retained. The default is `drawing`.

GUI Equivalent

Command	*Create – Label*
Field	*Use same layer as shape, select purpose* (Create Label Form)

Examples

```
envGetVal("layout" "labelPurpose")
envSetVal("layout" "labelPurpose" 'string "drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

labelText

`layout labelText string` *`any_string_or_pattern`*

Description

Specifies the label text or the pattern to use for generating label names in *Manual* mode. If no text is specified, the command does not proceed and prompts you to specify a label text.

GUI Equivalent

Command	*Create – Label*
Field	*Label (Pattern)* (Create Label Form)

Examples

```
envGetVal("layout" "labelPurpose")
envSetVal("layout" "labelPurpose" 'string "A<0:3> A<<0:3>> A<1:4>-B<0:1>
A<0:6:2>")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

leCheckMissingLayers

```
graphic leCheckMissingLayers boolean { t | nil }
```

Description

Determines if checking is done to identify layers not defined in the technology file. When set to `t`, checking is done on maskLayout cellviews only. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "leCheckMissingLayers")
envSetVal("graphic" "leCheckMissingLayers" 'boolean t)
envSetVal("graphic" "leCheckMissingLayers" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

leCheckMissingMasters

```
graphic leCheckMissingMasters boolean { t | nil }
```

Description

Determines if unbound masters will be checked while opening a design. When set to `t`, or if the cellview is not maskLayout, the check is done. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "leCheckMissingMasters")
envSetVal("graphic" "leCheckMissingMasters" 'boolean t)
envSetVal("graphic" "leCheckMissingMasters" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

leChopPreservePathSeg

```
layout leChopPreservePathSeg boolean { t | nil }
```

Description

Determines if pathsegs are preserved from being chopped. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "leChopPreservePathSeg")
envSetVal("layout" "leChopPreservePathSeg" 'boolean t)
envSetVal("layout" "leChopPreservePathSeg" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

leDispNetExpr

```
layout leDispNetExpr boolean { t | nil }
graphic leDispNetExpr boolean { t | nil }
```

Description

Determines if the net expression of a terminal is displayed in the text label of pins. The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Net Expressions* (Display Options Form)

Examples

```
envGetVal("layout" "leDispNetExpr")
envSetVal("layout" "leDispNetExpr" 'boolean t)
envSetVal("layout" "leDispNetExpr" 'boolean nil)
envGetVal("graphic" "leDispNetExpr")
envSetVal("graphic" "leDispNetExpr" 'boolean t)
envSetVal("graphic" "leDispNetExpr" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

leWindowBBox

`layout leWindowBBox string` *`any_valid_screen_coordinates`*

Description

Specifies the default location and size of a layout window. The default is `((120 185) (1020 875))`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "leWindowBBox")
envSetVal("layout" "leWindowBBox" 'string "xyz")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

lockAngles

```
layout lockAngles boolean { t | nil }
```

Description

Specifies that angles of partially selected shapes are not modified when the shapes are stretched. This option has no effect on instances or arrays. The default is `t`.

GUI Equivalent

Command	*Edit – Stretch*
Field	*Lock angles* (Stretch Form)

Examples

```
envGetVal("layout" "lockAngles")
envSetVal("layout" "lockAngles" 'boolean t)
envSetVal("layout" "lockAngles" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

lppVisibilityMode

```
layout lppVisibilityMode cyclic { checkValidity | donotCheckValidity }
```

Description

Controls the visibility of a layer purpose pair (LPP) based on its visibility and validity. When set to `checkValidity`, only the shapes on valid and visible LPPs are displayed in the design window. When set to `donotCheckValidity`, shapes on all visible LPPs are displayed in the design window, irrespective of the validity of the LPPs. The default is `donotCheckValidity`.

GUI Equivalent

Command	*Options – Display*
Field	*Set LPP Visibility Mode* (Display Options Form)

Examples

```
envGetVal("layout" "lppVisibilityMode")
envSetVal("layout" "lppVisibilityMode" 'cyclic "checkValidity")
envSetVal("layout" "lppVisibilityMode" 'cyclic "donotCheckValidity")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magAnchorPosition

```
layout magAnchorPosition cyclic { top left corner | top right corner | bottom left
     corner | bottom right corner }
```

Description

Specifies where to position the magnifier with respect to the pointer position. By default, the magnifier appears at the *top left corner* of the design display area with respect to the pointer.

GUI Equivalent

Command	*Options – Magnifier*
Field	*Anchored at* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magAnchorPosition")
envSetVal("layout" "magAnchorPosition" 'cyclic "top left corner")
envSetVal("layout" "magAnchorPosition" 'cyclic "top right corner")
envSetVal("layout" "magAnchorPosition" 'cyclic "bottom left corner")
envSetVal("layout" "magAnchorPosition" 'cyclic "bottom right corner")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magConfined

```
layout magConfined boolean { t | nil }
```

Description

Keeps the magnifier within the design display area and always displays it completely. The default is `t`. If set to `nil`, the magnifier might not be visible completely if it is close to the edge of the design display canvas.

GUI Equivalent

Command	*Options – Magnifier*
Field	*Keep Within Window* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magConfined")
envSetVal("layout" "magConfined" 'boolean t)
envSetVal("layout" "magConfined" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magCursorOffset

`layout magCursorOffset int` *`integer_number`*

Description

Specifies the distance of the magnifier from the pointer position. You can specify any value from `0` through `100`. The default value is `15`.

GUI Equivalent

Command	*Options – Magnifier*
Field	*offset by* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magCursorOffset")
envSetVal("layout" "magCursorOffset" 'int 35)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magDisplayMode

```
layout magDisplayMode cyclic { Anchored at Cursor Position | Anchored at | Not
     Anchored }
```

Description

Specifies the type of anchoring to use for the magnifier. The default setting is *Not Anchored*.

GUI Equivalent

Command	*Options – Magnifier*
Field	*Anchored at Cursor Position, Anchored at, Not Anchored* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magDisplayMode")
envSetVal("layout" "magDisplayMode" 'cyclic "Anchored at Cursor Position")
envSetVal("layout" "magDisplayMode" 'cyclic "Anchored at")
envSetVal("layout" "magDisplayMode" 'cyclic "Not Anchored")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magFloatPosition

```
layout magFloatPosition cyclic { Center | Position Magnfier }
```

Description

Enables you to customize the position of the magnifier. If you set the value to *Center*, the center of the magnifier coincides the pointer position. If you set the value to *Position Magnifier*, you can specify the magCursorOffset after setting the magOffsetCorner. The default setting is *Position Magnifier*.

GUI Equivalent

Command	*Options – Magnifier*
Field	*Not Anchored: Center, Position Magnifier* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magFloatPosition")
envSetVal("layout" "magFloatPosition" 'cyclic "Center")
envSetVal("layout" "magFloatPosition" 'cyclic "Position Magnfier")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magOffsetCorner

```
layout magOffsetCorner cyclic { top left | top right | bottom right | bottom left }
```

Description

Specifies the magnifier position with respect to the pointer position. The default position is *top left*.

GUI Equivalent

Command	*Options – Magnifier*
Field	*Position Magnifier* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magOffsetCorner")
envSetVal("layout" "magOffsetCorner" 'cyclic "top left")
envSetVal("layout" "magOffsetCorner" 'cyclic "top right")
envSetVal("layout" "magOffsetCorner" 'cyclic "bottom right")
envSetVal("layout" "magOffsetCorner" 'cyclic "bottom left")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magScale

`layout magScale float` *`float_number`*

Description

Changes the zooming factor of the magnifier. You can set a value from `1` through `1000`. The default value is `4.0`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "magScale")
envSetVal("layout" "magScale" 'float 10.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

magSize

`layout magSize int` *`integer_number`*

Description

Specifies the size of the magnifier. You can set any value from `50` through `250`. The default value is `200`.

GUI Equivalent

Command	*Options – Magnifier*
Field	*Magnifier Size* (Magnifier Options Form)

Examples

```
envGetVal("layout" "magSize")
envSetVal("layout" "magSize" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

maintainConnections

```
layout maintainConnections boolean { t | nil }
```

Description

Determines whether automatic re-connection mode for pin-path and pin-pathseg connections is active. The default is `nil`.

GUI Equivalent

Command	*Options – Editor*
Field	*Maintain Connections* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "maintainConnections")
envSetVal("layout" "maintainConnections" 'boolean t)
envSetVal("layout" "maintainConnections" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

makeCellCellType

```
layout makeCellCellType cyclic { none | block | blockRing | cover | coverBump | pad
     | padSpacer | core | coreSpacer | coreAntenna | corner | softMacro | via |
     blockBlackBox | padAreaIO }
```

Description

Specifies the type of cell you want to create. The default is `none`.

GUI Equivalent

Command	*Edit – Hierarchy – Make Cell*
Field	*Type* (Make Cell Form)

Examples

```
envGetVal("layout" "makeCellCellType")
envSetVal("layout" "makeCellCellType" 'cyclic "none")
envSetVal("layout" "makeCellCellType" 'cyclic "block")
envSetVal("layout" "makeCellCellType" 'cyclic "blockRing")
envSetVal("layout" "makeCellCellType" 'cyclic "cover")
envSetVal("layout" "makeCellCellType" 'cyclic "coverBump")
envSetVal("layout" "makeCellCellType" 'cyclic "pad")
envSetVal("layout" "makeCellCellType" 'cyclic "padSpacer")
envSetVal("layout" "makeCellCellType" 'cyclic "core")
envSetVal("layout" "makeCellCellType" 'cyclic "coreSpacer")
envSetVal("layout" "makeCellCellType" 'cyclic "coreAntenna")
envSetVal("layout" "makeCellCellType" 'cyclic "corner")
envSetVal("layout" "makeCellCellType" 'cyclic "softMacro")
envSetVal("layout" "makeCellCellType" 'cyclic "via")
envSetVal("layout" "makeCellCellType" 'cyclic "blockBlackBox")
envSetVal("layout" "makeCellCellType" 'cyclic "padAreaIO")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerAccessMode

```
graphic markerAccessMode cyclic { read | write }
```

Description

Specifies the mode, read or write, for opening a cellview. The default is `write`.

GUI Equivalent

Command	*Verify – Markers – Find*
Form Field	*Cellview Access Mode* (Find Marker Form)

Examples

```
envGetVal("graphic" "markerAccessMode")
envSetVal("graphic" "markerAccessMode" 'cyclic "read")
envSetVal("graphic" "markerAccessMode" 'cyclic "write")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerDeleteAccessModeIsEdit

```
graphic markerDeleteAccessModeIsEdit boolean { t | nil }
```

Description

Enables you to specify whether you want to open the cellview in read or edit mode when deleting the markers. The default setting is `t`. In the default mode, the cellview is re-opened in edit mode for deleting markers. In this mode, you can additionally set markerDeleteConfirmModeChange to `t` to get a confirmation dialog box. If `markerDeleteAccessModeIsEdit` is `nil`, markers are deleted and the cellview remains in read mode.

GUI Equivalent

Command	*Verify – Markers – Delete*, *Verify – Markers – Delete All*
Field	*Cellview Access Mode* (Delete All Markers Form, Delete Marker Form)

Examples

```
envGetVal("graphic" "markerDeleteAccessModeIsEdit")
envSetVal("graphic" "markerDeleteAccessModeIsEdit" 'boolean t)
envSetVal("graphic" "markerDeleteAccessModeIsEdit" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerDeleteConfirmModeChange

```
graphic markerDeleteConfirmModeChange boolean { t | nil }
```

Description

Opens a dialog box that enables you to confirm whether or not the cellview(s) are to be re-opened in edit mode when you try to delete markers on the cellview(s) (*Verify – Markers – Delete All*). By default, the environment variable is set to `nil`. In this mode, markers are deleted and the cellviews are re-opened in edit mode with no dialog box. If set to `t`, a dialog box appears where you can specify all, none, or some cellviews to reopen in edit mode to delete markers on them. In this mode, markers are not deleted in cellviews that stay in read mode.

This environment variable has no effect if markerDeleteAccessModeIsEdit is set to `nil`.

GUI Equivalent

Command	*Verify – Markers – Delete*, *Verify – Markers – Delete All*
Field	*Confirm Access Mode Change* (Delete All Markers Form, Delete Marker Form)

Examples

```
envGetVal("graphic" "markerDeleteConfirmModeChange")
envSetVal("graphic" "markerDeleteConfirmModeChange" 'boolean t)
envSetVal("graphic" "markerDeleteConfirmModeChange" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerDeleteScope

```
graphic markerDeleteScope cyclic { cellview | hierarchy starting from top cellview
     | hierarchy starting from current cellview }
```

Description

Defines the scope of the deletion of markers. You can set the scope for deletion of markers from the current cellview, throughout the hierarchy, or throughout the hierarchy starting with the current cellview. By default, the environment variable is set to `cellview`.

GUI Equivalent

Command	*Verify – Markers – Delete All*
Field	*Search Scope* (Delete All Markers Form)

Examples

```
envGetVal("graphic" "markerDeleteScope")
envSetVal("graphic" "markerDeleteScope" 'cyclic "cellview")
envSetVal("graphic" "markerDeleteScope" 'cyclic "hierarchy starting from top
cellview")
envSetVal("graphic" "markerDeleteScope" 'cyclic "hierarchy starting from current
cellview")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerDeleteSeverities

```
graphic markerDeleteSeverities string { info | acknowledgedWarning | warning |
     signedOffError | error | signedOffCriticalError | criticalError | fatalError }
```

Description

Defines the default severities. By default, the environment variable is set to `info`, `acknowledgedWarning`, `warning`, `signedOffError`, `error`, `signedOffCriticalError`, `criticalError`, and `fatalError` severities.

GUI Equivalent

Command	*Verify – Markers – Delete All*
Field	*Severity* (Delete All Markers Form)

Examples

```
envGetVal("graphic" "markerDeleteSeverities")
envSetVal("graphic" "markerDeleteSeverities" 'string "info")
envSetVal("graphic" "markerDeleteSeverities" 'string "acknowledgedWarning")
envSetVal("graphic" "markerDeleteSeverities" 'string "warning")
envSetVal("graphic" "markerDeleteSeverities" 'string "signedOffError")
envSetVal("graphic" "markerDeleteSeverities" 'string "error")
envSetVal("graphic" "markerDeleteSeverities" 'string "signedOffCriticalError")
envSetVal("graphic" "markerDeleteSeverities" 'string "criticalError")
envSetVal("graphic" "markerDeleteSeverities" 'string "fatalError")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerDeleteSources

`graphic markerDeleteSources string { VLS Connectivity:nil | Default:t }`

Description

Defines the state of the sources. The string to be specified in the environment variable is tool name followed by its state `t` or `nil`. The tool name and state are seperated by a colon (:). Multiple tooks are separated by a comma. The `Default` key can be used to set the state of any other tool by default.

GUI Equivalent

Command	*Verify – Markers – Delete All*
Field	*Sources: select* (Delete All Markers Form)

Examples

```
envGetVal("graphic" "markerDeleteSources")
envSetVal("graphic" "markerDeleteSources" 'string "VLS Connectivity:t")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerDeleteSourcesMode

```
graphic markerDeleteSourcesMode cyclic { all | select }
```

Description

Defines the source mode for deletion of markers. By default, the environment variable is set to `all`. The value `all`signifies all markers that match the severities and scope criteria will be deleted. The value `select` signifies that a list of marker owners will be computed and only the sources that have been selected will be deleted.

GUI Equivalent

Command	*Verify – Markers – Delete All*
Field	*Sources* (Delete All Markers Form)

Examples

```
envGetVal("graphic" "markerDeleteSourcesMode")
envSetVal("graphic" "markerDeleteSourcesMode" 'cyclic "all")
envSetVal("graphic" "markerDeleteSourcesMode" 'cyclic "select")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerSearchBy

```
graphic markerSearchBy cyclic { master | occurrences }
```

Description

Specifies which markers will display, markers in the master cell only (`master`) or all occurrences of markers (`occurences`). The default is `occurrences`.

GUI Equivalent

Command	*Verify – Markers – Find*
Form Field	*Find by* (Find Marker Form)

Examples

```
envGetVal("graphic" "markerSearchBy")
envSetVal("graphic" "markerSearchBy" 'cyclic "master")
envSetVal("graphic" "markerSearchBy" 'cyclic "occurrences")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerSearchScope

```
graphic markerSearchScope cyclic { cellview | hierarchy starting from top cellview
     | hierarchy starting from current cellview | library }
```

Description

Specifies what part of the design hierarchy you want to search.The default is `cellview`.

GUI Equivalent

Command	*Verify – Markers – Find*
Form Field	*Search Scope* (Find Marker Form)

Examples

```
envGetVal("graphic" "markerSearchScope")
envSetVal("graphic" "markerSearchScope" 'cyclic "cellview")
envSetVal("graphic" "markerSearchScope" 'cyclic "hierarchy starting from top
cellview")
envSetVal("graphic" "markerSearchScope" 'cyclic "hierarchy starting from current
cellview")
envSetVal("graphic" "markerSearchScope" 'cyclic "library")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerSeverity

```
graphic markerSeverity string { fatalError | criticalError | signedOffCriticalError
     | error | signedOffError | warning | acknowledgedWarning | info | annotation }
```

Description

Specifies the types of warnings and errors to be searched for and highlighted in a design. Replace *list* with one or more of the following values, separated by white space: `fatalError`, `criticalError`, `signedOffCriticalError`, `error`, `signedOffError`, `warning`, `acknowledgedWarning`, `info`, `annotation`. Set `markerSeverity` to `all` or `both` to display all types of markers. Set the environment variable to `none` to switch off marker display.

The default is `none`.

GUI Equivalent

Command	*Verify – Markers – Find*
Field	*Severity* (Find Marker Form)

Examples

```
envSetVal("graphic" "markerSeverity" 'string "info fatalError warning")
```

Searches for and displays markers of type `info`, `fatalError`, and `warning`.

```
envGetVal("graphic" "markerSeverity")
envSetVal("graphic" "markerSeverity" 'string "fatalError")
envSetVal("graphic" "markerSeverity" 'string "criticalError")
envSetVal("graphic" "markerSeverity" 'string "signedOffCriticalError")
envSetVal("graphic" "markerSeverity" 'string "error")
envSetVal("graphic" "markerSeverity" 'string "signedOffError")
envSetVal("graphic" "markerSeverity" 'string "warning")
envSetVal("graphic" "markerSeverity" 'string "acknowledgedWarning")
envSetVal("graphic" "markerSeverity" 'string "info")
envSetVal("graphic" "markerSeverity" 'string "annotation")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerZoom

```
graphic markerZoom boolean { t | nil }
```

Description

Determines whether to zoom to the next or previous marker while using the schematic editor's Find Marker form. Does not apply to the layout editor. The default is `nil`.

GUI Equivalent

Command	*Verify – Markers – Find*
Field	*Zoom To Markers* (Find Marker Form)

Examples

```
envGetVal("graphic" "markerZoom")
envSetVal("graphic" "markerZoom" 'boolean t)
envSetVal("graphic" "markerZoom" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markerZoomScale

`layout markerZoomScale int` *`any_positive_integer`*

`graphic markerZoomScale int` *`any_positive_integer`*

Description

Determines how close to the selected objects to zoom. The layout default is `1`, the graphic default is `4`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "markerZoomScale")
envSetVal("layout" "markerZoomScale" 'int 2)
envGetVal("graphic" "markerZoomScale")
envSetVal("graphic" "markerZoomScale" 'int 3)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetAutoColorCycle

```
layout markNetAutoColorCycle boolean { t | nil }
```

Description

Determines whether Mark Net marks the nets by cycling automatically through layers y0 through y9 (`t`) using the colors and styles of those layers, or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Auto Highlight Color* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetAutoColorCycle")
envSetVal("layout" "markNetAutoColorCycle" 'boolean t)
envSetVal("layout" "markNetAutoColorCycle" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetConnectedLayer

```
layout markNetConnectedLayer boolean { t | nil }
```

Description

Determines whether the settings defined for the connected layers (on the *Connected Layers* tab) are used for tracing a net. The default is `t`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Customized Connected Layers* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetConnectedLayer")
envSetVal("layout" "markNetConnectedLayer" 'boolean t)
envSetVal("layout" "markNetConnectedLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetExcludePurposes

```
layout markNetExcludePurposes boolean { t | nil }
```

Description

Determines whether to exclude the specified purposes from tracing. The default purpose values that are listed are *annotate*, *boundary*, *error*, *flight*, *label*, and *warning*. You can delete any of the default purpose values. The default is `t`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Exclude Purposes* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetExcludePurposes")
envSetVal("layout" "markNetExcludePurposes" 'boolean t)
envSetVal("layout" "markNetExcludePurposes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetHilightObj

```
layout markNetHilightObj boolean { t | nil }
```

Description

Sets the Mark Net command to display the traced connectivity after every 10,000 shapes are processed. The default is `nil`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Highlight Incrementally* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetHilightObj")
envSetVal("layout" "markNetHilightObj" 'boolean t)
envSetVal("layout" "markNetHilightObj" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetNumThreads

`layout markNetNumThreads int` *`integer_number`*

Description

Controls the chaser thread mode of the Mark Net command. If you set this environment variable to an integer less than or equal to `0`, the Mark Net command runs in auto mode. If you set the value to `1`, the Mark Net command runs in single thread mode. If you set the value to an interger greater than `1`, the Mark Net command runs in custom thread mode.

The default value of this environment variable is `0`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "markNetNumThreads")
envSetVal("layout" "markNetNumThreads" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetOptionsFile

```
layout markNetOptionsFile string <dir-path/filename>
```

Description

Specifies the path to the `markNetOptions` file to be read when a new Virtuoso session is started. You can specify the absolute file path or only the filename. The file path can be specified using Unix shell environment variables such as $HOME and $PWD; for example, `$HOME/myMarkNetOptions` or `$PWD/myMarkNetOptions`.

If you specify only the filename, the file is looked up using the Cadence Search File (CSF) mechanism.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Save/Load custom settings for Via, Connected and Stop Layers* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetOptionsFile")
envSetVal("layout" "markNetOptionsFile" 'string "markNetOptions")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetRange

```
layout markNetRange cyclic { Top to bottom | Current to bottom | Top to current
     |Top to stop level | Current to stop level | User defined }
```

Description

Sets the hierarchical range that the Mark Net command uses to determine which nets will highlight and which will not. `Top to bottom` makes available all nets in the entire hierarchy of the cellview for marking. `Current to bottom` makes available the nets in the hierarchy from the current level of hierarchy to the Display Level To field in the Display Options form (stop level) for marking. `Top to current` makes available the nets in the hierarchy from the top level to the current level of hierarchy for marking. `Top to stop level` makes available the nets in the hierarchy from the top level to the level defined as the stop level. `Current to stop level` makes available the nets in the hierarchy from the current level to the level defined as the stop level. `User defined` makes available the nets in a numerical hierarchical range for marking in conjunction with `markNetStartLevel` and `markNetStopLevel`. The default is `Top to bottom`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Hierarchy Range* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetRange")
envSetVal("layout" "markNetRange" 'cyclic "Top to bottom")
envSetVal("layout" "markNetRange" 'cyclic "Current to bottom")
envSetVal("layout" "markNetRange" 'cyclic "Top to current")
envSetVal("layout" "markNetRange" 'cyclic "Top to stop level")
envSetVal("layout" "markNetRange" 'cyclic "Current to stop level")
envSetVal("layout" "markNetRange" 'cyclic "User defined")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetSaveViaInfo

```
layout markNetSaveViaInfo boolean { t | nil }
```

Description

Enables via shapes to be saved in the cellview created by running the *Connectivity – Nets – Save All Mark Nets* command. The default is `nil`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Retain Via Info for Save MarkNets* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetSaveViaInfo")
envSetVal("layout" "markNetSaveViaInfo" 'boolean t)
envSetVal("layout" "markNetSaveViaInfo" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetSelectOverlapLayer

```
layout markNetSelectOverlapLayer boolean { t | nil }
```

Description

Enables you to select the LPP from where the *Mark Net* command starts tracing the net when multiple overlapping shapes are selected. The default is `nil`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Select From Overlaps* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetSelectOverlapLayer")
envSetVal("layout" "markNetSelectOverlapLayer" 'boolean t)
envSetVal("layout" "markNetSelectOverlapLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetStartLevel

`layout markNetStartLevel int` *`any_integer_from_0_to_32`*

Description

Sets the start level of the hierarchical range the Mark Net command uses to determine which nets will highlight and which will not. The default is `0`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Start* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetStartLevel")
envSetVal("layout" "markNetStartLevel" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetStopLayer

```
layout markNetStopLayer boolean { t | nil }
```

Description

Determines whether the settings defined for the stop layers (on the *Stop Layers* tab) are used for tracing a net. The default is `t`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Enable Stop Layers* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetStopLayer")
envSetVal("layout" "markNetStopLayer" 'boolean t)
envSetVal("layout" "markNetStopLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetStopLevel

`layout markNetStopLevel int` *`any_integer_from_0_to_32`*

Description

Sets the stop level of the hierarchical range the Mark Net command uses to determine which nets will highlight and which will not. The default is `0`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Stop* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetStopLevel")
envSetVal("layout" "markNetStopLevel" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetThickline

```
layout markNetThickline boolean { t | nil }
```

Description

Determines whether the traced nets are displayed in bold (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Thick Line* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetThickline")
envSetVal("layout" "markNetThickline" 'boolean t)
envSetVal("layout" "markNetThickline" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

markNetViaLayer

```
layout markNetViaLayer boolean { t | nil }
```

Description

Determines whether the settings defined for the via layers (on the *Via Layers* tab) are used for tracing a net. The default is `t`. If set to `nil`, the via layers defined in the technology file are used.

GUI Equivalent

Command	*Connectivity – Nets – Mark – Options*
Field	*Customized Via Layers* (MarkNet Options Form)

Examples

```
envGetVal("layout" "markNetViaLayer")
envSetVal("layout" "markNetViaLayer" 'boolean t)
envSetVal("layout" "markNetViaLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

maskLayoutDefaultApp

`graphic maskLayoutDefaultApp string` *`registered_application`*

Description

Automates the launching of a default application for cellviews of type `maskLayout`.

Note: This environment variable is ignored when launching Layout XL from a schematic window. In this case, the layout view always opens in Layout XL. This environment variable setting is also ignored when descending into or editing-in-place in the same view type.

Note: To apply this setting, the environment variable saveViewSubType must be set to nil, otherwise the Layout cellview automatically opens in the same application in which it was last saved.

GUI Equivalent

Command	*File – Set Default Application*
Field	*Set default application for all layout cellviews to* (Set Default Application Form)

Examples

```
envGetVal("graphic" "maskLayoutDefaultApp")
envSetVal("graphic" "maskLayoutDefaultApp" 'string "Layout XL")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

maxAutoStoreSelectionObj

`graphic maxAutoStoreSelectionObj int` *`integer_number`*

Description

Sets the maximum number of selected elements with a selection set for the *Max number of selected elements within a set* field on the Selection Options form, accessed by choosing *Options – Selection*. If the number of elements in the selection exceeds this number, the selection set is not kept. The default is `100`.

GUI Equivalent

Command	*Options – Selection*
Field	*Max number of selected elements within a set* (Selection Options Form)

Examples

```
envGetVal("graphic" "maxAutoStoreSelectionObj")
envSetVal("graphic" "maxAutoStoreSelectionObj" 'int 80)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

maxAutoStoreSelectionSet

`graphic maxAutoStoreSelectionSet int` *`integer_number`*

Description

Sets the maximum number of stored sets for the *Max number of Stored sets* field on the Selection Options form, accessed by choosing *Options – Selection*. The default is `10`.

GUI Equivalent

Command	*Options – Selection*
Field	*Max number of Stored sets* (Selection Options Form)

Examples

```
envGetVal("graphic" "maxAutoStoreSelectionSet")
envSetVal("graphic" "maxAutoStoreSelectionSet" 'int 7)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

maxPathSegInSpineAreaSelection

`graphic maxPathSegInSpineAreaSelection int` *`integer_number`*

Description

Controls the maximum number of pathsegs up to which spine can be computed. If the number of pathsegs exceeds the environment variable setting, the spine, comprising the pathsegs, is not computed. Setting this variable appropriately helps prevent performance issues. The default is `10000`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "maxPathSegInSpineAreaSelection")
envSetVal("graphic" "maxPathSegInSpineAreaSelection" 'int 9000)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

measurementDisplayOn

```
layout measurementDisplayOn boolean { t | nil }
```

Description

Determines whether dynamic measurement is set to on (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Options – Dynamic Display*
Field	*Measurement Display On* (Dynamic Display Form)

Examples

```
envGetVal("layout" "measurementDisplayOn")
envSetVal("layout" "measurementDisplayOn" 'boolean t)
envSetVal("layout" "measurementDisplayOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

measurementTextHeight

```
layout measurementTextHeight int { 4.0 |5.0 | 6.0| ... | 50.0 }
```

Description

Specifies the height of the text displayed during dynamic measurement. Valid values are any integer between `4` and `50`. The default value is `10`.

GUI Equivalent

Command	*Options – Dynamic Display – Dynamic Measurement*
Field	*Text Height (pixels)* (Dynamic Display Form)

Examples

```
envGetVal("layout" "measurementTextHeight")
envSetVal("layout" "measurementTextHeight" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mergeEnable

```
layout mergeEnable boolean { t | nil }
graphic mergeEnable boolean { t | nil }
```

Description

Makes the display more readable by removing all the extra edges in the design.

The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Merge Shapes (*Dynamic Selection Assistant Options Form*)*

Examples

```
envGetVal("layout" "mergeEnable")
envSetVal("layout" "mergeEnable" 'boolean t)
envSetVal("layout" "mergeEnable" 'boolean nil)
envGetVal("graphic" "mergeEnable")
envSetVal("graphic" "mergeEnable" 'boolean t)
envSetVal("graphic" "mergeEnable" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mergeScope

```
layout mergeScope cyclic { pcells | except current | all }
graphic mergeScope cyclic { pcells | except current | all }
```

Description

Specifies the hierarchies for which the *Enable Merge* command applies.

- `pcells`: All the Pcells in the hierarchy including the current cellview
- `except current`: All the hierarchies except the current cellview
- `all`: All the hierarchies including the current cellview

The default is `pcells`.

GUI Equivalent

Command	*Options – Display*
Field	*Scope* (Display Options Form)

Examples

```
envGetVal("layout" "mergeScope")
envSetVal("layout" "mergeScope" 'cyclic "pcells")
envSetVal("layout" "mergeScope" 'cyclic "except current")
envSetVal("layout" "mergeScope" 'cyclic "all")
envGetVal("graphic" "mergeScope")
envSetVal("graphic" "mergeScope" 'cyclic "pcells")
envSetVal("graphic" "mergeScope" 'cyclic "except current")
envSetVal("graphic" "mergeScope" 'cyclic "all")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mergeVias

```
graphic mergeVias boolean { t | nil }
```

Description

Skips via shapes during the Merge operation. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "mergeVias")
envSetVal("graphic" "mergeVias" 'boolean t)
envSetVal("graphic" "mergeVias" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mergeXStippleShapes

```
graphic mergeXStippleShapes boolean { t | nil }
```

Description

Merge layers with X stipple. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "mergeXStippleShapes")
envSetVal("graphic" "mergeXStippleShapes" 'boolean t)
envSetVal("graphic" "mergeXStippleShapes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

meshNBThread

`graphic meshNBThread int` *`any_integer`*

Description

Controls the maximum number of threads activated when the display is in high resolution. If this environment variable is set to an integer less than or equal to `0`, the display in high resolution runs in auto mode. The default value is `5`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "meshNBThread")
envSetVal("graphic " "meshNBThread" 'int 10)
envSetVal("graphic " "meshNBThread" 'int 0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

modalCommands

```
layout modalCommands boolean { t | nil }
```

Description

Specifies if commands are modal (automatically repeat). The default is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Repeat Commands* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "modalCommands")
envSetVal("layout" "modalCommands" 'boolean t)
envSetVal("layout" "modalCommands" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

modCornerRadius

`layout modCornerRadius float` *`float_number`*

Description

Specifies the default modify-corner-radius value. The default is `1.0` user units.

GUI Equivalent

Command	*Edit – Advanced – Modify Corner*
Field	*Radius* (Modify Corner Form)

Examples

```
envGetVal("layout" "modCornerRadius")
envSetVal("layout" "modCornerRadius" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

modCornerType

```
layout modCornerType cyclic { radial | chamfer }
```

Description

Specifies the default modify-corner type. The default is `radial`.

GUI Equivalent

Command	*Edit – Advanced – Modify Corner*
Field	*Type of Corner* (Modify Corner Form)

Examples

```
envGetVal("layout" "modCornerType")
envSetVal("layout" "modCornerType" 'cyclic "radial")
envSetVal("layout" "modCornerType" 'cyclic "chamfer")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mppASCIIFileName

```
layout mppASCIIFileName string { cellTechLibName.mpp.templates.il |
     your_ASCII_file_name.il }
```

Description

Specifies the default file name (or path and file name) displayed in the *ASCII File* field in the Save Multipart Path Templates form and the *Template File* field in the Load Multipart Path Template form. The default is `cellTechLibName.mpp.templates.il`.

GUI Equivalent

Command	*Create – Multipart Path – Load Template* or *Create – Multipart Path – Save Template*
Field	*Template File* (Load Multipart Path Template Form) or *ASCII File* (Save Multipart Path Templates Form)

Examples

```
envGetVal("layout" "mppASCIIFileName")
envSetVal("layout" "mppASCIIFileName" 'string "cellTechLibName.mpp.templates.il")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mppTemplate

```
layout mppTemplate string { New|
     the_name_of_any_MPP_template_defined_in_your_technology_file }
```

Description

Specifies the default for the *MPP Template* field in the *Create Multipart Path* form. You can choose *New* or the name of a specific MPP template you created. The default is `New`.

GUI Equivalent

Command	*Create – Multipart Path*
Field	*MPP Template* (Create Multipart Path Form)

Examples

```
envGetVal("layout" "mppTemplate")
envSetVal("layout" "mppTemplate" 'string "New")
envSetVal("layout" "mppTemplate" 'string "cellTechLibName.mpp.templates.il")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

mustConnectAllPins

```
layout mustConnectAllPins boolean { t | nil }
```

Description

Determines whether the `LEF58_MUSTJOINALLPORTS` property is to be applied to all newly created pin terminals in the layout, which implies that the router must connect all pins of the selected terminal. This setting is applicable only when *Connectivity* in the Create Pin form is set to *Weak*.

The default value is `nil`, in which case the property is not applied.

GUI Equivalent

Command	*Create – Pin*
Field	*Must Connect All Pins (*Create Pin Form*)*

Examples

```
envGetVal("layout" "mustConnectAllPins")
envSetVal("layout" "mustConnectAllPins" 'boolean t)
envSetVal("layout" "mustConnectAllPins" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

multiThreadInfo

```
graphic multiThreadInfo boolean { t | nil }
```

Description

Displays information messages about the number of threads running when redrawing the canvas. The number of threads varies according to the display resolution. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "multiThreadInfo")
envSetVal("graphic" "multiThreadInfo" 'boolean t)
envSetVal("graphic" "multiThreadInfo" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

muteUnstretchableOption

```
layout muteUnstretchableOption boolean { t | nil }
```

Description

Lets you control whether to display a dialog box when you attempt to stretch multiple figures and only some of the figures in the selected set can be legally stretched. When set to `nil`, a form opens to let you choose whether to stretch only the legally stretchable figures within the selected set or to cancel the stretch command. When set to `t`, no form appears, and you are only able to stretch the legally stretchable figures; figures that cannot legally be stretched do not stretch. The default value is `t`.

For example: When this variable is set to `nil`, you partially select three rectangles, and your stretch would cause one rectangle to have zero area, a dialog box appears, stating that one shape in the selected set will not be stretched. You can click *OK* to stretch the two stretchable rectangles and leave the un-stretchable rectangle un-stretched, or you can cancel the command.

GUI Equivalent

None

Examples

```
envGetVal("layout" "muteUnstretchableOption")
envSetVal("layout" "muteUnstretchableOption" 'boolean t)
envSetVal("layout" "muteUnstretchableOption" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netBasedSelectionNetName

```
layout netBasedSelectionNetName string { the_name_of_the_net }
```

```
graphic netBasedSelectionNetName string { the_name_of_the_net }
```

Description

Specifies the name of the nets for which the shapes, vias, and instances can be selected. You can specify the name of a single net or multiple nets separated by a space.

GUI Equivalent

None

Examples

```
envGetVal("layout" "netBasedSelectionNetName")
envSetVal("layout" "netBasedSelectionNetName" 'string "net1 net2")
envGetVal("graphic" "netBasedSelectionNetName")
envSetVal("graphic" "netBasedSelectionNetName" 'string "net1 net2")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netBasedSelectionOn

```
layout netBasedSelectionOn boolean { t | nil }
```

```
graphic netBasedSelectionOn boolean { t | nil }
```

Description

Determines whether the net-based selection option is enabled. The default value is `nil`.

GUI Equivalent

Command	*Options – Selection*
Field	*Net Based Selection* (Selection Options Form)

Examples

```
envGetVal("layout" "netBasedSelectionOn")
envSetVal("layout" "netBasedSelectionOn" 'boolean t)
envSetVal("layout" "netBasedSelectionOn" 'boolean nil)
envGetVal("graphic" "netBasedSelectionOn")
envSetVal("graphic" "netBasedSelectionOn" 'boolean t)
envSetVal("graphic" "netBasedSelectionOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netBasedSelectionReverse

```
layout netBasedSelectionReverse boolean { t | nil }
```

```
graphic netBasedSelectionReverse boolean { t | nil }
```

Description

Determines whether all assigned shapes, vias, and instances with a net other than that specified in the text field can be selected.The default value is `nil`.

GUI Equivalent

Command	*Windows – Toolbars – Net Based Selection*
Field	*Reverse*

Examples

```
envGetVal("layout" "netBasedSelectionReverse")
envSetVal("layout" "netBasedSelectionReverse" 'boolean t)
envSetVal("layout" "netBasedSelectionReverse" 'boolean nil)
envGetVal("graphic" "netBasedSelectionReverse")
envSetVal("graphic" "netBasedSelectionReverse" 'boolean t)
envSetVal("graphic" "netBasedSelectionReverse" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netExprDisplay

```
graphic netExprDisplay string { value only | value (default) | value [property] |
     value [expression] | expression only }
```

Description

Specifies how to display inherited connections values. The value is what the connection resolves to; if it is overridden at a higher level in the hierarchy, the full path to a local signal might be displayed. The default is `expression only`.

GUI Equivalent

Command	*Options – Display*
Field	*Net Expressions (*Display Options Form*)*

Examples

```
envGetVal("graphic" "netExprDisplay")
envSetVal("graphic" "netExprDisplay" 'string "value only")
envSetVal("graphic" "netExprDisplay" 'string "value (default)")
envSetVal("graphic" "netExprDisplay" 'string "value [property]")
envSetVal("graphic" "netExprDisplay" 'string "value [expression]")
envSetVal("graphic" "netExprDisplay" 'string "expression only")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netName

```
layout netName string { the_name_of_the_net }
```

Description

Specifies the name of the net with which you want to associate the object being created.

GUI Equivalent

Command	Occurs in multiple forms
Field	*Net Name*

Examples

```
envGetVal("layout" "netName")
envSetVal("layout" "netName" 'string "net1")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayDrawBackground

```
graphic netNameDisplayDrawBackground boolean { t | nil }
```

Description

Displays net name labels with a black background.

The default is `t`.

When set to `nil`, net name labels are displayed without a background.

GUI Equivalent

Command	*Options – Net Name Display*
Field	*Draw Background* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayDrawBackground")
envSetVal("graphic" "netNameDisplayDrawBackground" 'boolean t)
envSetVal("graphic" "netNameDisplayDrawBackground" 'boolean nil)
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayDrawOnTop

```
graphic netNameDisplayDrawOnTop boolean { t | nil }
```

Description

Displays net name labels on top of the shapes in the design.

The default is `nil`.

GUI Equivalent

Command	*Options – Net Name Display*
Field	*Draw Labels on Top* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayDrawOnTop")
envSetVal("graphic" "netNameDisplayDrawOnTop" 'boolean t)
envSetVal("graphic" "netNameDisplayDrawOnTop" 'boolean nil)
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayEnabled

```
graphic netNameDisplayEnabled boolean { t | nil }
```

Description

Displays net name labels on layout shapes that are big enough to display a label.

The default is `t`.

GUI Equivalent

Command	*Options – Net Name Display*
Field	*Enable net name display* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayEnabled")
envSetVal("graphic" "netNameDisplayEnabled" 'boolean t)
envSetVal("graphic" "netNameDisplayEnabled" 'boolean nil)
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayFilter

```
graphic netNameDisplayFilter boolean { t | nil }
```

Description

Controls the selective display of net name labels. This means, depending on the filter mode selected, the specified nets have their name labels either shown or hidden in the layout canvas.

The default is `nil`, which means all nets have their labels displayed.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*filter net names* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayFilter")
envSetVal("graphic" "netNameDisplayFilter" 'boolean t)
envSetVal("graphic" "netNameDisplayFilter" 'boolean nil)
```

Related Topics

Name Filtering

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayFilterDefaultList

`graphic netNameDisplayFilterDefaultList string "net1 net2 net3"`

Description

Adds a default list of net names to be used for name filtering, if no information is currently stored in the design.

The default is `""`. This means, by default, no filter list is available.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Filter net names* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayFilterDefaultList")
envSetVal("graphic" "netNameDisplayFilterDefaultList" 'string "vdd vss")
envSetVal("graphic" "netNameDisplayFilterDefaultList" 'string "")
```

Related Topics

Name Filtering

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayFilterLayers

```
graphic netNameDisplayFilterLayers boolean { t | nil }
```

Description

Controls the selective display of layers for which name of the net is not displayed.

The default is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Hide Layers* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayFilterLayers")
envSetVal("graphic" "netNameDisplayFilterLayers" 'boolean t)
envSetVal("graphic" "netNameDisplayFilterLayers" 'boolean nil)
```

Related Topics

Name Filtering

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayFilterMode

```
graphic netNameDisplayFilterMode boolean { t | nil }
```

Description

Controls whether the net name labels for the specified nets are displayed or hidden.

The default is `Hide`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Filter Mode* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayFilterMode")
envSetVal("graphic" "netNameDisplayFilterMode" 'boolean t)
envSetVal("graphic" "netNameDisplayFilterMode" 'boolean nil)
```

Related Topics

Name Filtering

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayFontStyle

```
graphic netNameDisplayFontStyle string { "euroStyle" | "gothic" | "math" | "roman"
     | "script" | "stick" | "fixed" | "swedish" | "raster" | "milSpec" | "serif" |
     "sansSerif" | "mono" | "serifItalic" | "serifBold" | "serifBoldItalic" |
     "sansBold" | "sansLight" | "nil" }
```

Description

Sets the font style to use for displaying the net name label.

The default is `"stick"`.

Note: The font style that you set using the environment variable applies only for the current session.

GUI Equivalent

None

Examples

```
nvGetVal("graphic" "netNameDisplayFontStyle")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "euroStyle")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "gothic")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "math")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "roman")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "script")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "stick")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "fixed")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "swedish")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "raster")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "milSpec")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "serif")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "sansSerif")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "mono")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "serifItalic")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "serifBold")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "serifBoldItalic")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "sansBold")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "sansLight")
envSetVal("graphic" "netNameDisplayFontStyle" 'string "nil")
```


© 2020 All Rights Reserved.

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayMaxLabelSize

`graphic netNameDisplayMaxLabelSize int` *`any_positive_integer`*

Description

Specifies the size of the label on the net. You can use the slider to select the size. The default is `15`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Label size slider* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayMaxLabelSize")
envSetVal("graphic" "netNameDisplayMaxLabelSize" 'int 30)
```

Related Topics

Name Filtering

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayMaxLabelSizeFull

```
graphic netNameDisplayMaxLabelSizeFull boolean { t | nil }
```

Description

Controls whether the label created is of maximum size such that it covers the net. The default is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Full size* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayMaxLabelSizeFull")
envSetVal("graphic" "netNameDisplayMaxLabelSizeFull" 'boolean t)
envSetVal("graphic" "netNameDisplayMaxLabelSizeFull" 'boolean nil)
```

Related Topics

Name Filtering

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayNetShow

```
graphic netNameDisplayNetShow boolean { t | nil }
```

Description

Displays net name labels on shapes. The default is `t`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Net Name: Show* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayNetShow")
envSetVal("graphic" "netNameDisplayNetShow" 'boolean t)
envSetVal("graphic" "netNameDisplayNetShow" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayScope

```
graphic netNameDisplayScope cyclic { edit | full }
```

Description

Displays the net name in full hierarchy or only in the current edit level.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Hierarchy Range* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayScope")
envSetVal("graphic" "netNameDisplayScope" 'cyclic "edit")
envSetVal("graphic" "netNameDisplayScope" 'cyclic "full")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplaySkipInstances

```
graphic netNameDisplaySkipInstances boolean { t | nil }
```

Description

Prevents net names from being displayed on instance pins.The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Net Name: On Instance pins* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplaySkipInstances")
envSetVal("graphic" "netNameDisplaySkipInstances" 'boolean t)
envSetVal("graphic" "netNameDisplaySkipInstances" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplaySkipPcells

```
graphic netNameDisplaySkipPcells boolean { t | nil }
```

Description

Prevents net names from being displayed on Pcell pins.The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Net Name: On Pcell pins* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplaySkipPcells")
envSetVal("graphic" "netNameDisplaySkipPcells" 'boolean t)
envSetVal("graphic" "netNameDisplaySkipPcells" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplaySkipLayers

`graphic netNameDisplaySkipLayers string "layerName1 layerName1 layerNamen"`

Description

Specifies the name of the layers for which the net name is not displayed.

The default is `""`. This means, by default, no filter list is available.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Layer Names* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplaySkipLayers")
envSetVal("graphic" "netNameDisplaySkipLayers" 'string "ndiff pdiff")
envSetVal("graphic" "netNameDisplaySkipLayers" 'string "")
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplaySkipVias

```
graphic netNameDisplaySkipVias boolean { t | nil }
```

Description

Prevents the net name labels from being displayed on vias.

The default is `nil`. This means, by default, net name labels are displayed on vias.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Skip labels on vias* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplaySkipVias")
envSetVal("graphic" "netNameDisplaySkipVias" 'boolean t)
envSetVal("graphic" "netNameDisplaySkipVias" 'boolean nil)
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayTermShow

```
graphic netNameDisplayTermShow boolean { t | nil }
```

Description

Displays terminal names on shapes.The default is `t`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Terminal Name: Show* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayTermShow")
envSetVal("graphic" "netNameDisplayTermShow" 'boolean t)
envSetVal("graphic" "netNameDisplayTermShow" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayTermSkipInstances

```
graphic netNameDisplayTermSkipInstances boolean { t | nil }
```

Description

Prevents terminal names from being displayed on instance pins. The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Terminal Name: On Instance pins* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayTermSkipInstances")
envSetVal("graphic" "netNameDisplayTermSkipInstances" 'boolean t)
envSetVal("graphic" "netNameDisplayTermSkipInstances" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayTermSkipPcells

```
graphic netNameDisplayTermSkipPcells boolean { t | nil }
```

Description

Prevents terminal names from being displayed on Pcell pins. The default value is `nil`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Terminal Name: On Pcell pins* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayTermSkipPcells")
envSetVal("graphic" "netNameDisplayTermSkipPcells" 'boolean t)
envSetVal("graphic" "netNameDisplayTermSkipPcells" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayUserColor

```
graphic netNameDisplayUserColor string { "white" | "yellow" | "anyColorFromPalette"
     }
```

Description

Sets the font color to use for the net name label. You can choose from any of the colors in the palette.

The default is `"white"`.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Color – User* (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayUserColor")
envSetVal("graphic" "netNameDisplayUserColor" 'string "magenta")
envSetVal("graphic" "netNameDisplayUserColor" 'string "cyan")
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netNameDisplayUseShapeColor

```
graphic netNameDisplayUseShapeColor boolean { t | nil }
```

Description

Sets the font color for the net name label to that of the shape with which the label is associated.

The default is `"nil"`. This means all the net name labels are drawn using the font color that is either specified using the netNameDisplayUserColor environment variable or selected from the *Color – User* color palette in the Net Name Display Options form.

GUI Equivalent

Command	(In Layout XL) *Options – Net Name Display*
Field	*Color – Same as Shape*
	Color – User (Net Name Display Options form)

Examples

```
envGetVal("graphic" "netNameDisplayUseShapeColor")
envSetVal("graphic" "netNameDisplayUseShapeColor" 'boolean t)
envSetVal("graphic" "netNameDisplayUseShapeColor" 'boolean nil)
```

Related Topics

Enabling Net Name Display

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

netsOn

```
layout netsOn boolean { t | nil }
graphic netsOn boolean { t | nil }
```

Description

Determines whether nets are highlighted (`t`), or not (`nil`). The layout default is `nil`, the graphic default is `t`.

GUI Equivalent

Command	*Options – Display*
Form Field	*Nets* (Display Options Form)

Examples

```
envGetVal("layout" "netsOn")
envSetVal("layout" "netsOn" 'boolean t)
envSetVal("layout" "netsOn" 'boolean nil)
envGetVal("graphic" "netsOn")
envSetVal("graphic" "netsOn" 'boolean t)
envSetVal("graphic" "netsOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

nPtsToTaper

`layout nPtsToTaper int` *`any_positive_integer`*

Description

Specifies the default number of segments in the taper. This is only used for exponentially tapered lines. The default is `20`.

GUI Equivalent

Command	*Create – Microwave – Taper*
Field	*Resolution* (Transmission Line Taper Form)

Examples

```
envGetVal("layout" "nPtsToTaper")
envSetVal("layout" "nPtsToTaper" 'int 30)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

numLevels

`layout numLevels int` *`any_integer_from_0_to_32`*

Description

Specifies the number of levels to copy in `leYankFigs`. The default is `32`.

GUI Equivalent

Command	*Edit – Basic – Yank*
Field	*Yank Levels* (Yank Form)

Examples

```
envGetVal("layout" "numLevels")
envSetVal("layout" "numLevels" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

numSides

`layout numSides int` *`any_positive_integer_from_3_to_4000`*

Description

Specifies the number of sides used to convert a conic to a polygon. The default is `20`.

GUI Equivalent

Command	*Options – Editor*
Field	*Conic Sides* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "numSides")
envSetVal("layout" "numSides" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

openSavedCellViewInWindow

```
layout openSavedCellViewInWindow boolean { t | nil }
```

Description

Determines whether you want to open a saved copy of the design. The default is `nil`.

GUI Equivalent

Command	*File – Save a Copy*
Field	*Open Saved Copy* (Save a Copy Form)

Examples

```
envGetVal("layout" "openSavedCellViewInWindow")
envSetVal("layout" "openSavedCellViewInWindow" 'boolean t)
envSetVal("layout" "openSavedCellViewInWindow" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

openToStopLevel

```
layout openToStopLevel boolean { t | nil }
graphic openToStopLevel boolean { t | nil }
```

Description

Determines when the cellview data is loaded during rendering. When set to `t` all data is loaded to the window stop level (corresponding to the `displayStopLevel` variable), then rendered. All data is loaded, even data too small to see. When set to `nil` data loads on demand during rendering, to the `filterSize` limit. This can slightly slow initial redraw, zooms or pans, but will dramatically reduce the amount of data loaded, since data too small to see is not rendered. The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Open To Stop Level* (Display Options Form)

Examples

```
envGetVal("layout" "openToStopLevel")
envSetVal("layout" "openToStopLevel" 'boolean t)
envSetVal("layout" "openToStopLevel" 'boolean nil)
envGetVal("graphic" "openToStopLevel")
envSetVal("graphic" "openToStopLevel" 'boolean t)
envSetVal("graphic" "openToStopLevel" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

orientation

```
layout orientation cyclic { R0 | R90 | R180 | R270 | MY | MYR90 | MX | MXR90 }
```

Description

Specifies the incremental orientation with which objects are placed during creation or editing, with an initial setting of `R0`. This environment variable corresponds to the *Rotate*, *Sideways*, and *Upside Down* buttons on any form. It resets to `R0` after each usage, so you can use it only to set an orientation bindkey. Each time you press the bindkey the value specified is added to whichever one of the eight legal orientations the object is currently in (`R0`, `R90`, `R180`, `R270`, `MY`, `MYR90`, `MX`, or `MXR90`). Therefore, this variable is only used as a relative value. The default is `R0`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "orientation")
envSetVal("layout" "orientation" 'cyclic "R0")
envSetVal("layout" "orientation" 'cyclic "R90")
envSetVal("layout" "orientation" 'cyclic "R180")
envSetVal("layout" "orientation" 'cyclic "R270")
envSetVal("layout" "orientation" 'cyclic "MY")
envSetVal("layout" "orientation" 'cyclic "MYR90")
envSetVal("layout" "orientation" 'cyclic "MX")
envSetVal("layout" "orientation" 'cyclic "MXR90")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

originMarkersOn

```
layout originMarkersOn boolean { t | nil }
graphic originMarkersOn boolean { t | nil }
```

Description

Determines whether the cell origins are displayed when viewing the bounding boxes of cell instances (`t`), or not (`nil`). The origins appear as plus signs. The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Instance Origins* (Display Options Form)

Examples

```
envGetVal("layout" "originMarkersOn")
envSetVal("layout" "originMarkersOn" 'boolean t)
envSetVal("layout" "originMarkersOn" 'boolean nil)
envGetVal("graphic" "originMarkersOn")
envSetVal("graphic" "originMarkersOn" 'boolean t)
envSetVal("graphic" "originMarkersOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

overrideFontName

`layout overrideFontName string "font_name"`

`graphic overrideFontName string "font_name"`

Description

Specifies fonts that override the default fonts. Supported font choices are `euroStyle`, `gothic`, `math`, `roman`, `script`, `stick`, `fixed`, `swedish`, `raster`, `milSpec`, `serif`, `sansSerif`, `mono`, `serifItalic`, `serifBold`, `serifBoldItalic`, `sansBold`, and `sansLight`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "overrideFontName")
envSetVal("layout" "overrideFontName" 'string "(stick sansLight) (gothic raster)"
)
envGetVal("graphic" "overrideFontName")
envSetVal("graphic" "overrideFontName" 'string "(stick sansLight) (gothic raster)"
)
```

In the above example, the second pair of font replaces the first one.

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

packetDialogBoxes

```
graphic packetDialogBoxes boolean { t | nil }
```

Description

Determines whether a message box appears if a packet assigned to a layer in the technology file is not defined in the `display.drf` file. The message box lists the undefined packets. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "packetDialogBoxes")
envSetVal("graphic" "packetDialogBoxes" 'boolean t)
envSetVal("graphic" "packetDialogBoxes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

padLabelLPP

`layout padLabelLPP string { lpp }`

Description

Specifies the layer-purpose pair on which the *Pad Opening Info* command draws pad labels. The default is `"text drawing"`.

GUI Equivalent

Command	*Tools – Pad Opening Info; Mode: Generate Labels*
Field	*Label Control: Label Options: Layer/Purpose* (Pad Opening Info Form)

Examples

```
envGetVal("layout" "padLabelLPP")
envSetVal("layout" "padLabelLPP" 'string "edgeLayer drawing Via5 drawing Metal7
drawing NWell drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

partialSelect

```
layout partialSelect boolean { t | nil }
```

```
graphic partialSelect boolean { t | nil }
```

Description

Controls whether you can select vertices or line segments of an object (`t`), or not (`nil`). The layout default is `nil`, the graphic default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "partialSelect")
envSetVal("layout" "partialSelect" 'boolean t)
envSetVal("layout" "partialSelect" 'boolean nil)
envGetVal("graphic" "partialSelect")
envSetVal("graphic" "partialSelect" 'boolean t)
envSetVal("graphic" "partialSelect" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

partialSelectProximityMaxPixel

`graphic partialSelectProximityMaxPixel int` *`integer_number`*

Description

Overrides the current shape selection behavior. Setting this variable, will enable you to select the entire shape by clicking anywhere on the center line or by selecting an edge, when partial select is on and you are zoomed in at the shape end. The default value is `60`. To override current selection behavior, set this value to `5`.

Note: If you want to select a vertex, the distance between the vertex and the cursor should be less than the value you set here.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "partialSelectProximityMaxPixel")
envSetVal("graphic" "partialSelectProximityMaxPixel" 'int 50)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

partialSelectionMode

```
graphic partialSelectionMode string { vertexAndEdge | enclosedEdge | crossedEdge |
     enclosedAndCrossed}
```

Description

Specifies whether the partial selection mode is set to vertex and edge (`vertexAndEdge`), enclosed edge (`enclosedEdge`) or crossed edge (`crossedEdge`), or enclosed and crossed (`enclosedAndCrossed`). Vertex (`vertexAndEdge`) fully selects all objects you completely enclose with the area selection box, vertices of any objects you partially select with the area selection box, and edges of any objects you enclose with the area selection box. Enclosed Edges (`enclosedEdge`) fully selects all objects you completely enclose with the area selection box, and the edges of any objects you partially enclose with the area selection box. Crossed Edges (`crossedEdge`) selects the edges of any objects you enclose with the area selection box, and does not select fully enclosed objects. Enclosed and Crossed Edges (`enclosedAndCrossed`) fully selects all objects you completely enclose with the area selection box, the vertices of any objects you partially enclose with the area selection box, and the edges of any objects you enclose with the area selection box. The default is `vertexAndEdge`.

GUI Equivalent

Command	*Options – Selection*
Field	*Partial Mode* (Selection Options Form)

Examples

```
envGetVal("graphic" "partialSelectionMode")
envSetVal("graphic" "partialSelectionMode" 'string "vertexAndEdge")
envSetVal("graphic" "partialSelectionMode" 'string "enclosedEdge")
envSetVal("graphic" "partialSelectionMode" 'string "crossedEdge")
envSetVal("graphic" "partialSelectionMode" 'string "enclosedAndCrossed")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pasteRef

```
layout pasteRef cyclic { origin | lowerLeft | lowerCenter | lowerRight | centerLeft
     | centerCenter | centerRight | upperLeft | upperCenter | upperRight | user
     specified}
```

Description

Specifies the reference point of the bounding box when pasting. You can select one of the ten reference points when pasting. The reference points are `origin`, `lowerLeft`, `centerLeft`, `upperLeft`, `lowerCenter`, `centerCenter`, `upperCenter`, `lowerRight`, `centerRight`, `upperRight`, and `user specified`.

GUI Equivalent

Command	*Edit – Basic – Paste*
Field	*Reference Points* (Paste Form)

Examples

```
envGetVal("layout" "pasteRef")
envSetVal("layout" "pasteRef" 'cyclic "origin")
envSetVal("layout" "pasteRef" 'cyclic "lowerLeft")
envSetVal("layout" "pasteRef" 'cyclic "lowerCenter")
envSetVal("layout" "pasteRef" 'cyclic "lowerRight")
envSetVal("layout" "pasteRef" 'cyclic "centerLeft")
envSetVal("layout" "pasteRef" 'cyclic "centerCenter")
envSetVal("layout" "pasteRef" 'cyclic "centerRight")
envSetVal("layout" "pasteRef" 'cyclic "upperLeft")
envSetVal("layout" "pasteRef" 'cyclic "upperCenter")
envSetVal("layout" "pasteRef" 'cyclic "upperRight")
envSetVal("layout" "pasteRef" 'cyclic "user specified")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathBeginExt

`layout pathBeginExt float` *`float_number`*

Description

Specifies the beginning extension of paths with variable end type. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Shape – Path* and *Create – Wiring –Wire*
Field	*Begin Extension* (Create Path Form) and *Begin* (Create Wire Form)

Examples

```
envGetVal("layout" "pathBeginExt")
envSetVal("layout" "pathBeginExt" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathCL

```
layout pathCL cyclic { yes | no | only }
graphic pathCL string { yes | no | only }
```

Description

Determines whether you see path borders and center lines (`yes`), path borders only (`no`), or path center lines only (`only`). The default is `yes`.

GUI Equivalent

Command	*Options – Display*
Field	*Path Display* (Display Options Form)

Examples

```
envGetVal("layout" "pathCL")
envSetVal("layout" "pathCL" 'cyclic "yes")
envSetVal("layout" "pathCL" 'cyclic "no")
envSetVal("layout" "pathCL" 'cyclic "only")
envGetVal("graphic" "pathCL")
envSetVal("graphic" "pathCL" 'string "yes")
envSetVal("graphic" "pathCL" 'string "no")
envSetVal("graphic" "pathCL" 'string "only")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathEndExt

`layout pathEndExt float` *`float_number`*

Description

Specifies the end extension of paths with variable end type. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Shape – Path* and *Create – Wiring –Wire*
Field	*End Extension* (Create Path Form) and *End* (Create Wire Form)

Examples

```
envGetVal("layout" "pathEndExt")
envSetVal("layout" "pathEndExt" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathJustify

```
layout pathJustify cyclic { left | center | right }
```

Description

Specifies the justification of paths. The default is `center`.

GUI Equivalent

Command	*Create – Shape – Path*
Field	*Justification* (Create Path Form)

Examples

```
envGetVal("layout" "pathJustify")
envSetVal("layout" "pathJustify" 'cyclic "left")
envSetVal("layout" "pathJustify" 'cyclic "center")
envSetVal("layout" "pathJustify" 'cyclic "right")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathOffset

`layout pathOffset float` *`float_number`*

Description

Specifies the offset of a path center-line from the drawn coordinates. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Shape – Path*
Field	*Offset* (Create Path Form)

Examples

```
envGetVal("layout" "pathOffset")
envSetVal("layout" "pathOffset" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathStyle

```
layout pathStyle cyclic { truncateExtend | extendExtend | roundRound |
     varExtendExtend }
```

Description

Specifies the default end type for paths. The end type can be `truncateExtend` (truncate), `extendExtend` (extend), `roundRound` (round), or `varExtendExtend` (variable). The default is `truncateExtend`.

Note: (ICADVM20.1 Virtuoso RF Solution Only) The default is `extendExtend`.

GUI Equivalent

Command	*Create – Shape – Path* and *Create – Wiring –Wire*
Field	*End Type* (Create Path Form) and *End Style* (Create Wire Form)

Examples

```
envGetVal("layout" "pathStyle")
envSetVal("layout" "pathStyle" 'cyclic "truncateExtend")
envSetVal("layout" "pathStyle" 'cyclic "extendExtend")
envSetVal("layout" "pathStyle" 'cyclic "roundRound")
envSetVal("layout" "pathStyle" 'cyclic "varExtendExtend")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pathWidth

`layout pathWidth float` *`float_number`*

`graphic pathWidth float` *`float_number`*

Description

Specifies the default width of paths. The layout default is `0.6` user units, the graphic default is `5.0` user units.

GUI Equivalent

Command	*Create – Shape – Path*
Field	*Width* (Create Path Form)

Examples

```
envGetVal("layout" "pathWidth")
envSetVal("layout" "pathWidth" 'float 2.0)
envGetVal("graphic" "pathWidth")
envSetVal("graphic" "pathWidth" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicDisplay

```
graphic photonicDisplay boolean { t | nil }
```

Description

(ICADVM20.1 Photonics Only) Controls whether to display photonic pin attributes in layout (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "photonicDisplay")
envSetVal("graphic" "photonicDisplay" 'boolean t)
envSetVal("graphic" "photonicDisplay" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinWidth

`layout photonicPinWidth float` *`float_number`*

Description

(ICADVM20.1 Photonics Only) Specifies the width of the facet of the waveguide that connects to a pin. The default value is `0.1`.

GUI Equivalent

Command	*Create – Pin – Manual*
Field	*Photonic Width* (Create Pin Form)

Examples

```
envGetVal("layout" "photonicPinWidth")
envSetVal("layout" "photonicPinWidth" 'float 0.5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinAngle

`layout photonicPinAngle float` *`float_number`*

Description

(ICADVM20.1 Photonics Only) Specifies the angle of the facet of the waveguide at the intersection of the pin. The default value is `0.0`.

GUI Equivalent

Command	*Create – Pin – Manual*
Field	*Photonic Angle* (Create Pin Form)

Examples

```
envGetVal("layout" "photonicPinAngle")
envSetVal("layout" "photonicPinAngle" 'float 15)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinFacetInPacket

`graphic photonicPinFacetInPacket string` *`packetName`*

Description

(ICADVM20.1 Photonics Only) Specifies the name of the packet for the pin on which a waveguide facet is drawn inside.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "photonicPinFacetInPacket")
envSetVal("graphic" "photonicPinFacetInPacket" 'string "phoPinFacetIn")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinFacetOutPacket

`graphic photonicPinFacetOutPacket string` *`packetName`*

Description

(ICADVM20.1 Photonics Only) Specifies the name of the packet for the pin on which a waveguide facet is drawn outside.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "photonicPinFacetOutPacket")
envSetVal("graphic" "photonicPinFacetOutPacket" 'string "phoPinFacetOut")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinLabelPacket

`graphic photonicPinLabelPacket string` *`packetName`*

Description

(ICADVM20.1 Photonics Only) Specifies the name of the packet in which the photonic pin label is drawn.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "photonicPinLabelPacket")
envSetVal("graphic" "photonicPinLabelPacket" 'string "phoPinLabel")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinRadius

`layout photonicPinRadius float` *`float_number`*

Description

(ICADVM20.1 Photonics Only) Specifies the radius of the curve of the waveguide center at the point it meets the facet.The default value is `0.0`.

GUI Equivalent

Command	*Create – Pin – Manual*
Field	*Photonic Radius* (Create Pin Form)

Examples

```
envGetVal("layout" "photonicPinRadius")
envSetVal("layout" "photonicPinRadius" 'float 2)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinRadiusLinePacket

`graphic photonicPinRadiusLinePacket string` *`packetName`*

Description

(ICADVM20.1 Photonics Only) Specifies the name of the packet in which the photonic pin radius line is drawn.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "photonicPinRadiusLinePacket")
envSetVal("graphic" "photonicPinRadiusLinePacket" 'string "phoPinRadiusLine")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

photonicPinWidthLinePacket

`graphic photonicPinWidthLinePacket string` *`packetName`*

Description

(ICADVM20.1 Photonics Only) Specifies the name of the packet in which the photonic pin width line is drawn.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "photonicPinWidthLinePacket")
envSetVal("graphic" "photonicPinWidthLinePacket" 'string "phoPinWidthLine")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinAccessDir

`layout pinAccessDir string { any_combination_of _the_default_values | "" }`

Description

Specifies the default access direction of a pin. The default is `list ("left" "right" "top" "bottom")`.

GUI Equivalent

Command	*Create – Pin*
Form Field	*Access Direction*

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinDefaultObjectType

```
layout pinDefaultObjectType cyclic ( Label | Text Display | Both )
```

Description

Specifies the default object type while creating pins from objects.The default value is `Both`.

GUI Equivalent

Command	*Tools – Create Pins From Labels*
Form Field	*Object Type*

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinLabelPlacementMode

```
layout pinLabelPlacementMode cyclic { manual | auto }
```

Description

Determines the mode to create label for pins, manual or auto. The default is manual.

GUI Equivalent

Command	*Create – Pin*
Form Field	*Create Label*

Examples

```
envGetVal("layout" "pinLabelPlacementMode")
envSetVal("layout" "pinLabelPlacementMode" 'cyclic "manual")
envSetVal("layout" "pinLabelPlacementMode" 'cyclic "auto")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinLayerPurpose

`layout pinLayerPurpose string { any_valid_layer_purpose | "" }`

Description

Specifies the layer purpose on which a shape pin is placed. You can specify any valid layer purpose, such as `"pin"` or `"drawing"`. The default is " ".

GUI Equivalent

None

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinLPP

```
layout pinLPP cyclic { sameLayer | currentLayer }
```

Description

Determines the layer purpose pair to be used for creating auto pins. The default is `sameLayer`.

GUI Equivalent

Command	*Create – Pin; Mode: Auto*
Form Field	*Pin Layer/Purpose*

Examples

```
envGetVal("layout" "pinLPP")
envSetVal("layout" "pinLPP" 'cyclic "sameLayer")
envSetVal("layout" "pinLPP" 'cyclic "sameLayer")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinMode

```
layout pinMode cyclic { Manual | Auto }
```

Description

Determines the mode for creating pins. The default is `Manual`.

GUI Equivalent

Command	*Create – Pin*
Field	*Mode* (Create Pin Form)

Examples

```
envGetVal("layout" "pinMode")
envSetVal("layout" "pinMode" 'cyclic "Manual")
envSetVal("layout" "pinMode" 'cyclic "Auto")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinSelectionMode

```
layout pinSelectionMode cyclic { ignoreLPP | useLPPIncludeShapes |
     useLPPExcludeShapes }
```

Description

Determines the selectability of pins and shapes.

- If set to `ignoreLPP`, pins are selectable if the *Pin* check box in the *Objects* assistant is selected.
- If set to `useLPPIncludeShapes`, instance pins are selectable if the *Pin* check box in the *Object* pane is selected and the LPP of the shape corresponding to the instance pin is set to selectable in the *Layer* pane.
- If set to `useLPPExcludeShapes`, instance pins are selectable only if the *Pin* check box in the *Object* pane is selected. The LPP of the shape corresponding to the instance pin is not considered.

The default is `useLPPIncludeShapes`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pinSelectionMode")
envSetVal("layout" "pinSelectionMode" 'cyclic "ignoreLPP")
envSetVal("layout" "pinSelectionMode" 'cyclic "useLPPIncludeShapes")
envSetVal("layout" "pinSelectionMode" 'cyclic "useLPPExcludeShapes")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinShape

```
layout pinShape cyclic { rectangle | polygon | auto pin | dot | circle | instance
     | via | existing object}
```

Description

Specifies the default type of pin shape. The default is `rectangle`.

GUI Equivalent

Command	*Create – Pin*
Field	*Pin Shape* (Create Pin Form)

Examples

```
envGetVal("layout" "pinShape")
envSetVal("layout" "pinShape" 'cyclic "rectangle")
envSetVal("layout" "pinShape" 'cyclic "polygon")
envSetVal("layout" "pinShape" 'cyclic "auto pin")
envSetVal("layout" "pinShape" 'cyclic "dot")
envSetVal("layout" "pinShape" 'cyclic "circle")
envSetVal("layout" "pinShape" 'cyclic "instance")
envSetVal("layout" "pinShape" 'cyclic "via")
envSetVal("layout" "pinShape" 'cyclic "existing object")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextCreateAsLabel

```
layout pinTextCreateAsLabel boolean { t | nil }
```

Description

Specifies that a pin is created as a label (`t`) or a text display (`nil`). The default is `nil`.

GUI Equivalent

Command	*Create – Pin – Create Label – Options – Set Pin Label Text Style*
Field	*Text Options: Create As Label* (Set Pin Label Text Style Form)

Examples

```
envGetVal("layout" "pinTextCreateAsLabel")
envSetVal("layout" "pinTextCreateAsLabel" 'boolean t)
envSetVal("layout" "pinTextCreateAsLabel" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextLayer

`layout pinTextLayer string` *`any_valid_layer`*

Description

Specifies the layer of the layer-purpose pair used for pin text displays and on the Pin Name Display form. These are the labels created for pin names. The default is `text`.

GUI Equivalent

Command	*Create – Pin – Option*
Field	*Layer Name* (Set Pin Label Text Style Form)

Examples

```
envGetVal("layout" "pinTextLayer")
envSetVal("layout" "pinTextLayer" 'string "via1")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextLayerNames

```
layout pinTextLayerNames string { "Pimp" | "Poly" | "any_other_validLayer_name" }
```

Description

Specifies the layer names used to filter the Layer Names for the Set Pin Label Text Style form.

The default is `""`.

Note: If the layer names to be used for the filter are not specified, or none of the specified layer names match a valid layer name, all the valid layers are used.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pinTextLayerNames")
envSetVal("layout" "pinTextLayerNames" 'string "prBoundary")
envSetVal("layout" "pinTextLayerNames" 'string "Resdum")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextPurpose

`layout pinTextPurpose string` *`any_valid_purpose`*

Description

Specifies the purpose of the layer-purpose pair used for pin text displays and on the Pin Name Display form. These are the labels created for pin names. The default is `drawing`.

GUI Equivalent

Command	*Create – Pin – Option*
Field	*Layer Purpose* (Set Pin Label Text Style Form)

Examples

```
envGetVal("layout" "pinTextPurpose")
envSetVal("layout" "pinTextPurpose" 'string "drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextPurposeNames

```
layout pinTextPurposeNames string { "drawing" | "any_other_validPurpose_name" }
```

Description

Specifies the layer purpose names used to filter the Layer Purpose for the Set Pin Label Text Style Form form.

The default is `""`.

Note: If the layer purpose names to be used for the filter are not specified, or none of the specified layer purpose names match a valid layer purpose, all the valid layer purposes are used.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pinTextPurposeNames")
envSetVal("layout" "pinTextPurposeNames" 'string "drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextSamePurpose

```
layout pinTextSamePurpose boolean { t | nil }
```

Description

Specifies whether the pin name should have the same purpsoe as the pin (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Create – Pin – Create Label – Options – Set Pin Label Text Style*
Field	*Layer Purpose: Same As Pin* (Set Pin Label Text Style Form)

Examples

```
envGetVal("layout" "pinTextSamePurpose")
envSetVal("layout" "pinTextSamePurpose" 'boolean t)
envSetVal("layout" "pinTextSamePurpose" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextSameLayer

```
layout pinTextSameLayer boolean { t | nil }
```

Description

Determines whether to create the pin name label on the same layer/purpose as the pin being created or create it on the layer specified by `pinTextLayer` and the purpose specified by `pinTextPurpose`. When set to `t`, creates the pin name label on the same layer/purpose as the pin being created. When set to `nil`, the pin name is created on the layer specified by `pinTextLayer` and the purpose specified by `pinTextPurpose`. The default is `nil`.

GUI Equivalent

Command	*Create – Pin – Create Label – Options – Set Pin Label Text Style*
Field	*Layer Name: Same As Pin* (Set Pin Label Text Style Form)

Examples

```
envGetVal("layout" "pinTextSameLayer")
envSetVal("layout" "pinTextSameLayer" 'boolean t)
envSetVal("layout" "pinTextSameLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinTextOrientation

```
layout pinTextOrientation cyclic { RO | R90 | R180 | R270 | MY | MYR90 | MX | MXR90}
```

Description

Sets the orientation of the pin name on the Set Pin Label Text Style form. The valid values are `R0`, `R90`, `R180`, `R270`, `MX`, `MY`, `MXR90`, and `MYR90`. The default value is `R0`.

GUI Equivalent

Command	*Create – Pin – Create Label – Options – Set Pin Label Text Style*
Field	*Orientation* (Set Pin Label Text Style Form)

Examples

```
envGetVal("layout" "pinTextOrientation")
envSetVal("layout" "pinTextOrientation" 'cyclic "RO")
envSetVal("layout" "pinTextOrientation" 'cyclic "R90")
envSetVal("layout" "pinTextOrientation" 'cyclic "R180")
envSetVal("layout" "pinTextOrientation" 'cyclic "R270")
envSetVal("layout" "pinTextOrientation" 'cyclic "MY")
envSetVal("layout" "pinTextOrientation" 'cyclic "MYR90")
envSetVal("layout" "pinTextOrientation" 'cyclic "MX")
envSetVal("layout" "pinTextOrientation" 'cyclic "MXR90")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinWidth

`layout pinWidth float` *`float_number`*

Description

Specifies the default pin width. The default is as defined in the technology file.

GUI Equivalent

Command	*Create – Pin; Mode: Auto*
Field	*Width* (Create Pin Form)

Examples

```
envGetVal("layout" "pinWidth")
envSetVal("layout" "pinWidth" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinXPitch

`layout pinXPitch float` *`float_number`*

Description

Specifies the horizontal distance between the center points of pins in an array. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Pin*
Field	*X Pitch* (Create Pin Form)

Examples

```
envGetVal("layout" "pinXPitch")
envSetVal("layout" "pinXPitch" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pinYPitch

`layout pinYPitch float` *`float_number`*

Description

Specifies the vertical distance between the center points of pins in an array. The default is `0.0` user units.

GUI Equivalent

Command	*Create – Pin*
Field	*Y Pitch* (Create Pin Form)

Examples

```
envGetVal("layout" "pinYPitch")
envSetVal("layout" "pinYPitch" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pointSelectAperture

`graphic pointSelectAperture int` *`integer_number`*

Description

Sets the number of pixels for *aperture* for the *Point Selection Mode* field on the *Selection Options* form, accessed by choosing *Options – Selection*. The aperture defines a box around the point inferred by a mouse click to consolidate unintentional pointer movements between two clicks due to a jittery hand. Two clicks within the aperture are treated as a double click the same point. The default is `5` pixels.

GUI Equivalent

Command	*Options – Selection*
Field	*Point Selection Controls: User Aperture Mode* (Selection Options Form)

Examples

```
envGetVal("graphic" "pointSelectAperture")
envSetVal("graphic" "pointSelectAperture" 'int 7)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pointSelectCycleMode

```
graphic pointSelectCycleMode string { toggle | cycle }
```

Description

Sets the *Overlap Mode* cyclic field to *Toggle* or *Cycle* in the *Point Selection Controls* section on the *Selection Options* form, accessed by choosing *Options – Selection*. The default is `"toggle"`.

GUI Equivalent

Command	*Options – Selection*
Field	*Overlap Mode: Toggle* or *Overlap Mode: Cycle* (Selection Options Form)

Examples

```
envGetVal("graphic" "pointSelectCycleMode")
envSetVal("graphic" "pointSelectCycleMode" 'string "toggle")
envSetVal("graphic" "pointSelectCycleMode" 'string "cycle")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

polygonArc

```
layout polygonArc boolean { t | nil }
```

Description

When set to `t`, indicates that the next three points you enter in the polygon define an arc. Resets to `nil` when you enter the third point. The default is `nil`.

GUI Equivalent

Command	*Create – Shape – Polygon*
Field	*Create Arc* (Create Polygon Form)

Examples

```
envGetVal("layout" "polygonArc")
envSetVal("layout" "polygonArc" 'boolean t)
envSetVal("layout" "polygonArc" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

preservePins

```
layout preservePins boolean { t | nil }
```

Description

Specifies whether pin information is preserved when an instance or mosaic is flattened (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Preserve: Pins* (Flatten Form)

Examples

```
envGetVal("layout" "preservePins")
envSetVal("layout" "preservePins" 'boolean t)
envSetVal("layout" "preservePins" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

preservePinFigs

```
layout preservePinFigs boolean { t | nil }
```

Description

Specifies whether the figure of the pin is preserved when the instance is flattened. When set to `t`, the figure moves up to the top level without any connectivity information; when set to `nil`, no figure is preserved and there is no connectivity. The default is `t`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Preserve: Pin geometries* (Flatten Form)

Examples

```
envGetVal("layout" "preservePinFigs")
envSetVal("layout" "preservePinFigs" 'boolean t)
envSetVal("layout" "preservePinFigs" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

preserveRodObjects

```
layout preserveRodObjects boolean { t | nil }
```

Description

Specifies whether ROD properties are preserved when an instance is flattened (`t`), or not (`nil`). The default is `nil`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Preserve: ROD objects* (Flatten Form)

Examples

```
envGetVal("layout" "preserveRodObjects")
envSetVal("layout" "preserveRodObjects" 'boolean t)
envSetVal("layout" "preserveRodObjects" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

preserveTerminalContacts

```
layout preserveTerminalContacts boolean { t | nil }
```

Description

Setting this environment variable to `t` means that the terminal contacts will be preserved when you abut devices in Layout XL.The default is `nil`.

Tip

To preserve properties on a single instance, create a boolean property called `preserveTerminalContacts` with a value of `t` on the instance in question. This lets you preserve properties (and therefore contacts) on selected instances while dropping them for all other instances.

GUI Equivalent

None

Examples

```
envGetVal("layout" "preserveTerminalContacts")
envSetVal("layout" "preserveTerminalContacts" 'boolean t)
envSetVal("layout" "preserveTerminalContacts" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

preserveTermName

```
layout preserveTermName boolean { t | nil }
```

Description

Specifies whether the terminal name of the pin is preserved when the instance is flattened. When set to `t`, the terminal name of the pin is preserved when the instance is flattened. When set to `nil`, the instance name is added as a prefix to the terminal name of the pin when the instance is flattened. The default is `nil`.

GUI Equivalent

Command	*Edit – Hierarchy – Flatten*
Field	*Preserve: Term name* (Flatten Form)

Examples

```
envGetVal("layout" "preserveTermName")
envSetVal("layout" "preserveTermName" 'boolean t)
envSetVal("layout" "preserveTermName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

preventDeletionOfLogicalInstance

```
layout preventDeletionOfLogicalInstance boolean { t | nil }
```

Description

Prevents the deletion of non-physical-only instances. The default is `nil`.

GUI Equivalent

Command	*Edit – Delete*
Field	*Preserve Non-Physical-Only Instances* (Delete Form)

Examples

```
envGetVal("layout" "preventDeletionOfLogicalInstance")
envSetVal("layout" "preventDeletionOfLogicalInstance" 'boolean t)
envSetVal("layout" "preventDeletionOfLogicalInstance" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeCycleBool

```
graphic probeCycleBool boolean { t | nil }
```

Description

Specifies whether to cycle the probe layer-purpose pairs. If cycling, the current layer-purpose pair for the probe is picked from probeHiliteLppList. The default is `t`.

GUI Equivalent

Command	*Options – Highlight*
Field	*Cycle* check box in the *Display Layer* section (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeCycleBool")
envSetVal("graphic" "probeCycleBool" 'boolean t)
envSetVal("graphic" "probeCycleBool" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeFilledInstance

```
graphic probeFilledInstance boolean { t | nil }
```

Description

Probes the content of the master symbol if the inherited connection exists in the hierarchy. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "probeFilledInstance")
envSetVal("graphic" "probeFilledInstance" 'boolean t)
envSetVal("graphic" "probeFilledInstance" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeFilledInstanceParam

```
graphic probeFilledInstanceParam int integer
```

Description

Sets the transparency level when probing. The value `0` signifies that probe is fully transparent and the value `1000` signifies that probe is not transparent.The default is `500`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "probeFilledInstanceParam")
envSetVal("graphic" "probeFilledInstanceParam" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeHighlight

```
graphic probeHighlight boolean { t | nil }
```

Description

Determines whether the probes will be drawn with probe highlight or not. The default is `t`.

GUI Equivalent

Command	*Options – Highlight*
Field	*Highlight Color* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeHighlight")
envSetVal("graphic" "probeHighlight" 'boolean t)
envSetVal("graphic" "probeHighlight" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeHiliteLppList

`graphic probeHiliteLppList string "lppList"`

Description

Enables you to specify the layer-purpose list to be used for probe. The default value is `hilite drawing hilite drawing1 hilite drawing2 hilite drawing3 hilite drawing4 hilite drawing5 hilite drawing6 hilite drawing7 hilite drawing8 hilite drawing9`.

GUI Equivalent

Command	*Options – Highlight*
Field	LPP drop-down list in the *Display Layer* section (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeHiliteLppList")
envSetVal("graphic" "probeHiliteLppList" 'string "x y z")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeHiliteLppString

`graphic probeHiliteLppString string "lpp"`

Description

Specifies the layer-purpose pair to be used by probe if probeCycleBool is set to `nil`. If probeCycleBool is `t`, `probeHiliteLppString` stores the current layer-purpose pair used by the probe.

GUI Equivalent

Command	*Options – Highlight*
Field	LPP drop-down list in the *Display Layer* section (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeHiliteLppString")
envSetVal("graphic" "probeHiliteLppString" 'string "x y z")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeInstDisplayRadio

```
graphic probeInstDisplayRadio string { instancesOnly | instancesAndNet }
layout probeInstDisplayRadio string { instancesOnly | instancesAndNet }
```

Description

Specifies whether the nets associated with probed or selected instances are also highlighted.

GUI Equivalent

Command	*Options – Highlight*
Field	*From Selected Objects (Pins, Instances, Shapes)* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeInstDisplayRadio")
envSetVal("graphic" "probeInstDisplayRadio" 'string "instancesOnly")
envSetVal("graphic" "probeInstDisplayRadio" 'string "instancesAndNet")
envGetVal("layout" "probeInstDisplayRadio")
envSetVal("layout" "probeInstDisplayRadio" 'string "instancesOnly")
envSetVal("layout" "probeInstDisplayRadio" 'string "instancesAndNet")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetDisplayCheckInstance

```
graphic probeNetDisplayCheckInstance boolean { t | nil }
```

Description

Specifies that instances on probed nets are highlighted.

GUI Equivalent

Command	*Options – Highlight*
Field	*Instances* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeNetDisplayCheckInstance")
envSetVal("graphic" "probeNetDisplayCheckInstance" 'boolean t)
envSetVal("graphic" "probeNetDisplayCheckInstance" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetDisplayCheckPins

```
graphic probeNetDisplayCheckPins boolean { t | nil }
```

Description

Specifies that pins on probed nets are also highlighted.

GUI Equivalent

Command	*Options – Highlight*
Field	*Pins* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeNetDisplayCheckPins")
envSetVal("graphic" "probeNetDisplayCheckPins" 'boolean t)
envSetVal("graphic" "probeNetDisplayCheckPins" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetDisplayFlightLines

```
graphic probeNetDisplayFlightLines string { physical | logical }
```

Description

Specifies whether the probe flightlines show physical or logical connectivity.

- `physical` specifies that all shapes and opens on the probed net are highlighted
- `logical` specifies that segments linking all pins on the probed net are highlighted.

GUI Equivalent

Command	*Options – Highlight*
Field	*Flightlines* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeNetDisplayFlightLines")
envSetVal("graphic" "probeNetDisplayFlightLines" 'string "physical")
envSetVal("graphic" "probeNetDisplayFlightLines" 'string "logical")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetDisplayHierarchyLevel

`layout probeNetDisplayHierarchyLevel int` *`integer`*

`graphic probeNetDisplayHierarchyLevel int` *`integer`*

Description

Sets the number of hierarchy levels to display when probing.

GUI Equivalent

Command	*Options – Highlight*
Field	*Hierarchy Depth* (Highlight Options Form)

Examples

```
envGetVal("layout" "probeNetDisplayHierarchyLevel")
envSetVal("layout" "probeNetDisplayHierarchyLevel" 'int 2)
envGetVal("graphic" "probeNetDisplayHierarchyLevel")
envSetVal("graphic" "probeNetDisplayHierarchyLevel" 'int 2)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetNameFilterByName

```
graphic probeNetNameFilterByName boolean { t | nil }
```

Description

Enables the selective display of probes depending on their net names. The default is `nil`, which implies all probes are displayed.

GUI Equivalent

Command	*Options – Highlight*
Form Field	*Filter net names* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeNetNameFilterByName")
envSetVal("graphic" "probeNetNameFilterByName" 'boolean t)
envSetVal("graphic" "probeNetNameFilterByName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetNameFilterByNameDefaultList

```
graphic probeNetNameFilterByNameDefaultList string "net1 net2 net3"
```

Description

Adds a default list of net names to be used for probe filtering by name, if no information is currently stored in the design. The default is `""`, which implies no filter list is available.

GUI Equivalent

Command	*Options – Highlight*
Form Field	*Net Names* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeNetNameFilterByNameDefaultList")
envSetVal("graphic" "probeNetNameFilterByNameDefaultList" 'string "netA netB")
envSetVal("graphic" "probeNetNameFilterByName" 'string "")
```

Related Topics

Highlight Options Form

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetNameFilterByNameMode

```
graphic probeNetNameFilterByNameMode boolean { t | nil }
```

Description

Controls whether the probes that have the specified nets are displayed or hidden. The default is *Hide*, this implies if the probe net is in the filtering list, it will be hidden.

GUI Equivalent

Command	*Options – Highlight*
Form Field	*Filter Mode* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeNetNameFilterByNameMode")
envSetVal("graphic" "probeNetNameFilterByNameMode" 'boolean t)
envSetVal("graphic" "probeNetNameFilterByNameMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeNetStopAtDisplayLevel

```
graphic probeNetStopAtDisplayLevel boolean { t | nil }
```

```
layout probeNetStopAtDisplayLevel boolean { t | nil }
```

Description

Specifies whether to stop the drawing of probed net at the display level. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "probeNetStopAtDisplayLevel")
envSetVal("graphic" "probeNetStopAtDisplayLevel" 'boolean t)
envSetVal("graphic" "probeNetStopAtDisplayLevel" 'boolean nil
envGetVal("layout" "probeNetStopAtDisplayLevel")
envSetVal("layout" "probeNetStopAtDisplayLevel" 'boolean t)
envSetVal("layout" "probeNetStopAtDisplayLevel" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probePromptBusBit

```
graphic probePromptBusBit boolean { t | nil }
```

Description

Specifies whether to display the prompt that lets you select the specific bit of a bus when the `geAddNetProbe()` function is called. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "probePromptBusBit")
envSetVal("graphic" "probePromptBusBit" 'boolean t)
envSetVal("graphic" "probePromptBusBit" 'boolean nil
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeShapeDisplay

```
graphic probeShapeDisplay string { shapesOnly | shapesAndNet }
```

Description

Specifies whether the nets associated with probed shapes are also highlighted.

GUI Equivalent

Command	*Options – Highlight*
Field	*From Selected Objects (Pins, Instances, Shapes)* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeShapeDisplay")
envSetVal("graphic" "probeShapeDisplay" 'string "shapesOnly")
envSetVal("graphic" "probeShapeDisplay" 'string "shapesAndNet")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeTermDisplayRadio

```
graphic probeTermDisplayRadio string { pinOnly | pinsAndNet }
```

Description

Specifies whether the nets associated with probed pins are also highlighted.

GUI Equivalent

Command	*Options – Highlight*
Field	*From Selected Objects (Pins, Instances, Shapes)* (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeTermDisplayRadio")
envSetVal("graphic" "probeTermDisplayRadio" 'string "pinOnly")
envSetVal("graphic" "probeTermDisplayRadio" 'string "pinsAndNet")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

probeVisibleLpp

```
graphic probeVisibleLpp boolean { t | nil }
```

Description

Specifies that only shapes and pins on the visible layer-purpose pairs are highlighted. The default is `nil`.

GUI Equivalent

Command	*Options – Highlight*
Field	Visible Shapes Only (Highlight Options Form)

Examples

```
envGetVal("graphic" "probeVisibleLpp")
envSetVal("graphic" "probeVisibleLpp" 'boolean t)
envSetVal("graphic" "probeVisibleLpp" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

propEditHideTopSection

```
layout propEditHideTopSection boolean { t | nil }
```

Description

Setting this environment variable to `nil` to unhide the top section of the Edit Properties form. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "propEditHideTopSection")
envSetVal("layout" "propEditHideTopSection" 'boolean t)
envSetVal("layout" "propEditHideTopSection" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

propEditorHeight

`layout propEditorHeight int` *`any_integer_greater_than_zero`*

Description

Specifies the height of the Property Editor form.The default is `320`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "propEditorHeight")
envSetVal("layout" "propEditorHeight" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

propEditorWidth

`layout propEditorWidth int` *`any_integer_greater_than_zero`*

Description

Specifies the width of the Property Editor form.The default is `575`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "propEditorWidth")
envSetVal("layout" "propEditorWidth" 'int 200)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

propertyEditorAutoResize

```
layout propertyEditorAutoResize boolean { t | nil }
```

Description

Setting this environment variable to `t` means that the *Property Editor* is automatically resized. When set to `nil`, the *Property Editor* can be resized. It remains fixed for all object types in the current session. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "propertyEditorAutoResize")
envSetVal("layout" "propertyEditorAutoResize" 'boolean t)
envSetVal("layout" "propertyEditorAutoResize" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

propEditStretchableTree

```
layout propEditStretchableTree boolean { t | nil }
```

Description

Enables the selection tree in the *Property Editor* to be stretchable. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "propEditStretchableTree")
envSetVal("layout" "propEditStretchableTree" 'boolean t)
envSetVal("layout" "propEditStretchableTree" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

protectedObjHighlightHaloLevel

```
graphic protectedObjHighlightHaloLevel string { over | under }
```

Description

Determines the placement of the halo relative to the object boundary box.The default is `under`.

GUI Equivalent

Command	*Edit – Select – Selection Protection Options*
Form Field	*Level* ()

Examples

```
envGetVal("graphic" "protectedObjHighlightHaloLevel")
envSetVal("graphic" "protectedObjHighlightHaloLevel" 'string "over")
envSetVal("graphic" "protectedObjHighlightHaloLevel" 'string "under")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

protectedObjHighlightHaloMode

```
graphic protectedObjHighlightHaloMode string { normal | fadeout | glow | plain |
     random }
```

Description

Determines the placement of the halo relative to the object boundary box.The default is `fadeout`.

GUI Equivalent

Command	*Edit – Select – Selection Protection Options*
Form Field	*Halo Mode* (Selection Protection Highlight Options Form)

Examples

```
envGetVal("graphic" "protectedObjHighlightHaloMode")
envSetVal("graphic" "protectedObjHighlightHaloMode" 'string "normal")
envSetVal("graphic" "protectedObjHighlightHaloMode" 'string "fadeout")
envSetVal("graphic" "protectedObjHighlightHaloMode" 'string "glow")
envSetVal("graphic" "protectedObjHighlightHaloMode" 'string "plain")
envSetVal("graphic" "protectedObjHighlightHaloMode" 'string "random")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

protectedObjHighlightHaloTrans

`graphic protectedObjHighlightHaloTrans int` *`integer_number`*

Description

Controls the visibility of the halo. The visibility of the halo is the maximum at the lowest transparency setting. The default is `40`.

GUI Equivalent

Command	*Edit – Select – Selection Protection Options*
Form Field	*Transparency* (Selection Protection Highlight Options Form)

Examples

```
envGetVal("graphic" "protectedObjHighlightHaloTrans")
envSetVal("graphic" "protectedObjHighlightHaloTrans" 'int 60)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

protectedObjHighlightHaloWidth

```
graphic protectedObjHighlightHaloWidth string { thick | normal | thin }
```

Description

Specifies the thickness of the halo on a protected object. The default is `normal`.

GUI Equivalent

Command	*Edit – Select – Selection Protection Options*
Form Field	*Width* (Selection Protection Highlight Options Form)

Examples

```
envGetVal("graphic" "protectedObjHighlightHaloWidth")
envSetVal("graphic" "protectedObjHighlightHaloWidth" 'string "thick")
envSetVal("graphic" "protectedObjHighlightHaloWidth" 'string "normal")
envSetVal("graphic" "protectedObjHighlightHaloWidth" 'string "thin")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteAllLayersScope

```
layout pteAllLayersScope cyclic { lswLayers | techfileLayers | nil }
```

Description

Defines the scope of layer-purpose pairs to be displayed in the *Layers* panel. When set to `lswLayers`, only the layer-purpose pairs defined in the LSW information file are displayed in the *Layers* panel. When set to `techfileLayers`, only the layer-purpose pairs defined in the technology file are displayed in the *Layers* panel.

Default is `lswLayers`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Scope of 'All Layers' Layer Set* (Palette Options Form)

Examples

```
envGetVal("layout" "pteAllLayersScope")
envSetVal("layout" "pteAllLayersScope" 'cyclic "lswLayers")
envSetVal("layout" "pteAllLayersScope" 'cyclic "techfileLayers")
envSetVal("layout" "pteAllLayersScope" 'cyclic "nil")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteApplyParentState

```
layout pteApplyParentState boolean { t | nil }
```

Description

Toggles between the synchronized and desynchronized states of parent and child objects in the *Objects* and *Grids* panels.

By default, the environment variable is set to `t`, which is the synchronized mode. In this mode, when the visibility or selectability of a parent object is changed, the visibility and selectability of child objects is reset to coincide with that of the parent object.

To switch to the desynchronized mode, set the environment variable to `nil`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Visibility and selectability reset by parent*
	Sync visibility/selectability by parent (Palette Options Form)

Examples

```
envGetVal("layout" "pteApplyParentState")
envSetVal("layout" "pteApplyParentState" 'boolean t)
envSetVal("layout" "pteApplyParentState" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteAutoRedrawMode

```
layout pteAutoRedrawMode cyclic { currentWindow | noWindow | allWindows }
```

Description

Determines the scope of the auto redraw feature.

When set to `currentWindow`, the current window canvas is updated in real time whenever a change is made to a Palette panel.

When set to `noWindow`, the auto redraw feature is turned off. As a result, manual redraw is needed to update all windows.

When set to `allWindows`, all synchronized window canvases are updated in real time, whenever a change is made to a Palette panel.

The default value is `currentWindow`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Auto Redraw* (Palette Options Form)

Examples

```
envGetVal("layout" "pteAutoRedrawMode")
envSetVal("layout" "pteAutoRedrawMode" 'cyclic "currentWindow")
envSetVal("layout" "pteAutoRedrawMode" 'cyclic "noWindow")
envSetVal("layout" "pteAutoRedrawMode" 'cyclic "allWindows")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteCheckableColumnClickMode

```
layout pteCheckableColumnClickMode cyclic { turnOnOff | sort }
```

Description

Determines the behavior when the Validity (*V*), Selectability (*S*), or Membership (*m*) columns in the *Layers* panel is clicked.

When set to `turnOnOff`, the statuses toggle between the on and off modes.

When set to `sort`, the layers are sorted based on their validity, selectability, or membership statuses.

The default value is `turnOnOff`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Checkable Column Click*
	Header Column Click (Palette Options Form)

Examples

```
envGetVal("layout" "pteCheckableColumnClickMode")
envSetVal("layout" "pteCheckableColumnClickMode" 'cyclic "turnOnOff")
envSetVal("layout" "pteCheckableColumnClickMode" 'cyclic "sort")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteCollapseLayersInViewBy

```
layout pteCollapseLayersInViewBy boolean { t | nil }
```

Description

Collapses the *Layers* panel when you select the *View By* options. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteCollapseLayersInViewBy")
envSetVal("layout" "pteCollapseLayersInViewBy" 'boolean t)
envSetVal("layout" "pteCollapseLayersInViewBy" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteColoredLppSortOrder

`layout pteColoredLppSortOrder string` *`string_value`*

Description

Specifies the order in which colored layer-purpose pairs are listed within each master layer-purpose pair grouping. The default value is `""`.

You can use a space, a comma (`,`), a semicolon (`;`), a hyphen (`-`), a forward slash (`/`), a period (`.`), or an underscore (_) to separate the values in *`string_value`*.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteColoredLppSortOrder")
envSetVal("layout" "pteColoredLppSortOrder" 'string "metal1 drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteColorMixedSort

```
layout pteColorMixedSort boolean { t | nil }
```

Description

Causes the *Color mixed sort* feature to be enabled or disabled. The default value is `nil`, which causes layer-purpose pairs to be sorted into two distinct groups, colored and uncolored.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Color mixed sort* (Palette Options Form)

Examples

```
envGetVal("layout" "pteColorMixedSort")
envSetVal("layout" "pteColorMixedSort" 'boolean t)
envSetVal("layout" "pteColorMixedSort" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteCompactMPT

```
layout pteCompactMPT boolean { t | nil }
```

Description

Controls the setting of the *Compact MPT mode* check box available in the Palette Options form. If set to `t`, *MPT Support* opens in compact mode in a new layout window. The default value is `nil`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Compact MPT mode* (Palette Options Form)

Examples

```
envGetVal("layout" "pteCompactMPT")
envSetVal("layout" "pteCompactMPT" 'boolean t)
envSetVal("layout" "pteCompactMPT" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteDeferRedrawOnMouseMove

```
layout pteDeferRedrawOnMouseMove boolean { t | nil }
```

Description

Stops canvas redraw when there is MMB and mouse movement on the *Layers* panel. Only one redraw is done when the mouse is released. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteDeferRedrawOnMouseMove")
envSetVal("layout" "pteDeferRedrawOnMouseMove" 'boolean t)
envSetVal("layout" "pteDeferRedrawOnMouseMove" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteDisplayActiveLppInForeground

```
layout pteDisplayActiveLppInForeground boolean { t | nil }
```

Description

Causes the *Display Active LPP in Foreground* check box to be selected or deselected. The default value is `nil`.

GUI Equivalent

Command	*Options – Editor*
Field	*Display Active LPP in Foreground* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "pteDisplayActiveLppInForeground")
envSetVal("layout" "pteDisplayActiveLppInForeground" 'boolean t)
envSetVal("layout" "pteDisplayActiveLppInForeground" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteEnableMouseBindings

```
layout pteEnableMouseBindings boolean { t | nil }
```

Description

Causes a right-click in the *Layers* panel to display the context-sensitive Palette menu if set to `nil`. If set to `t`, a right-click in the *Layers* panel turns on or off the selectability of a layer-purpose pair. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteEnableMouseBindings")
envSetVal("layout" "pteEnableMouseBindings" 'boolean t)
envSetVal("layout" "pteEnableMouseBindings" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteFindModeOn

```
layout pteFindModeOn boolean { t | nil }
```

Description

Toggles between the *Filter* and *Find* search modes.

When set to `t`, the *Find* mode is switched on. When set to `nil`, the *Filter* mode is switched on.

The default value is `nil`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Search Mode* (Palette Options Form)

Examples

```
envGetVal("layout" "pteFindModeOn")
envSetVal("layout" "pteFindModeOn" 'boolean t)
envSetVal("layout" "pteFindModeOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteIncludeCutLayers

```
layout pteIncludeCutLayers boolean { t | nil }
```

Description

Controls the inclusion and exclusion of cut layers as routing layers. When set to `t`, cut layers are included as routing layers. When set to `nil`, cut layers are excluded from routing layers. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteIncludeCutLayers")
envSetVal("layout" "pteIncludeCutLayers" 'boolean t)
envSetVal("layout" "pteIncludeCutLayers" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteInfraDefValueObjGrids

`layout pteInfraDefValueObjGrids string` *`any_string`*

Description

Sets the visibility and selectability of objects and grids. The default value is `""`.

Note: This environment variable does not take effect if typed in CIW in your current session. Exit and then restart a new virtuoso session after setting this environment variable in the `.cdsinit/.cdsenv` file.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteInfraDefValueObjGrids")
envSetVal("layout" "pteInfraDefValueObjGrids" 'string "Snap_Patterns nil")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteInfraObjSelFollowVis

`layout pteInfraObjSelFollowVis string` *`any_string`*

Description

Specifies the list of objects for which selectability follows visibility. The default value is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteInfraObjSelFollowVis")
envSetVal("layout" "pteInfraObjSelFollowVis" 'string "Snap_Patterns")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteInfraWirebondProfileColors

`layout pteInfraWirebondProfileColors string` *`wirebondProfileColors`*

Description

(ICADVM20.1 Virtuoso RF Solution Only) Specifies the list of colors in which wirebond profiles are displayed on the canvas. The default value is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteInfraWirebondProfileColors")
envSetVal("layout" "pteInfraWirebondProfileColors" 'string "blue magenta brown")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteInverseColorScheme

```
layout pteInverseColorScheme boolean { t | nil }
```

Description

When set to `t`, inverses the color scheme of the Palette panels. The default value is `nil`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Inverse color scheme* (Palette Options Form)

Examples

```
envGetVal("layout" "pteInverseColorScheme")
envSetVal("layout" "pteInverseColorScheme" 'boolean t)
envSetVal("layout" "pteInverseColorScheme" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLayerSetActivationMode

```
layout pteLayerSetActivationMode cyclic { filterOnly | filterAndApply |
     filterVisSelectability }
```

Description

Determines the default action when one or more layer sets are selected.

When set to `filterOnly`, displays all layer-purpose pairs present in the selected layer sets. The visibility and selectability status of the layer-purpose pairs remains unchanged.

When set to `filterAndApply`, displays all layer-purpose pairs present in the selected layer sets. The visibility and selectability status of layer-purpose pairs is reset to that defined in the layer set files.

When set to `filterVisSelectability`, displays all layer-purpose pairs present in the selected layer sets. The visibility and selectability of all layer-purpose pairs is turned on.

The default value is `filterOnly`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Layer Set Activation* (Palette Options Form)

Examples

```
envGetVal("layout" "pteLayerSetActivationMode")
envSetVal("layout" "pteLayerSetActivationMode" 'cyclic "filterOnly")
envSetVal("layout" "pteLayerSetActivationMode" 'cyclic "filterAndApply")
envSetVal("layout" "pteLayerSetActivationMode" 'cyclic "filterVisSelectability")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLayerSetManagerLeftClickMode

```
layout pteLayerSetManagerLeftClickMode cyclic { activateLayerSet |
     forceVisibilityAndSelectabilityOnly }
```

Description

Determines the default left mouse button click behavior for layer sets.

When set to `activateLayerSet`, the selected layer is activated. The visibility and selectability statuses for the selected layer are automatically turned on.

When set to `forceVisibilityAndSelectabilityOnly`, the selected layer set is activated and the visibility and selectability statuses of the constituent layers is forcibly turned on. The visibility and selectability statuses of all other layers are forcibly turned off.

The default value is `activateLayerSet`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Layer Set Manager Left Click* (Palette Options Form)

Examples

```
envGetVal("layout" "pteLayerSetManagerLeftClickMode")
envSetVal("layout" "pteLayerSetManagerLeftClickMode" 'cyclic "activateLayerSet")
envSetVal("layout" "pteLayerSetManagerLeftClickMode" 'cyclic
"forceVisibilityAndSelectabilityOnly")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLayersSelectabilityDepth

`layout pteLayersSelectabilityDepth int` *`integer_value`*

Description

Sets the selectability depth for layers. The depth value determines the number of layer-purpose pairs for which the selectability attribute is affected when you hold down `Ctrl` or both `Ctrl` and `Shift` and right-click a layer-purpose pair in the *Layers* panel. The default value is `1`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Selectability Depth For Layers*
	Depth for Selectability (Palette Options Form)

Examples

```
envGetVal("layout" "pteLayersSelectabilityDepth")
envSetVal("layout" "pteLayersSelectabilityDepth" 'int 3)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLayersVisibilityDepth

`layout pteLayersVisibilityDepth int` *`integer_value`*

Description

Sets the visibility depth for layers. The depth value determines the number of layer-purpose pairs for which the visibility attribute is affected when you hold down `Ctrl` or both `Ctrl` and `Shift` and click in the *Layers* panel a layer-purpose pair with the middle mouse button. The default value is `1`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Visibility Depth For Layers*
	Depth for Visibility (Palette Options Form)

Examples

```
envGetVal("layout" "pteLayersVisibilityDepth")
envSetVal("layout" "pteLayersVisibilityDepth" 'int 3)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLinkColorVisibilityToShapeVisibility

```
layout pteLinkColorVisibilityToShapeVisibility boolean { t | nil }
```

Description

Controls the visibility of colored shapes. When set to `t`, colored shapes become invisible if you deselect the Visibility (*V*) check box corresponding to the layer-purpose pairs on which they are drawn. When set to `nil`, only the colored outline is hidden; the shapes remain visible.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Shape visibility by color visibility* (Palette Options Form)

Examples

```
envGetVal("layout" "pteLinkColorVisibilityToShapeVisibility")
envSetVal("layout" "pteLinkColorVisibilityToShapeVisibility" 'boolean t)
envSetVal("layout" "pteLinkColorVisibilityToShapeVisibility" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLppGroup

`layout pteLppGroup string` *`purposeNames`*

Description

Enables you to group layers by the specified purposes.

GUI Equivalent

Command	*Palette Menu (Layers)*
Form Field	*View By – LPP Group*

Examples

```
envGetVal("layout" "pteLppGroup")
envSetVal("layout" "pteLppGroup" 'string "drawing label")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLppsDisplayOrderControl

```
layout pteLppsDisplayOrderControl cyclic { palette | techfile }
```

Description

Causes the *Palette LPPs Display Order* check box to be selected or deselected. When set to `techfile`, the default value, the check box is deselected, and when set to `palette`, the check box is selected.

When this environment variable is set to `palette`, you can change the display order of layer-purpose pairs in the canvas by dragging up and down the layer-purpose pairs listed in the Palette.

Note: Palette must be set to *Multi-select* mode.

GUI Equivalent

Command	*Options - Layout Editor Options*
	Options – Editor
Field	*Palette LPPs Display Order* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "pteLppsDisplayOrderControl")
envSetVal("layout" "pteLppsDisplayOrderControl" 'cyclic "palette")
envSetVal("layout" "pteLppsDisplayOrderControl" 'cyclic "techfile")
```

Related Topics

List of Layout Editor Environment Variables'

© 2020 All Rights Reserved.

pteLSManagerAsAssistant

```
layout pteLSManagerAsAssistant boolean { t | nil }
```

Description

Enables the layer set manager as an assistant. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteLSManagerAsAssistant")
envSetVal("layout" "pteLSManagerAsAssistant" 'boolean t)
envSetVal("layout" "pteLSManagerAsAssistant" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteLSManagerRespectFilters

```
layout pteLSManagerRespectFilters boolean { t | nil }
```

Description

When set to `t`, the *+* and – buttons in the *Layer Set Manager*, which are used to turn on and turn off the visibility or selectability status of the layers present in a layer set, work with respect to the filter settings defined on the *Scope* and *Filter* toolbars. The *Scope* toolbar provides three options for filtering layers, namely, *Valid*, *Used*, and *Routing*. The *Filter* toolbar enables you to define search criteria for filtering layers.

For example, if you select the *Valid* check box on the *Scope* toolbar, and click the *+* button in the *Layer Set Manager* to turn on the visibility of a layer set, then visibility is turned on for only the valid layers of the layer set. It does not affect the visibility setting of the invalid layers.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteLSManagerRespectFilters")
envSetVal("layout" "pteLSManagerRespectFilters" 'boolean t)
envSetVal("layout" "pteLSManagerRespectFilters" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteMPTMode

```
layout pteMPTMode boolean { t | nil }
```

Description

Determines if the Palette MPT feature is turned on or off at the time of opening a new layout window.

If set to `nil`, which is the default, the Palette MPT feature is disabled. If set to `t`, the *MPT* check box is displayed on the *Scope* toolbar and the Color (*C*) column is displayed in the *Layers* panel when you open a new layout window.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteMPTMode")
envSetVal("layout" "pteMPTMode" 'boolean t)
envSetVal("layout" "pteMPTMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteMPTScope

```
layout pteMPTScope boolean { t | nil }
```

Description

Controls the setting of the *MPT* check box at the time of opening a new layout window and is evaluated only if the pteMPTMode environment variable is set to `t`.

If set to `nil`, which is the default, the MPT check box is deselected and all layer-purpose pairs are listed in the in the *Layers* panel. If set to `t`, the *MPT* check box is selected and only the layer-purpose pairs with more than one color are listed in the *Layers* panel.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteMPTScope")
envSetVal("layout" "pteMPTScope" 'boolean t)
envSetVal("layout" "pteMPTScope" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteMovableTools

```
layout pteMovableTools boolean { t | nil }
```

Description

Specifies whether the toolbars can be moved in the workspace. The default value is `nil`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Movable tools* (Palette Options Form)

Examples

```
envGetVal("layout" "pteMovableTools")
envSetVal("layout" "pteMovableTools" 'boolean t)
envSetVal("layout" "pteMovableTools" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteMultiSelectionMode

```
layout pteMultiSelectionMode boolean { t | nil }
```

Description

Toggles between the single-selection and multi-selection mode. When set to `t`, multi-selection mode is turned on. When set to `nil`, single-section mode is turned on. The default value is `nil`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Multi-select mode* (Palette Options Form)

Examples

```
envGetVal("layout" "pteMultiSelectionMode")
envSetVal("layout" "pteMultiSelectionMode" 'boolean t)
envSetVal("layout" "pteMultiSelectionMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteRetainFilterSettings

```
layout pteRetainFilterSettings boolean { t | nil }
```

Description

When set to `t`, synchronizes the setting of the *Scope* filters, namely *Valid*, *Used*, *Routing*, and *MPT*, across synchronized layout windows.

For example, if you apply the *Routing* filter by selecting the *Routing* check box in one layout window, it is automatically applied to all open layout windows, and also to those that are opened subsequently, as long as the windows stay synchronized.

The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteRetainFilterSettings")
envSetVal("layout" "pteRetainFilterSettings" 'boolean t)
envSetVal("layout" "pteRetainFilterSettings" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteRoutingOnlyScope

```
layout pteRoutingOnlyScope boolean { t | nil }
```

Description

When set to `t`, only routing layers are displayed in the *Layers* panel. The default value is `nil`.

GUI Equivalent

Command	*Layers Panel*
Field	*Routing*

Examples

```
envGetVal("layout" "pteRoutingOnlyScope")
envSetVal("layout" "pteRoutingOnlyScope" 'boolean t)
envSetVal("layout" "pteRoutingOnlyScope" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteShowColorControlsForMPT

```
layout pteShowColorControlsForMPT boolean { t | nil }
```

Description

Displays the two check boxes that control the visibility of colored shapes in compact mode if set to `t`. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteShowColorControlsForMPT")
envSetVal("layout" "pteShowColorControlsForMPT" 'boolean t)
envSetVal("layout" "pteShowColorControlsForMPT" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteShowMPTScope

```
layout pteShowMPTScope boolean { t | nil }
```

Description

Hides the *MPT* check box if set to `nil`. The default value is `t`.

The *MPT* check box is displayed on the *Scope* toolbar when you enable *MPT Support*.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteShowMPTScope")
envSetVal("layout" "pteShowMPTScope" 'boolean t)
envSetVal("layout" "pteShowMPTScope" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteShowUsedSystemLpps

```
layout pteShowUsedSystemLpps boolean { t | nil }
```

Description

When set to `t`, on the application of the *Used* filter, the *Layers* panel displays layers containing all types of design objects. The default value is `nil`, which implies that, by default, the *Layers* panel displays only the layers containing design objects of type shape.

GUI Equivalent

None

Examples

```
envGetVal("layout" "pteShowUsedSystemLpps")
envSetVal("layout" "pteShowUsedSystemLpps" 'boolean t)
envSetVal("layout" "pteShowUsedSystemLpps" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteUsedLayersEipScope

```
layout pteUsedLayersEipScope cyclic { currentToBottom | topToBottom }
```

Description

When set to `currentToBottom`, layers used in the design at the current hierarchy level and below are listed in the *Layers* panel. When set to `topToBottom`, all layers used in the design are listed in the *Layers* panel. The default is `currentToBottom`.

GUI Equivalent

Command	*Palette Menu – Options*
Field	*Scope of Used Layers in EIP* (Palette Options Form)

Examples

```
envGetVal("layout" "pteUsedLayersEipScope")
envSetVal("layout" "pteUsedLayersEipScope" 'cyclic "currentToBottom")
envSetVal("layout" "pteUsedLayersEipScope" 'cyclic "topToBottom")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

pteWindowSynchronizationMode

```
layout pteWindowSynchronizationMode cyclic { independent | synchronized }
```

Description

Sets the synchronization mode for new windows to either independent or synchronized. The default value is synchronized.

GUI Equivalent

Command	*Palette Menu – Options*
Field	Window Synchronization
	Keep windows synchronized (Palette Options Form)

Examples

```
envGetVal("layout" "pteWindowSynchronizationMode")
envSetVal("layout" "pteWindowSynchronizationMode" 'cyclic "independent")
envSetVal("layout" "pteWindowSynchronizationMode" 'cyclic "synchronized")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

QAHoldTimeMilliSec

`layout QAHoldTimeMilliSec int` *`integer_number`*

Description

Enables you to specify the time (in milliseconds) required to hold the pointer over an object. While using the *Quick Align* command, if the pointer remains stationary over an object for longer than the specified time, the potential points, edges, and centerlines in the object are highlighted. The default value is `0`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "QAHoldTimeMilliSec")
envSetVal("layout" "QAHoldTimeMilliSec" 'int 100)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

QAMaxPixelsPerMilliSec

`layout QAMaxPixelsPerMilliSec int` *`integer_number`*

Description

Defines the maximum velocity with which the pointer should move between potential targets for the *Quick Align* command to be able to snap to them. If the pointer moves with the velocity higher than the value defined in the `QAMaxPixelsPerMilliSec` environment variable, the *Quick Align* command does not snap to those targets.

GUI Equivalent

None

Examples

```
envGetVal("layout" "QAMaxPixelsPerMilliSec")
envSetVal("layout" "QAMaxPixelsPerMilliSec" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

quickAlignMode

```
layout quickAlignMode cyclic { MoveOrStretch | Copy }
```

Description

Specifies the mode to be used for alignment.

- `MoveOrStretch`: The object is moved or stretched during alignment.
- `Copy`: The object is copied during alignment.

The default value is `MoveOrStretch`.

GUI Equivalent

Command	*Edit – Quick Align*
Field	*Move/Stretch, Copy*
	Copy reference (Quick Align Form)

Examples

```
envGetVal("layout" "quickAlignMode")
envSetVal("layout" "quickAlignMode" 'cyclic "MoveOrStretch")
envSetVal("layout" "quickAlignMode" 'cyclic "Copy")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

quickAlignSameLayerTarget

```
layout quickAlignSameLayerTarget boolean { t | nil }
```

Description

Snaps the pointer to only those targets that are on the same layer as the object with the reference point, edge, or centerline. When set to `nil`, the pointer snaps to edges, centerlines, and points on a target object on any layer if they are within the aperture distance from the pointer. The default value is `nil`.

GUI Equivalent

Command	*Edit – Quick Align*
Field	*On Same Layer* (Quick Align Form)

Examples

```
envGetVal("layout" "quickAlignSameLayerTarget")
envSetVal("layout" "quickAlignSameLayerTarget" 'boolean t)
envSetVal("layout" "quickAlignSameLayerTarget" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

quickAlignSnapTarget

```
layout quickAlignSnapTarget cyclic { Edges and Points | Edges | Points }
```

Description

Specifies the method to enable the pointer to the reference set and target object.

- `Edges and Points`: Snaps the pointer to edges, centerlines, vertices, nexus, and end points in the reference set and target object.
- `Edges`: Snaps the pointer to only the edges and centerlines in the reference set and target object.
- `Points`: Snaps the pointer to vertices, midpoint of edges, nexus, end points, and origin point of objects in reference set and target.

The default value is `Edges and Points`.

GUI Equivalent

Command	*Edit – Quick Align*
Field	*Snap Target* (Quick Align Form)

Examples

```
envGetVal("layout" "quickAlignSnapTarget")
envSetVal("layout" "quickAlignSnapTarget" 'cyclic "Edges and Points")
envSetVal("layout" "quickAlignSnapTarget" 'cyclic "Edges")
envSetVal("layout" "quickAlignSnapTarget" 'cyclic "Points")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

quickAlignSpacingType

```
layout quickAlignSpacingType cyclic { none | min | user }
```

Description

Specifies the spacing between the reference edge, centerline, or point and the target edge, centerline, or point after the alignment.

- `none`: No spacing is applied.
- `min`: The `minSpacing` value of the reference object layer is applied.
- `user`: A user-specified spacing value is applied. Specify the value by using the environment variable quickAlignUserSpacing.

The default value is `none`.

GUI Equivalent

Command	*Edit – Quick Align*
Field	*No Spacing, Min Spacing, User Spacing* (Quick Align Form)

Examples

```
envGetVal("layout" "quickAlignSpacingType")
envSetVal("layout" "quickAlignSpacingType" 'cyclic "none")
envSetVal("layout" "quickAlignSpacingType" 'cyclic "min")
envSetVal("layout" "quickAlignSpacingType" 'cyclic "user")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

quickAlignUserSpacing

`layout quickAlignUserSpacing float` *`float_number`*

Description

Specifies the user-defined spacing value between the reference edge, centerline, or point and the target edge, centerline, or point after the alignment. The default value is `0`. You can specify an integer.

GUI Equivalent

Command	*Edit – Quick Align*
Field	*User Spacing* (Quick Align Form)

Examples

```
envGetVal("layout" "quickAlignUserSpacing")
envSetVal("layout" "quickAlignUserSpacing" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

realColorSelection

```
graphic realColorSelection boolean { t | nil }
```

Description

Specifies the display of selected objects in their original colors when the design or other objects in the design are set to be dimmed.

GUI Equivalent

Command	*Options – Display*
Field	*True Color Selection only* (Display Options Form)

Examples

```
envGetVal("graphic" "realColorSelection")
envSetVal("graphic" "realColorSelection" 'boolean t)
envSetVal("graphic" "realColorSelection" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

realignLabelToPinCenter

```
layout realignLabelToPinCenter boolean { t | nil }
```

Description

Determines whether you want to realign the label to the center of the pin (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "realignLabelToPinCenter")
envSetVal("layout" "realignLabelToPinCenter" 'boolean t)
envSetVal("layout" "realignLabelToPinCenter" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectangularWSPPinAsSquare

```
layout rectangularWSPPinAsSquare boolean { t | nil }
```

Description

Determines whether you want to create a square pin (`t`) or not (`nil`). The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "rectangularWSPPinAsSquare")
envSetVal("layout" "rectangularWSPPinAsSquare" 'boolean t)
envSetVal("layout" "rectangularWSPPinAsSquare" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectBaseMode

```
layout rectBaseMode cyclic { Bottom left | Bottom right | Top left | Top right |
     Center}
```

Description

Specifies the base creation point of rectangle that you create: `Bottom left`, `Bottom right`, `Top left`, `Top right`, `Center`. The default value is `Bottom left`.

GUI Equivalent

Command	*Create – Shape – Rectangle*
Field	*Justification* (Create Rectangle Form)

Examples

```
envGetVal("layout" "rectBaseMode")
envSetVal("layout" "rectBaseMode" 'cyclic "Bottom left")
envSetVal("layout" "rectBaseMode" 'cyclic "Bottom right")
envSetVal("layout" "rectBaseMode" 'cyclic "Top left")
envSetVal("layout" "rectBaseMode" 'cyclic "Top right")
envSetVal("layout" "rectBaseMode" 'cyclic "Center")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectHeight

`layout rectHeight float` *`float_number`*

Description

Specifies the height of the rectangle. The default value is `0.0`.

GUI Equivalent

Command	*Create – Shape – Rectangle*
Field	*Size: Height* (Create Rectangle Form)

Examples

```
envGetVal("layout" "rectHeight")
envSetVal("layout" "rectHeight" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectOnSnapPatternGrid

```
layout rectOnSnapPatternGrid boolean { t | nil }
```

Description

When enabled, the first point of the rectangle snaps correctly to the appropriate track enclosure line. This implies when you create a rectangle downwards in the cellview, the first point snaps to the upper track enclosure line. When you create a rectangle upwards in the cellview, the first point snaps to lower track enclosure line. To specify the second point correctly, you are guided by the DRD discrete dimension markers. The discrete dimension markers appear only if discrete dimension is enabled and the `allowedWidthRanges` constraint is specified in the technology file. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "rectOnSnapPatternGrid")
envSetVal("layout" "rectOnSnapPatternGrid" 'boolean t)
envSetVal("layout" "rectOnSnapPatternGrid" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectSmartSnapEnable

```
layout rectSmartSnapEnable boolean { t | nil }
```

Description

Specifies whether the rectangle is to be created with smart snapping. The default value is `nil`.

GUI Equivalent	
Command	*Create – Shape – Rectangle*
Form Field	*Enable smart snapping* (Create Rectangle Form)

Examples

```
envGetVal("layout" "rectSmartSnapEnable")
envSetVal("layout" "rectSmartSnapEnable" 'boolean t)
envSetVal("layout" "rectSmartSnapEnable" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectSmartSnapOnActiveLayer

```
layout rectSmartSnapOnActiveLayer boolean { t | nil }
```

Description

Specifies whether the rectangle is to be created with smart snapping on current active layer only. The default value is `nil`.

GUI Equivalent

Command	*Create – Shape – Rectangle*
Field	*Active layer only* (Create Rectangle Form)

Examples

```
envGetVal("layout" "rectSmartSnapOnActiveLayer")
envSetVal("layout" "rectSmartSnapOnActiveLayer" 'boolean t)
envSetVal("layout" "rectSmartSnapOnActiveLayer" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rectWidth

`layout rectWidth float` *`float_number`*

Description

Specifies the width of the rectangle. The default value is `0.0`.

GUI Equivalent

Command	*Create – Shape – Rectangle*
Field	*Size: Width* (Create Rectangle Form)

Examples

```
envGetVal("layout" "rectWidth")
envSetVal("layout" "rectWidth" 'float 6.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

recursionCheck

```
layout recursionCheck boolean { t | nil }
```

Description

Specifies whether to check for recursive hierarchy when creating instances (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Recursion Check* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "recursionCheck")
envSetVal("layout" "recursionCheck" 'boolean t)
envSetVal("layout" "recursionCheck" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstCheckTerms

```
layout remasterInstCheckTerms boolean { t | nil }
```

Description

If `t`, checks the terminals of cellviews in the specified libraries for matching or a subset of matching terminals. The default is `nil`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Check Terminals* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstCheckTerms")
envSetVal("layout" "remasterInstCheckTerms" 'boolean t)
envSetVal("layout" "remasterInstCheckTerms" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstSearchCellText

```
layout remasterInstSearchCellText string { any_cell_name }
```

Description

Specifies the name of the cell that contains the instances you want to use to re-master the instances specified by `remasterInstUpdateCellText`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Search for: CellName* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstSearchCellText")
envSetVal("layout" "remasterInstSearchCellText" 'string "MyCell")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstSearchLibText

```
layout remasterInstSearchLibText string { any_library_name }
```

Description

Specifies the name of the library which contains the cellviews you want to use to re-master the instances specified by `remasterInstUpdateLibText`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Search for: ViewName* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstSearchViewText")
envSetVal("layout" "remasterInstSearchViewText" 'string "MyLib")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstSearchViewText

```
layout remasterInstSearchViewText string { any_view_name }
```

Description

Specifies the name of the view you want to use to re-master the instances specified by `remasterInstUpdateViewText`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Search for: ViewName* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstSearchViewText")
envSetVal("layout" "remasterInstSearchViewText" 'string "MyView")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstUpdateCellText

```
layout remasterInstUpdateCellText string { any_cell_name }
```

Description

Specifies the name of the cellview that you want updated with instances specified by `remasterInstSearchLibText`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Update to: CellName* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstUpdateCellText")
envSetVal("layout" "remasterInstUpdateCellText" 'string "MyCell")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstUpdateLibText

```
layout remasterInstUpdateLibText string { any_library_name }
```

Description

Specifies the name of the library that you want updated with instances specified by `remasterInstSearchCellText`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Update to: Library* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstUpdateLibText")
envSetVal("layout" "remasterInstUpdateLibText" 'string "MyLib")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

remasterInstUpdateViewText

```
layout remasterInstUpdateViewText string { any_cell_name }
```

Description

Specifies the name of the view that you want updated with instances specified by `remasterInstSearchViewText`.

GUI Equivalent

Command	*Tools – Remaster Instances*
Field	*Update to: ViewName* (Remaster Instances Form)

Examples

```
envGetVal("layout" "remasterInstUpdateViewText")
envSetVal("layout" "remasterInstUpdateViewText" 'string "MyCell")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

removeChopShape

```
layout removeChopShape boolean { t | nil }
```

Description

Specifies whether the area cut by `leHiChop` or `leChopShape` should be removed (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

Command	*Edit – Basic – Chop*
Field	*Remove chop* (Chop Form)

Examples

```
envGetVal("layout" "removeChopShape")
envSetVal("layout" "removeChopShape" 'boolean t)
envSetVal("layout" "removeChopShape" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

removeEmptyNetTerm

```
layout removeEmptyNetTerm boolean { t | nil }
```

Description

Specifies whether the empty net and terminal should be removed with the deleted pin (`t`), or not (`nil`). The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "removeEmptyNetTerm")
envSetVal("layout" "removeEmptyNetTerm" 'boolean t)
envSetVal("layout" "removeEmptyNetTerm" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

removeTerminals

```
layout removeTerminals cyclic { never | always | noInstTermsOnNet }
```

Description

Controls the deletion of the terminal associated with a pin when the pin is deleted.

- `never`: The terminal will not be deleted.
- `always`: The terminal will be deleted if it has no pin figure.
- `noInstTermsOnNet`: The terminal will be deleted if there is no instTerm and pin figure on the terminal.

The default is `noInstTermsOnNet`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "removeTerminals")
envSetVal("layout" "removeTerminals" 'cyclic "never")
envSetVal("layout" "removeTerminals" 'cyclic "always")
envSetVal("layout" "removeTerminals" 'cyclic "noInstTermsOnNet")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

replaceFigs

```
layout replaceFigs boolean { t | nil }
```

Description

Specifies whether selected figures are replaced with a new cell created from the selected figures (`t`), or not (`nil`). This setting is used by `leHiMakeCell` and `leMakeCell`. The default is `t`.

GUI Equivalent

Command	*Edit – Hierarchy – Make Cell*
Field	*Replace selected figures* (Make Cell Form)

Examples

```
envGetVal("layout" "replaceFigs")
envSetVal("layout" "replaceFigs" 'boolean t)
envSetVal("layout" "replaceFigs" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

resetOrientation

```
layout resetOrientation boolean { t | nil }
```

Description

Specifies whether the Move or Copy commands or SKILL code using `leHiMove` or `leHiCopy` resets orientation to 0 prior to the move or copy (`t`), or retains the specified orientation as set by the `orientation` environment variable (`nil`). Terminating the Move or Copy command always resets `orientation` to 0. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "resetOrientation")
envSetVal("layout" "resetOrientation" 'boolean t)
envSetVal("layout" "resetOrientation" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

reshapeType

```
layout reshapeType cyclic { unknown | line | rectangle }
```

Description

Specifies the default shape used by `leHiReShape`. The default is `rectangle`.

GUI Equivalent

Command	*Edit – Advanced – Reshape*
Field	*Mode* (Reshape Form)

Examples

```
envGetVal("layout" "reshapeType")
envSetVal("layout" "reshapeType" 'cyclic "unknown")
envSetVal("layout" "reshapeType" 'cyclic "line")
envSetVal("layout" "reshapeType" 'cyclic "rectangle")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

retainSelectionSet

```
layout retainSelectionSet boolean { t | nil }
```

Description

Specifies that the shapes selection set is retained for the `Move` and `Stretch` commands when the *Repeat Commands* option is selected in the Layout Editor Options form. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "retainSelectionSet")
envSetVal("layout" "retainSelectionSet" 'boolean t)
envSetVal("layout" "retainSelectionSet" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rodAutoName

```
layout rodAutoName string { pin | rectangle | polygon | path }
```

Description

Specifies that rectangles, polygons and/or shape pins be created as ROD objects. These shapes are given either system- or user-assigned names. The default is none.

GUI Equivalent

Command	*Create – Pin, Create – Shape – Rectangle, Create – Shape – Polygon, Create – Shape – Path, Create – Wiring – Wire*
Field	*Create as ROD object* (Create Pin Form, Create Rectangle Form, Create Polygon Form, Create Path Form)

Examples

```
envGetVal("layout" "rodAutoName")
envSetVal("layout" "rodAutoName" 'string "pin")
envSetVal("layout" "rodAutoName" 'string "rectangle")
envSetVal("layout" "rodAutoName" 'string "polygon")
envSetVal("layout" "rodAutoName" 'string "path")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rotateAroundCombinedCenter

```
layout rotateAroundCombinedCenter boolean { t | nil }
```

Description

Enables the multiple selected objects to be rotated around the center of the imaginary bounding box around the selected objects. The default value is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Rotate Around Combined Center* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "rotateAroundCombinedCenter")
envSetVal("layout" "rotateAroundCombinedCenter" 'boolean t)
envSetVal("layout" "rotateAroundCombinedCenter" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rotateInstParam

`layout rotateInstParam string` *`rotationParameter`*

Description

Enables you to specify the name of the parameter to be used to control the rotation of the Pcell. The default value of this environment variable is `rotation`.

GUI Equivalent

Command	*Edit – Properties– Parameter*
Form Field	*Rotation* (Edit Properties Form)

Examples

```
envGetVal("layout" "rotateInstParam")
envSetVal("layout" "rotateInstParam" 'string "rotation")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowAbutFirstUnit

```
layout rowAbutFirstUnit boolean { t | nil }
```

Description

Sets the *Options: Abut The First Row* check box in the *Definition* section of the *Create – P&R Objects – Row* form to abut the first and second rows. The valid values are `t` and `nil`. The default value is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Options: Abut The First Row* (Create Row Form)

Examples

```
envGetVal("layout" "rowAbutFirstUnit")
envSetVal("layout" "rowAbutFirstUnit" 'boolean t)
envSetVal("layout" "rowAbutFirstUnit" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowAbutNumberOfUnits

`layout rowAbutNumberOfUnits int` *`any_integer_greater_than_zero`*

Description

Sets the value of the data entry field for the *Abut* option in the *Definition* section of the *Create – P&R Objects – Row* form to specify how many rows you want to abut. The valid values are any integer greater than zero. The default value is `2`. To specify a value for this data entry field, the *Abut* option must be set to `t`. You can set the *Abut* option with the rowAbutUnit environment variable.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Options: Abut: rows* (Create Row Form)

Examples

```
envGetVal("layout" "rowAbutNumberOfUnits")
envSetVal("layout" "rowAbutNumberOfUnits" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowAbutUnit

```
layout rowAbutUnit boolean { t | nil }
```

Description

Sets the *Options: Abut* check box in the *Definition* section of the *Create – P&R Objects – Row* form to abut a specified number of rows. The valid values are `t` and `nil`. The default value is `nil`. When the Abut option is set to `t`, you can set the number of rows for the data entry field next to the *Abut* option with the rowAbutNumberOfUnits environment variable.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Options: Abut* (Create Row Form)

Examples

```
envGetVal("layout" "rowAbutUnit")
envSetVal("layout" "rowAbutUnit" 'boolean t)
envSetVal("layout" "rowAbutUnit" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowBaseName

`layout rowBaseName string` *`any_string`*

Description

Sets the *Base Name* field in the *Definition* section of the *Create – P&R Objects – Row* form to assign a unique name for each row object you create. The software adds an underscore followed by a number which is greater than or equal to 1. As you create new sites, each site is increment by 1. For example, if your base name is ROW_1, the name assigned to the first site is ROW_1_1, the next is ROW_1_2, the next is ROW_1_3, and so on. If you do not specify a base name, the software assigns the name row_number, which increments by 1 each time you add a site and rows.

The default value is `""`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Base Name* (Create Row Form)

Examples

```
envGetVal("layout" "rowBaseName")
envSetVal("layout" "rowBaseName" 'string "row1")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowCreateMethod

```
layout rowCreateMethod cyclic { "assisted" | "define" | "partition" | "rectilinear"
     }
```

Description

Determines the create method for rows. The assisted mode will not take into account any areas or boundaries. You can create rows that use a rectangular, rectilinear, or a partitioned area to determine the number and placement of the rows

The default value is `assisted`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Create Method* (Create Row Form)

Examples

```
envGetVal("layout" "rowCreateMethod")
envSetVal("layout" "rowCreateMethod" 'cyclic "assisted")
envSetVal("layout" "rowCreateMethod" 'cyclic "define")
envSetVal("layout" "rowCreateMethod" 'cyclic "partition")
envSetVal("layout" "rowCreateMethod" 'cyclic "rectilinear")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowChannelWidth

`layout rowChannelWidth float` *`float_number`*

Description

Sets the *Options: Channel Width* data entry field in the *Definition* section of the *Create – P&R Objects – Row* form to specify the space between the rows you create. If you specify to abut rows (with rowAbutUnit or in the form), *Channel Width* is overridden. The default value is `0.0` in user units.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Options: Channel Width* (Create Row Form)

Examples

```
envGetVal("layout" "rowChannelWidth")
envSetVal("layout" "rowChannelWidth" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowFirstUnitOrient

```
layout rowFirstUnitOrient string { Horizontal | Vertical }
```

Description

Sets the *Row Orientation* radio button in the *Definition* section of the *Create – P&R Objects – Row* form. The valid values are `Horizontal` and `Vertical`. The default value is `Horizontal`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Row Orientation* (Create Row Form)

Examples

```
envGetVal("layout" "rowFirstUnitOrient")
envSetVal("layout" "rowFirstUnitOrient" 'string "Horizontal")
envSetVal("layout" "rowFirstUnitOrient" 'string "Vertical")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowFlipFirstUnit

```
layout rowFlipFirstUnit boolean { t | nil }
```

Description

Sets the *Options: Flip The First Row* check box in the *Definition* section of the *Create – P&R Objects – Row* form to flip the first row in the site. The valid values are `t` and nil. The default value is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Options: Flip The First Row* (Create Row Form)

Examples

```
envGetVal("layout" "rowFlipFirstUnit")
envSetVal("layout" "rowFlipFirstUnit" 'boolean t)
envSetVal("layout" "rowFlipFirstUnit" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowFlipUnit

```
layout rowFlipUnit boolean { t | nil }
```

Description

Sets the *Options: Flip Every Other Row* check box in the *Definition* section of the *Create – P&R Objects – Row* form to flip every other row in the site. The valid values are `t` and `nil`. The default value is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Options: Flip Every Other Row* (Create Row Form)

Examples

```
envGetVal("layout" "rowFlipUnit")
envSetVal("layout" "rowFlipUnit" 'boolean t)
envSetVal("layout" "rowFlipUnit" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowNumberOfRows

`layout rowNumberOfRows int` *`any_integer_greater_than_zero`*

Description

Sets the value of the *Number of Rows* field in the *Create – P&R Objects – Row* form to the number of sites in each row. The default value is `1`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Number of Rows* (Create Row Form)

Examples

```
envGetVal("layout" "rowNumberOfRows")
envSetVal("layout" "rowNumberOfRows" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowPartitionBottomOffset

`layout rowPartitionBottomOffset float` *`any_number_greater_than_zero`*

Description

Sets the offset in which the rows are not allowed to come closer to the bottom of the boundary or cluster edge than the specified float number.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Offset: Bottom* (Create Row Form)

Examples

```
envGetVal("layout" "rowPartitionBottomOffset")
envSetVal("layout" "rowPartitionBottomOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowPartitionLeftOffset

`layout rowPartitionLeftOffset float` *`any_number_greater_than_zero`*

Description

Sets the offset in which the rows are not allowed to come closer to the left side of the boundary or cluster edge than the specified float number.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Offset: Left* (Create Row Form)

Examples

```
envGetVal("layout" "rowPartitionLeftOffset")
envSetVal("layout" "rowPartitionLeftOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowPartitionRightOffset

`layout rowPartitionRightOffset float` *`any_number_greater_than_zero`*

Description

Sets the offset in which the rows are not allowed to come closer to the right side of the boundary or cluster edge than the specified float number.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Offset: Right* (Create Row Form)

Examples

```
envGetVal("layout" "rowPartitionRightOffset")
envSetVal("layout" "rowPartitionRightOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowPartitionTopOffset

`layout rowPartitionTopOffset float` *`any_number_greater_than_zero`*

Description

Sets the offset in which the rows are not allowed to come closer to the top of the boundary or cluster edge than the specified float number.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Offset: Top* (Create Row Form)

Examples

```
envGetVal("layout" "rowPartitionTopOffset")
envSetVal("layout" "rowPartitionTopOffset" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowNumberOfSites

`layout rowNumberOfSites int` *`any_integer_greater_than_zero`*

Description

Sets the value of the *Number of Sites* data entry field in the *Definition* section of the *Create – P&R Objects – Row* form to the number of sites in each row. The number of sites is any integer equal to or greater than zero. The default value is `1`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Number of Sites* (Create Row Form)

Examples

```
envGetVal("layout" "rowNumberOfSites")
envSetVal("layout" "rowNumberOfSites" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowSiteEnable

```
layout rowSiteEnable boolean { t | nil }
```

Description

When `t`, instances will snap to the nearest site within a row. The instance must be either over a legal row or the distance between the instance and the legal row must be within the specified aperture in *Layout Editor Options* form. The default is `nil`. The priority of snapping is as follows:

- snap boundary: if the instance has a snap boundary, the snap boundary will snap to the row site.
- prBoundary: if the instance has a prBoundary but no snap boundary, the prBoundary will snap to the row site.
- cellview: if the instance neither has a snap boundary nor a prBoundary, then the lower left corner of the instance will snap to the row site.

Note: If you are creating or editing an instance and `rowSiteEnable` is set to `t`, `rowSiteEnable` takes precedence over `snapToGrid` for snapping of the instance.

GUI Equivalent

Command	*Options – Editor*
Field	*Instance to Row* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "rowSiteEnable")
envSetVal("layout" "rowSiteEnable" 'boolean t)
envSetVal("layout" "rowSiteEnable" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rowUnitOrient

```
layout rowUnitOrient cyclic { "R0" | "R90" | "R180" | "R270" | "MX" | "MY" | "MXR90"
     | "MYR90" }
```

Description

Sets the *Starting Site Orientation* cyclic field in the *Definition* section of the *Create – P&R Objects – Row* form to set the orientation of the site and the first row created. All subsequent rows will follow the orientation you set in this field. The valid values are `R0`, `R90`, `R180`, `R270`, `MX`, `MY`, `MXR90`, and `MYR90`. The default value is `RO`.

GUI Equivalent

Command	*Create – P&R Objects – Row*
Field	*Starting Site Orientation* (Create Row Form)

Examples

```
envGetVal("layout" "rowUnitOrient")
envSetVal("layout" "rowUnitOrient" 'cyclic "R0")
envSetVal("layout" "rowUnitOrient" 'cyclic "R90")
envSetVal("layout" "rowUnitOrient" 'cyclic "R180")
envSetVal("layout" "rowUnitOrient" 'cyclic "R270")
envSetVal("layout" "rowUnitOrient" 'cyclic "MX")
envSetVal("layout" "rowUnitOrient" 'cyclic "MY")
envSetVal("layout" "rowUnitOrient" 'cyclic "MXR90")
envSetVal("layout" "rowUnitOrient" 'cyclic "MYR90")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerColorPacketName

`graphic rulerColorPacketName string` *`packetName`*

Description

Enables you to specify the name of the packet which has the `Outline` attribute with the color information for the savable ruler. You can add a new packet with the required attributes or modify attributes of a packet in the `display.drf` file or through the *Display Resource Editor*.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "rulerColorPacketName")
envSetVal("graphic" "rulerColorPacketName" 'string "ref_drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerDisplayType

```
layout rulerDisplayType cyclic ( Ruler | Distance | Both )
```

Description

Specifies the display type of the ruler. The default value is `Ruler`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Type* (Create Measurement Form)

Examples

```
envGetVal("layout" "rulerDisplayType")
envSetVal("layout" "rulerDisplayType" 'cyclic "Ruler")
envSetVal("layout" "rulerDisplayType" 'cyclic "Distance")
envSetVal("layout" "rulerDisplayType" 'cyclic "Both")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerEdgeMode

```
layout rulerEdgeMode boolean { t | nil }
```

Description

Enables edge measurement mode for rulers. The default is `nil`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Edge Measurement* (Create Measurement Form)

Examples

```
envGetVal("layout" "rulerEdgeMode")
envSetVal("layout" "rulerEdgeMode" 'boolean t)
envSetVal("layout" "rulerEdgeMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerFontBold

```
layout rulerFontBold boolean { t | nil }
```

Description

Enables you to display the ruler labels in bold face. The default is `nil`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Display Options: Bold* (Create Measurement Form)

Examples

```
envGetVal("layout" "rulerFontBold")
envSetVal("layout" "rulerFontBold" 'boolean t)
envSetVal("layout" "rulerFontBold" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerFontSize

```
layout rulerFontSize cyclic { "Small" | "Medium" | "Large" }
graphic rulerFontSize int integer_number
```

Description

Specifies the font size for the ruler labels.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Display Options: Font Size* (Create Measurement Form)

Examples

```
envGetVal("layout" "rulerFontSize")
envSetVal("layout" "rulerFontSize" 'cyclic "Small")
envSetVal("layout" "rulerFontSize" 'cyclic "Medium")
envSetVal("layout" "rulerFontSize" 'cyclic "Large")
envGetVal("graphic" "rulerFontSize")
envSetVal("graphic" "rulerFontSize" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerLabelBackgroundColor

```
layout rulerLabelBackgroundColor cyclic { "nil" | "black" | "gray" | "pink" | "red"
     | "green" | "forest" | "blue" | "navy" | "chocolate" }
```

```
graphic rulerLabelBackgroundColor string { "nil" | "black" | "gray" | "pink" | "red"
     | "green" | "forest" | "blue" | "navy" | "chocolate" }
```

Description

Specifies the background color of the ruler labels.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Display Options: Background Color* (Create Measurement Form)

Examples

```
envGetVal("layout" "rulerLabelBackgroundColor")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "nil")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "black")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "gray")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "pink")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "red")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "green")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "forest")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "blue")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "navy")
envSetVal("layout" "rulerLabelBackgroundColor" 'cyclic "chocolate")
envGetVal("graphic" "rulerLabelBackgroundColor")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "nil")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "black")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "gray")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "pink")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "red")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "green")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "forest")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "blue")
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "navy")
```


© 2020 All Rights Reserved.

```
envSetVal("graphic" "rulerLabelBackgroundColor" 'string "chocolate")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

rulerSegmentModeType

```
layout rulerSegmentModeType cyclic ( Single | Multiple | Auto )
```

Description

Specifies the type of ruler created by the measurement. You can set the environment variable to one of the following values:

- `Single`: Creates a single-segment ruler. This is the measurement between points on the canvas.
- `Multiple`: Creates a multiple-segment ruler. In this mode, you can view the measurement between consecutive points that you specify.
- `Auto`: Creates a multiple-segment ruler. In this mode, points are automatically calculated. There are two ways in which the ruler is created in the auto mode:
  - An auto ruler is created between two specified points on the canvas.
  - You press the `Shift` key in the auto mode. In this case, an orthogonal ruler is created with all edges visible on the canvas in vertical or horizontal direction. In this case, the direction of the ruler is based on the edge closest to the pointer.

The default value is `Single`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Segment Mode* (Create Measurement Form)

Examples

```
envGetVal("layout" "rulerSegmentModeType")
envSetVal("layout" "rulerSegmentModeType" 'cyclic "Single")
envSetVal("layout" "rulerSegmentModeType" 'cyclic "Multiple")
envSetVal("layout" "rulerSegmentModeType" 'cyclic "Auto")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

saveACopyLibName

`layout saveACopyLibName string` *`any_library_name`*

Description

Specifies the default library name where the design is saved while saving a copy. The default is none.

GUI Equivalent

Command	*File – Save a Copy*
Field	*Library* cyclic field (Save a Copy Form)

Examples

```
envGetVal("layout" "saveACopyLibName")
envSetVal("layout" "saveACopyLibName" 'string "MyLib")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

saveRulers

```
layout saveRulers boolean { t | nil }
```

Description

Saves the rulers in a cellview when you save the cellview The default is `t`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Savable Measurement* (Create Measurement Form)

Examples

```
envGetVal("layout" "saveRulers")
envSetVal("layout" "saveRulers" 'boolean t)
envSetVal("layout" "saveRulers" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

schematicDefaultApp

`graphic schematicDefaultApp string` *`registered_application`*

Description

Automates the launching of a default application for cellviews of type `schematic`. This environment variable setting is ignored when descending into or editing-in-place in the same view type.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "schematicDefaultApp")
envSetVal("graphic" "schematicDefaultApp" 'string "VSE L")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

schematicDefaultTier

```
graphic schematicDefaultTier string { L | XL }
```

Description

Sets the *Choose Schematic/Symbol Level* field in the *File – Set Default Application* form to a default application to launch for cellviews of type `schematic` or `symbol`. The default is `""`.

GUI Equivalent

Command	In a schematic or symbol cellview, *File – Set Default Application*
Field	*Choose Schematic/Symbol Level (Set Default Schematic Application Form)*

Examples

```
envGetVal("graphic" "schematicDefaultTier")
envSetVal("graphic" "schematicDefaultTier" 'string "L")
envSetVal("graphic" "schematicDefaultTier" 'string "XL")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

schematicSymbolDefaultApp

`graphic schematicSymbolDefaultApp string` *`registered_application`*

Description

Automates the launching of a default application for cellviews of type `schematicSymbol`. This environment variable setting is ignored when descending into or editing-in-place in the same view type.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "schematicSymbolDefaultApp")
envSetVal("graphic" "schematicSymbolDefaultApp" 'string "VSE L")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

scrollPercent

`graphic scrollPercent int` *`integer_value`*

`layout scrollPercent int` *`integer_value`*

Description

Enables you to specify the scrolling percent per window. The default value is `25`.

GUI Equivalent

Command	*Options – Display*
Field	*Scroll Percent* (Display Options Form)

Examples

```
envGetVal("graphic" "scrollPercent")
envSetVal("graphic" "scrollPercent" 'int 30)
envGetVal("layout" "scrollPercent")
envSetVal("layout" "scrollPercent" 'int 30)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

segSnapMode

```
layout segSnapMode string { orthogonal | diagonal | anyAngle | horizontal | vertical
     }
```

```
graphic segSnapMode string { orthogonal | diagonal | anyAngle | horizontal |
     vertical }
```

Description

Determines how *Edit* command entry points snap to the grid. The layout default is `orthogonal`, the graphic default is `anyAngle`.

Note: `anyAngle` is not a valid value for the *Create Wire* (XL, GXL) command.

GUI Equivalent

Command	*Options – Display*
Field	*Snap Modes*: *Edit* (Display Options Form)

Examples

```
envGetVal("layout" "segSnapMode")
envSetVal("layout" "segSnapMode" 'cyclic "orthogonal")
envSetVal("layout" "segSnapMode" 'cyclic "diagonal")
envSetVal("layout" "segSnapMode" 'cyclic "anyAngle")
envSetVal("layout" "segSnapMode" 'cyclic "horizontal")
envSetVal("layout" "segSnapMode" 'cyclic "vertical")
envGetVal("graphic" "segSnapMode")
envSetVal("graphic" "segSnapMode" 'cyclic "orthogonal")
envSetVal("graphic" "segSnapMode" 'cyclic "diagonal")
envSetVal("graphic" "segSnapMode" 'cyclic "anyAngle")
envSetVal("graphic" "segSnapMode" 'cyclic "horizontal")
envSetVal("graphic" "segSnapMode" 'cyclic "vertical")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectCreatedObj

```
layout selectCreatedObj boolean { t | nil }
```

Description

Determines whether or not an object remains selected after its creation. If set to `t`, all objects you create are selected after creation. Any object selected before starting the create command also remain selected. The default is `nil`.

GUI Equivalent

Command	*Options – Editor*
Field	*Select Created Object* (Layout Editor Options Form)

Examples

```
envGetVal("graphic" "selectCreatedObj")
envSetVal("graphic" "selectCreatedObj" 'boolean t)
envSetVal("graphic" "selectCreatedObj" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectHilightPacket

`graphic selectHilightPacket string` *`packet_name`*

Description

Specifies the packet name to be used to display the selection highlight.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "selectHilightPacket")
envSetVal("graphic" "selectHilightPacket" 'string "myPacket")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectionAperture

`layout selectionAperture int` *`any_positive_integer`*

`graphic selectionAperture int` *`any_positive_integer`*

Description

Creates a selection aperture box around the pointer with the specified number of pixels. A setting of 5 will create a square of 6 by 6 pixels, which includes the center pixel. When a point is clicked the pointer will select the closest object contained within the aperture area. The next click will select the second closest object contained within the aperture area, and so on. Only one object is selected with each click. When no more objects are contained within the aperture area that have not already been selected, the pointer will cycle back to the first object selected. and each subsequent click will repeat the selection order. The default is `5`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "selectionAperture")
envSetVal("layout" "selectionAperture" 'int 10)
envGetVal("graphic" "selectionAperture")
envSetVal("graphic" "selectionAperture" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectMarkersAtLast

```
graphic selectMarkersAtLast boolean { t | nil }
```

Description

Selects markers at last, after the shapes even if the the marker edge is closer than the shape edge or markers and shapes overlap. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "selectionAperture")
envSetVal("layout" "selectionAperture" 'int 10)
envGetVal("graphic" "selectionAperture")
envSetVal("graphic" "selectionAperture" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectObjectLargerThanWindow

```
graphic selectObjectLargerThanWindow boolean { t | nil }
```

Description

Controls whether or not you can select an object when its size is greater than the size of the design canvas in which it is displayed. The default is `t`.

GUI Equivalent

Command	*Options – Selection*
Field	*Select objects larger than the window* (Selection Options Form)

Examples

```
envGetVal("graphic" "selectObjectLargerThanWindow")
envSetVal("graphic" "selectObjectLargerThanWindow" 'boolean t)
envSetVal("graphic" "selectObjectLargerThanWindow" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectObjectInEditCvFirst

```
graphic selectObjectInEditCvFirst boolean { t | nil }
```

Description

Controls whether you can select objects belonging to the cellview being edited in preference over the other objects, which might be on a lower layers. The default is `nil`.

GUI Equivalent

Command	*Options – Selection*
Field	*Select objects of edit cellview first* (Selection Options Form)

Examples

```
envGetVal("graphic" "selectObjectInEditCvFirst")
envSetVal("graphic" "selectObjectInEditCvFirst" 'boolean t)
envSetVal("graphic" "selectObjectInEditCvFirst" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectOnlyInstWithSelectableShapes

```
layout selectOnlyInstWithSelectableShapes boolean { t | nil }
```

Description

Determines whether instances are selectable. If `t` an instance is selected only if *Instances* is selected in the *Objects* assistant and the instance or instances contain shapes that are on selectable layers. If `nil` an instance or instances are selected when the *Instances* option is on, even if none of the shapes within them are on selectable layers. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "selectOnlyInstWithSelectableShapes")
envSetVal("layout" "selectOnlyInstWithSelectableShapes" 'boolean t)
envSetVal("layout" "selectOnlyInstWithSelectableShapes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectProtectedItems

```
graphic selectProtectedItems boolean { t | nil }
```

Description

Determines whether protected items should be selected or not. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "selectProtectedItems")
envSetVal("graphic" "selectProtectedItems" 'boolean t)
envSetVal("graphic" "selectProtectedItems" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectUnderCursorMenuAlphaSorting

```
graphic selectUnderCursorMenuAlphaSorting boolean { t | nil }
```

Description

Specifies the objects listed in the *(De)Select Under Cursor* context menu appear in alphabetical order. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "selectUnderCursorMenuAlphaSorting")
envSetVal("graphic" "selectUnderCursorMenuAlphaSorting" 'boolean t)
envSetVal("graphic" "selectUnderCursorMenuAlphaSorting" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

selectUnderCursorMenuLimit

`graphic selectUnderCursorMenuLimit int` *`integer_value`*

Description

Controls the appearance of the *(De)Select Under Cursor* context menu. The default is `20`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "selectUnderCursorMenuLimit")
envSetVal("graphic" "selectUnderCursorMenuLimit" 'int 25)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

setDxDyToZero

```
layout setDxDyToZero boolean { t | nil }
```

Description

Enables you to set the *dX* and *dY* fields to `0` on the status toolbar. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "setDxDyToZero")
envSetVal("layout" "setDxDyToZero" 'boolean t)
envSetVal("layout" "setDxDyToZero" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

setupConstraintGroup

`layout setupConstraintGroup string` *`any_valid_constraint_group`*

Description

Specifies the default set of legal interconnect layers which Virtuoso XL (`virtuosoDefaultExtractorSetup`) use for connectivity extraction. A custom constraint group can have user defined names (`myRoutingLayers`). A constraint group must minimally contain valid interconnect layers, and may optionally include a list of valid vias.

GUI Equivalent

None

Examples

```
envGetVal("layout" "setupConstraintGroup")
envSetVal("layout" "setupConstraintGroup" 'string "x y z")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

showWhatsNew

```
layout showWhatsNew string { yes | no }
```

Description

Determines whether the layout editor What's New window is displayed when a cellview window is opened (`yes`) or not (`no`). The default is `yes`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "showWhatsNew")
envSetVal("layout" "showWhatsNew" 'string "yes")
envSetVal("layout" "showWhatsNew" 'string "no")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotEdgeToSlotOffset

`layout slotEdgeToSlotOffset float` *`float_number`*

Description

Specifies the distance between the slot edge and the shape edge. The default value is `1.0`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path – Slot Options*
Field	*Spacing: Edge* (Create Slot Form, Create Slot Form, Create Rectangle Form)
	Slot Offset: Edge to Slot (Create Path Form)

Examples

```
envGetVal("layout" "slotEdgeToSlotOffset")
envSetVal("layout" "slotEdgeToSlotOffset" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotEnable

```
layout slotEnable boolean { t | nil }
```

```
graphic slotEnable boolean { t | nil }
```

Description

Overlaps the shape to slot with another shape that is displayed like a hole in the shape. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Slot Shapes (*Display Options Form*)*

Examples

```
envGetVal("layout" "slotEnable")
envSetVal("layout" "slotEnable" 'boolean t)
envSetVal("layout" "slotEnable" 'boolean nil)
envGetVal("graphic" "slotEnable")
envSetVal("graphic" "slotEnable" 'boolean t)
envSetVal("graphic" "slotEnable" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotEnableRect

```
layout slotEnableRect boolean { t | nil }
```

Description

Specifies whether the rectangle is to be created with slots. The default is `nil`.

GUI Equivalent

Command	*Create – Shape – Rectangle*
Field	*Slotting: Enable* (Create Rectangle Form)

Examples

```
envGetVal("layout" "slotEnableRect")
envSetVal("layout" "slotEnableRect" 'boolean t)
envSetVal("layout" "slotEnableRect" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotEnablePath

```
layout slotEnablePath boolean { t | nil }
```

Description

Specifies whether the path is to be created with slots. The default is `nil`.

GUI Equivalent

Command	*Create – Shape – Path – Path Options*
Field	*Enable Metal Slotting* (Create Path Form)

Examples

```
envGetVal("layout" "slotEnablePath")
envSetVal("layout" "slotEnablePath" 'boolean t)
envSetVal("layout" "slotEnablePath" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotInterSlotSpacing

`layout slotInterSlotSpacing float` *`float_number`*

Description

Specifies the distance between the slots in a row. The default value is `1.0`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path – Slot Options*
Field	*Spacing: Column* (Create Slot Form, Edit Slot Form, Create Rectangle Form)
	Slot Spacing: Between Slots (Create Path Form)

Examples

```
envGetVal("layout" "slotInterSlotSpacing")
envSetVal("layout" "slotInterSlotSpacing" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotLength

`layout slotLength float` *`float_number`*

Description

Specifies the length of a slot. The default value is `1.0`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path – Slot Options*
Field	*Dimensions: Length* (Create Slot Form, Edit Slot Form, Create Rectangle Form) *Slot Dimensions: Length* (Create Path Form)

Examples

```
envGetVal("layout" "slotLength")
envSetVal("layout" "slotLength" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotRowSpacing

`layout slotRowSpacing float` *`float_number`*

Description

Specifies the distance between two slot rows. The default value is `1.0`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path – Slot Options*
Field	*Spacing: Row* (Create Slot Form, Edit Slot Form, Create Rectangle Form)
	Slot Spacing: Between Slot Rows (Create Path Form)

Examples

```
envGetVal("layout" "slotRowSpacing")
envSetVal("layout" "slotRowSpacing" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotShapeMode

```
layout slotShapeMode cyclic { "Preserve" | "Polygon" | "ROD" }
```

Description

Specifies the type of the resultant shape with slots. The default value is `Preserve`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*
Field	*Shape Type* (Create Slot Form, Edit Slot Form)

Examples

```
envGetVal("layout" "slotShapeMode")
envSetVal("layout" "slotShapeMode" 'cyclic "Preserve")
envSetVal("layout" "slotShapeMode" 'cyclic "Polygon")
envSetVal("layout" "slotShapeMode" 'cyclic "ROD")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotSpecifyArea

```
layout slotSpecifyArea boolean ( t | nil )
```

Description

Selects the *Width/Area* option. The default value is `nil`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path – Slot Options*
Field	*Specify width, area* (Create Slot Form, Edit Slot Form, Create Rectangle Form)
	Slot Dimensions: Specify: Width/Area (Create Path Form)

Examples

```
envGetVal("layout" "slotSpecifyArea")
envSetVal("layout" "slotSpecifyArea" 'boolean t)
envSetVal("layout" "slotSpecifyArea" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotStaggerOffset

`layout slotStaggerOffset float` *`float_number`*

Description

Specifies the spacing between the edges of slots in alternate rows. The default value is `0.0`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Path – Slot Options*
Field	*Stagger* (Create Rectangle Form, Edit Slot Form)
	Slot Offset: Stagger (Create Path Form)

Examples

```
envGetVal("layout" "slotStaggerOffset")
envSetVal("layout" "slotStaggerOffset" 'float 1.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotWarnWhenMoreThanRows

`layout slotWarnWhenMoreThanRows int` *`integer_number`*

Description

Displays a warning message in the CIW if you try to create a rectangle or a path with more slot rows than specified by this environment variable. You can create a rectangle or a path with more slot rows than the number specified, but the slots do not appear in the drag set while creating the rectangle or path if the slot rows exceed the specified number.

The default value of this environment variable is 1000.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path*
Field	NA

Examples

```
envGetVal("layout" "slotWarnWhenMoreThanRows")
envSetVal("layout" "slotWarnWhenMoreThanRows" 'int 1500)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

slotWidth

`layout slotWidth float` *`float_number`*

Description

Specifies the width of a slot. The default value is `1.0`.

GUI Equivalent

Command	*Create – Slot*, *Edit – Advanced – Slot*, *Create – Shape – Rectangle*, *Create – Shape – Path – Slot Options*
Field	*Dimensions: Width* (Create Slot Form, Edit Slot Form)
	Slot Dimensions: Width (Create Path Form)

Examples

```
envGetVal("layout" "slotWidth")
envSetVal("layout" "slotWidth" 'float 2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapDeviceInDetailMode

```
layout snapDeviceInDetailMode boolean { t | nil }
```

Description

Snaps devices that do not have any property override, in detail mode. Also, detail mode snapping in horizontal direction is performed using shapes on `poly` layer only. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapDeviceInDetailMode")
envSetVal("layout" "snapDeviceInDetailMode" 'boolean t)
envSetVal("layout" "snapDeviceInDetailMode" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapMode

```
layout snapMode string { orthogonal | diagonal | anyAngle | L90XFirst | L90YFirst }
graphic snapMode string { orthogonal | diagonal | anyAngle | L90XFirst | L90YFirst }
```

Description

Determines how *Create* command entry points snap to the grid. The layout default is `orthogonal`, the graphic default is `anyAngle`.

Note: `anyAngle` is not a valid value for the *Create Wire* (XL, GXL) command.

GUI Equivalent

Command	*Options – Display*
Field	*Snap Modes: Create* (Display Options Form)

Examples

```
envGetVal("layout" "snapMode")
envSetVal("layout" "snapMode" 'string "orthogonal")
envSetVal("layout" "snapMode" 'string "diagonal")
envSetVal("layout" "snapMode" 'string "anyAngle")
envSetVal("layout" "snapMode" 'string "L90XFirst")
envSetVal("layout" "snapMode" 'string "L90YFirst")
envGetVal("graphic" "snapMode")
envSetVal("graphic" "snapMode" 'string "orthogonal")
envSetVal("graphic" "snapMode" 'string "diagonal")
envSetVal("graphic" "snapMode" 'string "anyAngle")
envSetVal("graphic" "snapMode" 'string "L90XFirst")
envSetVal("graphic" "snapMode" 'string "L90YFirst")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternDrawingStyle

```
layout snapPatternDrawingStyle cyclic { full | tracks | boundary | periods |
     periodsAndTracks }
```

Description

Determines whether you see the boundary, the tracks, the fins, and periods (`full)` or the boundary and the tracks (`tracks`), or just the boundary (`boundary`) for snap patterns, or periods (`periods`), or periods and tracks (`periodsandTracks`) for width spacing patterns. The default is `full`.

GUI Equivalent

Command	*Options – Display*
Field	*Snap Pattern Display* (Display Options Form)

Examples

```
envGetVal("layout" "snapPatternDrawingStyle")
envSetVal("layout" "snapPatternDrawingStyle" 'cyclic "full")
envSetVal("layout" "snapPatternDrawingStyle" 'cyclic "tracks")
envSetVal("layout" "snapPatternDrawingStyle" 'cyclic "boundary")
envSetVal("layout" "snapPatternDrawingStyle" 'cyclic "periods")
envSetVal("layout" "snapPatternDrawingStyle" 'cyclic "periodsAndTracks")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternFastDetailModeSnapping

```
layout snapPatternFastDetailModeSnapping boolean { t | nil }
```

Description

Prioritizes the snapping of grids at the higher level of hierarchy and snapping is done level by level by level. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapPatternFastDetailModeSnapping")
envSetVal("layout" "snapPatternFastDetailModeSnapping" 'boolean t)
envSetVal("layout" "snapPatternFastDetailModeSnapping" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternGlobalGridAnchoring

```
graphic snapPatternGlobalGridAnchoring boolean { t | nil }
```

Description

Determines whether the snap pattern global grid anchoring is enabled (`t`) or not (`nil`).

Note: This environment variable applies only to newly created cellviews. Once a cellview is created, the setting of this environment variable is retained. To view or edit the setting after the cellview is created, you can use the functions, `dbSetGlobalGridOffsetReferenceType` and `dbGetGlobalGridOffsetReferenceType`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "snapPatternGlobalGridAnchoring")
envSetVal("graphic" "snapPatternGlobalGridAnchoring" 'boolean t)
envSetVal("graphic" "snapPatternGlobalGridAnchoring" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternSnapping

```
layout snapPatternSnapping boolean { t | nil }
```

Description

Determines whether the snap pattern snapping is enabled (`t`) or not (`nil`). The default value is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Snap Pattern Snapping* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "snapPatternSnapping")
envSetVal("layout" "snapPatternSnapping" 'boolean t)
envSetVal("layout" "snapPatternSnapping" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternSnappingDepth

`layout snapPatternSnappingDepth int` *`integer_number`*

Description

Lets you specify the snap depth for snap pattern snapping. The default value is `32`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapPatternSnappingDepth")
envSetVal("layout" "snapPatternSnappingDepth" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternSnappingDepthMethod

```
layout snapPatternSnappingDepthMethod cyclic { DisplayPlusOne | UserDefined }
```

Description

Determines the method to be used to define the snap depth for snap pattern snapping. If the `snapPatternSnappingDepthMethod` is set to `DisplayPlusOne`, the value display stop level + 1 is used as the snap depth. If the `snapPatternSnappingDepthMethod` is set to `UserDefined`, the snap depth specified by the environment variable snapPatternSnappingDepth is used.The default value is `DisplayPlusOne`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapPatternSnappingDepthMethod")
envSetVal("layout" "snapPatternSnappingDepthMethod" 'cyclic "DisplayPlusOne")
envSetVal("layout" "snapPatternSnappingDepthMethod" 'cyclic "UserDefined")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternSnappingMode

```
layout snapPatternSnappingMode cyclic { detail | abstract }
```

Description

Determines whether the instance placement mode is `detail` or `abstract`. The default value is `detail`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapPatternSnappingMode")
envSetVal("layout" "snapPatternSnappingMode" 'cyclic "detail")
envSetVal("layout" "snapPatternSnappingMode" 'cyclic "abstract")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapPatternSnapToXYGridFirst

```
layout snapPatternSnapToXYGridFirst boolean { t | nil }
```

Description

Enables snapping to the snap pattern grids closest to the X and Y snap spacing values. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapPatternSnapToXYGridFirst")
envSetVal("layout" "snapPatternSnapToXYGridFirst" 'boolean t)
envSetVal("layout" "snapPatternSnapToXYGridFirst" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapRectToWSPGrid

```
layout snapRectToWSPGrid boolean { t | nil }
```

Description

Snaps rectangles to the width spacing pattern grid. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapRectToWSPGrid")
envSetVal("layout" "snapRectToWSPGrid" 'boolean t)
envSetVal("layout" "snapRectToWSPGrid" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapResizeWireObjects

```
layout snapResizeWireObjects boolean { t | nil }
```

Description

Snaps, resizes, and colors all pathsegs and vias in the selection set during the Move and Copy commands.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapResizeWireObjects")
envSetVal("layout" "snapResizeWireObjects" 'boolean t)
envSetVal("layout" "snapResizeWireObjects" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

snapToMfgGrid

```
layout snapToMfgGrid boolean { t | nil }
```

Description

When `snapToMfgGrid` is `t`, the converted polygon vertices are snapped to mfgGrid. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "snapToMfgGrid")
envSetVal("layout" "snapToMfgGrid" 'boolean t)
envSetVal("layout" "snapToMfgGrid" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

softFence

```
layout softFence boolean { t | nil }
```

Description

Determines whether or not to create the area boundary as a soft fence. The default is `nil`.

GUI Equivalent

Command	*Create – P&R Objects – Area Boundary*
Field	*Soft Fence* (Create Area Boundary Form)

Examples

```
envGetVal("layout" "softFence")
envSetVal("layout" "softFence" 'boolean t)
envSetVal("layout" "softFence" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

spineSelection

```
graphic spineSelection boolean { t | nil }
```

Description

When a pathseg is selected, this variable determines whether any additional contiguous pathsegs are selected (`t`) or not (`nil`). The default value is `t`.

GUI Equivalent

Command	*Options – Selection*
Field	*Selection Controls: Spine* (Selection Options Form)

Examples

```
envGetVal("graphic" "spineSelection")
envSetVal("graphic" "spineSelection" 'boolean t)
envSetVal("graphic" "spineSelection" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

startLevel

`layout startLevel int` *`any_integer_from_0_to_32`*

`graphic startLevel int` *`any_integer_from_0_to_32`*

Description

Number of the highest level in the hierarchy visible. The default is `0`.

GUI Equivalent

Command	*Options – Display*
Field	*Display Levels: Start* (Display Options Form)

Examples

```
envGetVal("layout" "startLevel")
envSetVal("layout" "startLevel" 'int 10)
envGetVal("graphic" "startLevel")
envSetVal("graphic" "startLevel" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

statusBarXYdXdYLength

```
layout statusBarXYdXdYLength int integer
```

Description

Controls the width of the *X*, *Y*, *dX*, and *dY* fields on the status toolbar. The default is `10`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "statusBarXYdXdYLength")
envSetVal("layout" "statusBarXYdXdYLength" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

statusToolbarFields

```
layout statusToolbarFields string { information and style }
```

Description

Specifies the information that you want to display on the status toolbar. You can display fields, such as cellsize, number of selected objects, environment variable values, shell environment variable values, window variable values. It also specifies the display format of the information displayed on the status toolbar. The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "statusToolbarFields")
```

```
envSetVal("layout" "statusToolbarFields" 'string "(Select t \"<span
style=\\\"color:red;font-weight:bold\\\">Select: %s</span>\")")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

stopLevel

`layout stopLevel int` *`any_integer_from_0_to_32`*

`graphic stopLevel int` *`any_integer_from_0_to_32`*

Description

Number of the lowest level in the hierarchy visible. The default is `0`.

GUI Equivalent

Command	*Options – Display*
Field	*Display Levels: Stop* (Display Options Form)

Examples

```
envGetVal("layout" "stopLevel")
envSetVal("layout" "stopLevel" 'int 10)
envGetVal("graphic" "stopLevel")
envSetVal("graphic" "stopLevel" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

stretchPcellApplyToName

```
layout stretchPcellApplyToName boolean { t | nil }
```

Description

For Pcells, in the Stretch in X and Stretch in Y forms, determines whether the *Reference Dimension* (default) and *Minimum and Maximum* fields apply to Name only (t) or to the whole expression (nil). The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "stretchPcellApplyToName")
envSetVal("layout" "stretchPcellApplyToName" 'boolean t)
envSetVal("layout" "stretchPcellApplyToName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

stretchHandlesLayer

`graphic stretchHandlesLayer string` *`any_valid_layer-pupose_pair`*

Description

Specifies the layer that stretch handles are displayed on for stretchable Pcells. The default is `y0 drawing`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "stretchHandlesLayer")
envSetVal("graphic" "stretchHandlesLayer" 'string "metal1 drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

stretchViaMode

```
layout stretchViaMode cyclic { Stretch Row/Col | Stretch Metal }
```

Description

Specifies the mode of stretching of the via. The default value is `Stretch Metal`.

GUI Equivalent

Command	*Edit – Stretch*
Field	*Special Objects: Via Mode* (Stretch Form)

Examples

```
envGetVal("layout" "stretchViaMode")
envSetVal("layout" "stretchViaMode" 'cyclic "Stretch Row/Col")
envSetVal("layout" "stretchViaMode" 'cyclic "Stretch Metal")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

supressFreezePcellUI

```
layout supressFreezePcellUI boolean { t | nil }
```

Description

When this variable is set to `t`, it freezes the Pcell instance without displaying the (Freeze Pcell Instance Form). The default values for library, cell, and view are used for the freeze operation. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "supressFreezePcellUI")
envSetVal("layout" "supressFreezePcellUI" 'boolean t)
envSetVal("layout" "supressFreezePcellUI" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

tapConnectedFigsOnly

```
layout tapConnectedFigsOnly boolean { t | nil }
```

Description

When set to `t`, the tapping of objects ignores figures that are not associated with a net. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "tapConnectedFigsOnly")
envSetVal("layout" "tapConnectedFigsOnly" 'boolean t)
envSetVal("layout" "tapConnectedFigsOnly" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

taperStyle

```
layout taperStyle cyclic { unknown | linear | exponential }
```

Description

Specifies the default type of taper. The default is `linear`.

GUI Equivalent

Command	*Create – Microwave– Taper*
Field	*Taper Style* (Transmission Line Taper Form)

Examples

```
envGetVal("layout" "taperStyle")
envSetVal("layout" "taperStyle" 'cyclic "unknown")
envSetVal("layout" "taperStyle" 'cyclic "linear")
envSetVal("layout" "taperStyle" 'cyclic "exponential")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

tapLayerFilter

```
layout tapLayerFilter string { list_of_any_valid_layers }
```

Description

Filters layers to be tapped. The layers names must be separated by a space.and enclosed in quotation marks; for example, `"metal1 metal2"`

The default is `""`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "tapLayerFilter")
envSetVal("layout" "tapLayerFilter" 'string "metal1 metal2")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

tapPurposeList

`layout tapPurposeList string { list_of_any_valid_layer_purposes }`

Description

Prioritizes the order in which tap cycles through layers. When you set Wire Auto Tap or Shape Auto Tap to on, and *Select from Overlaps* to off, the tap purpose list is used to prioritize the order in which layers are cycled through during tap. When Wire Auto Tap and Shape Auto Tap are set to off and *Select from Overlaps* is on, the *Choose object to tap from* form appears, so the Tap Purpose List is not considered. The default is `drawing`.

GUI Equivalent

Command	*Options – Editor*
Field	*Tap Purpose List* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "tapPurposeList")
envSetVal("layout" "tapPurposeList" 'string "metal1 drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

tapType

```
layout tapType string { layer | net | attribute }
```

Description

Sets what information is obtained from tapping. When set to `layer`, the tap command sets the entry layer to the layer of the object on which you have tapped. When set to `net`, the tap command captures the net names of rectangles, polygons and paths. When set to `attribute`, the tap command captures the path net name, beginning and ending extension in user units, the path end type, and the path width, for trls, bends and tapers the tap command captures the bend factor (maximum width before they are chamfered), taper style (linear or exponential), number of points to taper in user units, the bend style (bend chamfer or radial), the width of a chamfered bend in a transmission line as a multiple of its standard width in user units, the number of segments in a radial bend in a transmission line as a function of the number of segments in a 180-degree radial bend, the radius of a radial bend in a transmission line as a multiple of the transmission-line width in user units, the width of the first end of a transmission line bend in user units, the width of the second end of a transmission line bend, and the width of the second end of a transmission line bend in user units.

Tapping of nets is only supported in the layout editor XL and GXL applications.

The default is `layer`.

GUI Equivalent

Command	*Tools – Tap*
Field	*Layer, Net, Attributes* (Tap Form)

Examples

```
envGetVal("layout" "tapType")
envSetVal("layout" "tapType" 'string "layer")
envSetVal("layout" "tapType" 'string "net")
envSetVal("layout" "tapType" 'string "attribute")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

targetColor

`layout targetColor string color`

Description

Controls the color of the snap target highlight while creating a ruler. The default value is `yellow`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Snap Target Color* (Create Measurement Form)

Examples

```
envGetVal("layout" "targetColor")
envSetVal("layout" "targetColor" 'string "white pink red green")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

targetLineStyle

`layout targetLineStyle string "line_style"`

Description

Controls the style of the snap target highlight while creating a ruler. The default value is `thickLine`. Line style options include `dashed`, `dots`, `dashDot`, `shortDash`, `doubleDash`, `thickLine`, and `thickDash`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Snap Target Line Style* (Create Measurement Form)

Examples

```
envGetVal("layout" "targetLineStyle")
envSetVal("layout" "targetLineStyle" 'string "solid dashed dots thickLine")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

targetWidth

`layout targetWidth float` *`float_number`*

Description

Controls the width of the snap target highlight while creating a ruler. The value must be a integer point number. The default value is `3.0`.

GUI Equivalent

Command	*Tools – Create Measurement*
Field	*Snap Target Width* (Create Measurement Form)

Examples

```
envGetVal("layout" "targetWidth")
envSetVal("layout" "targetWidth" 'float 5.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

termDirection

```
layout termDirection cyclic { "input" | "output" | "inputOutput" | "switch" |
     "jumper" | "unused" | "tristate" }
```

Description

Specifies the default I/O type of a pin. The default is `inputOutput`.

GUI Equivalent

Command	*Create – Pin*
Field	*I/O Type* (Create Pin Form)

Examples

```
envGetVal("layout" "termDirection")
envSetVal("layout" "termDirection" 'cyclic "input")
envSetVal("layout" "termDirection" 'cyclic "output")
envSetVal("layout" "termDirection" 'cyclic "inputOutput")
envSetVal("layout" "termDirection" 'cyclic "switch")
envSetVal("layout" "termDirection" 'cyclic "jumper")
envSetVal("layout" "termDirection" 'cyclic "unused")
envSetVal("layout" "termDirection" 'cyclic "tristate")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

termName

`layout termName string` *`any_terminal_name`*

Description

Specifies one or more names for terminals on which pins are created. You can specify as many names as you want (separated by spaces). As each name is used, it is removed from the list. If all names are used, the field in the Pin form becomes empty, and you cannot create any more pins until you specify a terminal name. The default is `none`.

GUI Equivalent

Command	*Create – Pin*
Field	*Terminal Names* (Create Pin Form)

Examples

```
envGetVal("layout" "termName")
envSetVal("layout" "termName" 'string "term1")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

textJustificationOn

```
layout textJustificationOn boolean { t | nil }
graphic textJustificationOn boolean { t | nil }
```

Description

Determines whether the origins of labels are displayed. The origins appear as plus signs. The default is `t`.

GUI Equivalent

Command	*Options – Display*
Field	*Label Origins* (Display Options Form)

Examples

```
envGetVal("layout" "textJustificationOn")
envSetVal("layout" "textJustificationOn" 'boolean t)
envSetVal("layout" "textJustificationOn" 'boolean nil)
envGetVal("graphic" "textJustificationOn")
envSetVal("graphic" "textJustificationOn" 'boolean t)
envSetVal("graphic" "textJustificationOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

tinyInstDetail

```
layout tinyInstDetail boolean { t | nil }
```

```
graphic tinyInstDetail boolean { t | nil }
```

Description

Determines whether to display all the details of an instance, even those smaller than the size set in the variable, `filterSize`. When `nil`, the cellview redraws faster. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "tinyInstDetail")
envSetVal("layout" "tinyInstDetail" 'boolean t)
envSetVal("layout" "tinyInstDetail" 'boolean nil)
envGetVal("graphic" "tinyInstDetail")
envSetVal("graphic" "tinyInstDetail" 'boolean t)
envSetVal("graphic" "tinyInstDetail" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

transientRulerColorPacketName

`graphic transientRulerColorPacketName string` *`packetName`*

Description

Enables you to specify the name of the packet which has the `Outline` attribute with the color information for the non-savable ruler. You can add a new packet with the required attributes or modify attributes of a packet in the `display.drf` file or through the *Display Resource Editor*.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "transientRulerColorPacketName")
envSetVal("graphic" "transientRulerColorPacketName" 'string "ref_drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

transparentGroup

```
layout transparentGroup boolean { t | nil }
```

Description

Allows editing of groups without EIP into group when set to `t`. The variable setting is applicable only for the current window. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Transparent Group* (Display Options Form)

Examples

```
envGetVal("layout" "transparentGroup")
envSetVal("layout" "transparentGroup" 'boolean t)
envSetVal("layout" "transparentGroup" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trimEnable

```
layout trimEnable boolean { t | nil }
graphic trimEnable boolean { t | nil }
```

Description

Hides the layer that is cut using the `Trim` command. The default is `nil`.

GUI Equivalent

Command	*Options – Display*
Field	*Trim Shapes (*Display Options Form*)*

Examples

```
envGetVal("layout" "trimEnable")
envSetVal("layout" "trimEnable" 'boolean t)
envSetVal("layout" "trimEnable" 'boolean nil)
envGetVal("graphic" "trimEnable")
envSetVal("graphic" "trimEnable" 'boolean t)
envSetVal("graphic" "trimEnable" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trueColorProbe

```
graphic trueColorProbe boolean { t | nil }
```

Description

Determines the drawing mode for the next probe. The default is `nil`.

GUI Equivalent

Command	*Options – Highlight*
Field	*Drawing Mode* (Highlight Options Form)

Examples

```
envGetVal("graphic" "trueColorProbe")
envSetVal("graphic" "trueColorProbe" 'boolean t)
envSetVal("graphic" "trueColorProbe" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlBendFac

`layout trlBendFac float` *`float_number`*

Description

Specifies the maximum width (as a multiple of its standard width) that standard bends in a transmission line can achieve before they are automatically chamfered. This attribute is used only for bend, style bend. The default is `1.0` user units.

GUI Equivalent

Command	*Create – Microwave – Trl, Create – Microwave – Bend*
Field	*Bend Factor* (Transmission Line Form, Transmission Line Bend Form)

Examples

```
envGetVal("layout" "trlBendFac")
envSetVal("layout" "trlBendFac" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlBendStyle

```
layout trlBendStyle cyclic { bend | chamfer | radial }
```

Description

Specifies the types of bends used in creating a transmission line. The default is `bend`.

GUI Equivalent

Command	*Create – Microwave – Trl, Create – Microwave – Bend*
Field	*Bend Style* (Transmission Line Form, Transmission Line Bend Form)

Examples

```
envGetVal("layout" "trlBendStyle")
envSetVal("layout" "trlBendStyle" 'cyclic "bend")
envSetVal("layout" "trlBendStyle" 'cyclic "chamfer")
envSetVal("layout" "trlBendStyle" 'cyclic "radial")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlChamFac

`layout trlChamFac float` *`float_number`*

Description

Specifies the width of a chamfered bend in a transmission line as a multiple of its standard width. This attribute is used only for bend, style chamfer. The default is `0.6` user units.

GUI Equivalent

Command	*Create – Microwave – Trl, Create – Microwave – Bend*
Field	*Chamfer Factor* (Transmission Line Form, Transmission Line Bend Form

Examples

```
envGetVal("layout" "trlChamFac")
envSetVal("layout" "trlChamFac" 'float 0.8)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlNPtsToPI

`layout trlNPtsToPI int` *`any_positive_integer_in_user_units`*

Description

Specifies the number of segments in a radial bend in a transmission line as a function of the number of segments in a 180-degree radial bend. The default is `20`.

GUI Equivalent

Command	*Create – Microwave – Trl, Create – Microwave – Bend, Create – Microwave – Taper*
Field	*Resolution* (Transmission Line Form, Transmission Line Bend Form, Transmission Line Taper Form)

Examples

```
envGetVal("layout" "trlNPtsToPI")
envSetVal("layout" "trlNPtsToPI" 'int 30)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlRadFac

`layout trlRadFac float` *`float_number`*

Description

Specifies the radius of a radial bend in a transmission line as a multiple of the transmission-line width. This attribute is used only for bend-style radial. The default is `3.0` user units.

GUI Equivalent

Command	*Create – Microwave – Trl, Create – Microwave – Bend*
Field	*Radius Factor* (Transmission Line Form, Transmission Line Bend Form, Transmission Line Taper Form)

Examples

```
envGetVal("layout" "trlRadFac")
envSetVal("layout" "trlRadFac" 'float 5.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlWidth1

`layout trlWidth1 float` *`float_number`*

Description

Specifies the width of the first end of a transmission line bend. The default is `20` user units.

GUI Equivalent

Command	*Create – Microwave – Trl, Create – Microwave – Bend, Create – Microwave – Taper*
Field	*Width* (Transmission Line Form)
	Side 1 Width (Transmission Line Bend Form, Transmission Line Taper Form)

Examples

```
envGetVal("layout" "trlWidth1")
envSetVal("layout" "trlWidth1" 'float 30.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

trlWidth2

`layout trlWidth2 float` *`float_number`*

Description

Specifies the width of the second end of a transmission line bend. The default is `20` user units.

GUI Equivalent

Command	*Create – Microwave – Bend, Create – Microwave – Taper*
Field	*Side 2 Width* (Transmission Line Bend Form, Transmission Line Taper Form)

Examples

```
envGetVal("layout" "trlWidth2")
envSetVal("layout" "trlWidth2" 'float 30.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

turnInfixOffWhenSmartSnapping

```
layout turnInfixOffWhenSmartSnapping boolean { t | nil }
```

Description

Controls the infix mode when commands that use smart snapping are active. The default is `t`.

GUI Equivalent

Command	*Options – Editor*
Field	*Turn Off Infix in Smart Snapping* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "turnInfixOffWhenSmartSnapping")
envSetVal("layout" "turnInfixOffWhenSmartSnapping" 'boolean t)
envSetVal("layout" "turnInfixOffWhenSmartSnapping" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

undoRedoHistoryLimit

`layout undoRedoHistoryLimit int` *`integer_value`*

Description

Specifies the number of commands displayed in the drop-down menu of Undo and Redo commands. The default is `10`. Valid range is `5` to `20`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "undoRedoHistoryLimit")
envSetVal("layout" "undoRedoHistoryLimit" 'int 20)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

updateLabelLPPWithPin

```
layout updateLabelLPPWithPin boolean { t | nil }
```

Description

When this environment variable is set to `t`, whenever the LPP of a pin is changed to another LPP, the label attached to that pin changes to the new LPP. The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "updateLabelLPPWithPin")
envSetVal("layout" "updateLabelLPPWithPin" 'boolean t)
envSetVal("layout" "updateLabelLPPWithPin" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

updatePCellIncrement

`layout updatePCellIncrement` *`every_grid_snap`*

Description

Specifies how often the system updates Pcell parameters and regenerates a Pcell during a stretch operation. Grid snap is as defined by the technology file variable `mfgGridResolution`. The default is `-1.0`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "updatePCellIncrement")
envSetVal("layout" "updatePCellIncrement" 'float -2.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useAlphaBlending

```
graphic useAlphaBlending boolean { t | nil }
```

Description

Determines whether alpha blending is enabled (`t`) or not (`nil`).

GUI Equivalent

None

Examples

```
envGetVal("graphic" "useAlphaBlending")
envSetVal("graphic" "useAlphaBlending" 'boolean t)
envSetVal("graphic" "useAlphaBlending" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useDynamicHilightPacket

```
graphic useDynamicHilightPacket boolean { t | nil }
```

```
layout useDynamicHilightPacket boolean { t | nil }
```

Description

Creates the dynamicHilight display packet (`t`) and uses it to draw dynamic highlights in the design. When set to `nil`, the dynamic highlight is drawn with a nonconfigurable thin dashed outline. The default is `t` for layout, `nil` for others.

When the environment variable is set to `t`, and you do not have the dynamicHilight display packet defined in the display.drf file, then the packet is created internally on the first dynamically highlighted figure. However, this packet is not saved automatically in display.drf. Therefore, if you close Virtuoso at this stage, display.drf is not modified, and next time, you open Virtuoso and then DRE, the dynamicHilight display packet will not be displayed.

However, if you open DRE, save display.drf, and then, close Virtuoso, the next time you will open Virtuoso and DRE, the dynamicHilight displaypacket will be present, even if environment variable is set to `nil`.

GUI Equivalent

None

Examples

```
envGetVal("graphic" "useDynamicHilightPacket")
envSetVal("graphic" "useDynamicHilightPacket" 'boolean t)
envSetVal("graphic" "useDynamicHilightPacket" 'boolean nil)
envGetVal("layout" "useDynamicHilightPacket")
envSetVal("layout" "useDynamicHilightPacket" 'boolean t)
envSetVal("layout" "useDynamicHilightPacket" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useOverrideFontName

```
layout useOverrideFontName boolean { t | nil }
```

```
graphic useOverrideFontName boolean { t | nil }
```

Description

Determines whether to override the default font name (`t`), or not (`nil`) for a window. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "useOverrideFontName")
envSetVal("layout" "useOverrideFontName" 'boolean t)
envSetVal("layout" "useOverrideFontName" 'boolean nil)
envGetVal("graphic" "useOverrideFontName")
envSetVal("graphic" "useOverrideFontName" 'boolean t)
envSetVal("graphic" "useOverrideFontName" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useRelativeXY

```
layout useRelativeXY boolean { t | nil }
```

Description

When set to `t`, indicates that the points in the *XY* field are the points relative to the last entered points of the command (delta X and Y). The default is `nil`.

GUI Equivalent

Command	*Tools – Enter Points*
Field	*Use Relative XY* (Enter Points Form)

Examples

```
envGetVal("layout" "useRelativeXY")
envSetVal("layout" "useRelativeXY" 'boolean t)
envSetVal("layout" "useRelativeXY" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useSeparateLAndXlForms

```
layout useSeparateLAndXlForms boolean { t | nil }
```

Description

Specifies to use separate forms for VLS L and VLS XL. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "useSeparateLAndXlForms")
envSetVal("layout" "useSeparateLAndXlForms" 'boolean t)
envSetVal("layout" "useSeparateLAndXlForms" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useTapPurposeList

```
layout useTapPurposeList boolean { t | nil }
```

Description

Prioritizes the order in which tap cycles through layers. When you set Wire Auto Tap or Shape Auto Tap to on, and *Select from Overlaps* to off, the tap purpose list is used to prioritize the order in which layers are cycled through during tap. When Wire Auto Tap and Shape Auto Tap are set to off and *Select from Overlaps* is on, the *Choose object to tap from* form appears, so the Tap Purpose List is not considered. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "useTapPurposeList")
envSetVal("layout" "useTapPurposeList" 'boolean t)
envSetVal("layout" "useTapPurposeList" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

useTrueBBox

```
layout useTrueBBox boolean { t | nil }
graphic useTrueBBox boolean { t | nil }
```

Description

Specifies how instance outlines at the display stop level are determined for display. When set to `t`, one more level of hierarchy opens. If the instance's master contains a shape on the layer purpose "`instance`" "`drawing`", its bounding box displays as the outline. If the instance contains no shape on the layer purpose "`instance`" "`drawing`", the instance's master bounding box displays. Opening the master ensures an accurate display of the outline, but can affect loading and display time. When set to `nil`, the current instance bounding box displays without opening any additional hierarchy. This enables large designs to open and display much faster.

GUI Equivalent

Command	*Options – Display*
Field	*Display Controls: Use True BBox* (Display Options Form)

Examples

```
envGetVal("layout" "useTrueBBox")
envSetVal("layout" "useTrueBBox" 'boolean t)
envSetVal("layout" "useTrueBBox" 'boolean nil)
envGetVal("graphic" "useTrueBBox")
envSetVal("graphic" "useTrueBBox" 'boolean t)
envSetVal("graphic" "useTrueBBox" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

validLppFilterOn

```
layout validLppFilterOn boolean { t | nil }
```

Description

Specifies that only valid layers displayed in the *Layers* assistant are populated in the layer field of the *Property Editor*, *Move*, *Copy*, and *Label* forms.

GUI Equivalent

Command	*Options – Editor*
Field	*Use Layer Palette LPPs Only* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "validLppFilterOn")
envSetVal("layout" "validLppFilterOn" 'boolean t)
envSetVal("layout" "validLppFilterOn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAlignCuts

```
layout viaAlignCuts boolean { t | nil }
```

Description

Specifies the alignment of cuts between different tiles of a polygonal overlap. The default value is `t`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Align Cuts* (Create Via Form)

Examples

```
envGetVal("layout" "viaAlignCuts")
envSetVal("layout" "viaAlignCuts" 'boolean t)
envSetVal("layout" "viaAlignCuts" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoFromArea

```
layout viaAutoFromArea boolean { t | nil }
```

Description

Specifies the area, rectangular or polygonal, in which the layout editor will place the via(s) in all overlapping regions.The default is `t`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*From Area* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoFromArea")
envSetVal("layout" "viaAutoFromArea" 'boolean t)
envSetVal("layout" "viaAutoFromArea" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaAlignment

```
layout viaAutoViaAlignment cyclic {"auto" | "top-left" | "top-center" | "top-right"
     | "center-left" | "center-center" | "center-right" | "bottom-left" | "bottom-
     center" | "bottom-right"}
```

Description

Specifies the alignment of the via as either `auto`, `top-left`, `top-center`, `top-right`, `center-left`, `center-center`, `center-right`, `bottom-left`, `bottom-center`, `bottom-right`. The default is `auto`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Via Alignment* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaAlignment")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "auto")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "top-left")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "top-center")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "top-right")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "center-left")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "center-center")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "center-right")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "bottom-left")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "bottom-center")
envSetVal("layout" "viaAutoViaAlignment" 'cyclic "bottom-right")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaCutArrayDirection

```
layout viaAutoViaCutArrayDirection cyclic {"auto" | "horizontal" | "vertical"}
```

Description

Specifies the preferred direction for the cut box as either `auto`, `horizontal`, or `vertical`. The default is `auto`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Cut BBox Orientation* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaCutArrayDirection")
envSetVal("layout" "viaAutoViaCutArrayDirection" 'cyclic "auto")
envSetVal("layout" "viaAutoViaCutArrayDirection" 'cyclic "horizontal")
envSetVal("layout" "viaAutoViaCutArrayDirection" 'cyclic "vertical")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaEnableMaxStackDepth

```
layout viaAutoViaEnableMaxStackDepth boolean { t | nil }
```

Description

Specifies the maximum stack depth.The default is `nil`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Max Stack Depth* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaEnableMaxStackDepth")
envSetVal("layout" "viaAutoViaEnableMaxStackDepth" 'boolean t)
envSetVal("layout" "viaAutoViaEnableMaxStackDepth" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaEnclosureDirection

```
layout viaAutoViaEnclosureDirection cyclic {"XY" | "wire" | "minArea" |
     "horizontal" | "vertical"}
```

Description

Specifies the preferred enclosure direction for the via engine in auto mode as either `XY`, `wire`, `minArea`, `horizontal`, or `vertical`. This environment variable defines the way enclosures are set on the via when there are `minOppositeExtension` rules in the technology file.

- XY: Spcifies the X and Y coordiates of the via in the left right and top bottom direction. For example, `( minOppExtension "Metal1" "Via1" (0.0 0.03) )`. This rule means that the Metal1 enclosure around Via1 should be 0.0 in one direction and 0.03 in the other (opposite) direction.
- wire: The wire associated to the via is queried and if found, its direction is used to compute the enclosure direction, the higher enclosure value in the wire direction. For example, if the via Metal1-Metal2 connect two vertical pathsegs, the wire direction is vertical, thus 0.03 will be applied in Y (top/bottom) and 0 in X (left/right) for both the layers. In case the via connect two orthogonal pathsegs, the direction of the Metal1 pathseg is used to determine the enclosure direction for the rule, (`minOppExtension "Metal1" "Via1" (0.0 0.03) )`, and the Metal2 pathseg for the rule, `( minOppExtension "Metal2" "Via1" (0.0 0.03) )`.
- minArea: The enclosure direction is chosen to minimize the via area for each layer. Depending on the size of the `cutBox` , the enclosure will be applied in onedirection or the other. For example, for a 1x10 via (big via in X, small in Y) the `0.03` in `( minOppExtension "Metal1" "Via1" (0.0 0.03) )` will be applied in X because it leads to a smaller area of `Metal1` than setting the `0.03` in Y.
- horizontal: The higher enclosure value in the horizontal direction. In the example, `( minOppExtension "Metal1" "Via1" (0.0 0.03) )`, `0.03` is applied in left/right coordinates and `0` is applied in top/bottom coordinates.
- vertical: The higher enclosure value in the vertical direction. In the example, `( minOppExtension "Metal1" "Via1" (0.0 0.03) )`, `0.03` is applied in top/bottom coordinates and 0 is applied in left/right coordinates.

The default is `minArea`.

© 2020 All Rights Reserved.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaAutoViaEnclosureDirection")
envSetVal("layout" "viaAutoViaEnclosureDirection" 'cyclic "XY")
envSetVal("layout" "viaAutoViaEnclosureDirection" 'cyclic "wire")
envSetVal("layout" "viaAutoViaEnclosureDirection" 'cyclic "minArea")
envSetVal("layout" "viaAutoViaEnclosureDirection" 'cyclic "horizontal")
envSetVal("layout" "viaAutoViaEnclosureDirection" 'cyclic "vertical")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaHints

```
layout viaAutoViaHints boolean { t | nil }
```

Description

Displays the *Auto Via Preview Bindkeys* hint box on the canvas. The default is `nil`.

GUI Equivalent

Command	(In Layout XL) *Create – Via; Mode: Auto* and *Fast Edit*
Field	*Show Hints* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaHints")
envSetVal("layout" "viaAutoViaHints" 'boolean t)
envSetVal("layout" "viaAutoViaHints" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaMaxStackDepth

`layout viaAutoViaMaxStackDepth int` *`integer_number`*

Description

Specifies the value of the maximum stack depth. The default value is `1`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Max Stack Depth* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaMaxStackDepth")
envSetVal("layout" "viaAutoViaMaxStackDepth" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaMinNumCuts

`layout viaAutoViaMinNumCuts int` *`integer_number`*

Description

Specifies the minimum number of cuts to be automatically created for any overlapping area between two shapes. The default value is `1`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Min Number of Cuts* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaMinNumCuts")
envSetVal("layout" "viaAutoViaMinNumCuts" 'int 5)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaAutoViaPreviewShow

```
layout viaAutoViaPreviewShow boolean { t | nil }
```

Description

Displays a preview of the via that will be created. The default is `t`.

GUI Equivalent

Command	(In Layout XL) *Create – Via; Mode: Auto* and *Fast Edit*
Field	*Show Preview* (Create Via Form)

Examples

```
envGetVal("layout" "viaAutoViaPreviewShow")
envSetVal("layout" "viaAutoViaPreviewShow" 'boolean t)
envSetVal("layout" "viaAutoViaPreviewShow" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCheckBlockageTypes

```
layout viaCheckBlockageTypes boolean { t | nil }
```

Description

Specifies whether or not to create any via in case a blockage exists between the top and bottom metal of the current overlap. Only via and cut layer blockages are considered, metal layer blockages are not considered. The default value is `nil`.

GUI Equivalent

Command	*Create – Via*
Field	*Consider Blockages* (Create Via Form)

Examples

```
envGetVal("layout" "viaCheckBlockageTypes")
envSetVal("layout" "viaCheckBlockageTypes" 'boolean t)
envSetVal("layout" "viaCheckBlockageTypes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCheckMinNumCut

```
layout viaCheckMinNumCut boolean { t | nil }
```

Description

Determines whether the `minNumCut` constraint for the cut layers specified in the technology file is ignored (`nil`) or not (`t`) while creating a via. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaCheckMinNumCut")
envSetVal("layout" "viaCheckMinNumCut" 'boolean t)
envSetVal("layout" "viaCheckMinNumCut" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCheckPlateMaxWidth

```
layout viaCheckPlateMaxWidth boolean { t | nil }
```

Description

Determines whether the `maxWidth` constraint for the metal layers specified in the technology file is ignored (`nil`) or not (`t`) while creating a via. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaCheckPlateMaxWidth")
envSetVal("layout" "viaCheckPlateMaxWidth" 'boolean t)
envSetVal("layout" "viaCheckPlateMaxWidth" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCheckPlateMinWidth

```
layout viaCheckPlateMinWidth boolean { t | nil }
```

Description

Determines whether the `minWidth` constraint for the metal layers specified in the technology file is ignored (`nil`) or not (`t`) while creating a via. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaCheckPlateMinWidth")
envSetVal("layout" "viaCheckPlateMinWidth" 'boolean t)
envSetVal("layout" "viaCheckPlateMinWidth" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCheckShapeConn

```
layout viaCheckShapeConn boolean { t | nil }
```

Description

Specifies whether or not to look for connectivity information on the intersecting shapes. The default value is `nil`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Connect Shapes On The Same Net* (Create Via Form)

Examples

```
envGetVal("layout" "viaCheckShapeConn")
envSetVal("layout" "viaCheckShapeConn" 'boolean t)
envSetVal("layout" "viaCheckShapeConn" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCheckUserSet

```
layout viaCheckUserSet boolean { t | nil }
```

Description

Lets retain user specified parameters during via editing. When this environment variable is set to `t`, the Property Editor checks which via parameters are user specified and which have values based on the design rules. When this environment variable is set to `nil`, the Property Editor considers that all via parameters are user specified. So, when you change one via parameter, the other parameters do not change.

The default is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaCheckUserSet")
envSetVal("layout" "viaCheckUserSet" 'boolean t)
envSetVal("layout" "viaCheckUserSet" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaColumns

`layout viaColumns int` *`any_integer_greater_than_one`*

Description

Specifies the default number of columns of via to be cut in a via array. The *Columns* field on the Create Via form is updated based on the value you specify in this environment variable. The default number is `1`.

GUI Equivalent

Command	*Create – Via; Mode: Single*
Field	*Columns* (Create Via Form)

Examples

```
envGetVal("layout" "viaColumns")
envSetVal("layout" "viaColumns" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCompuEnclosures

```
layout viaCompuEnclosures boolean { t | nil }
```

Description

Automatically calculates enclosures when set to `t`. The default is `t`.

GUI Equivalent

Command	*Create – Via*
Field	*Enclosures: Compute* (Create Via Form)

Examples

```
envGetVal("layout" "viaCompuEnclosures")
envSetVal("layout" "viaCompuEnclosures" 'boolean t)
envSetVal("layout" "viaCompuEnclosures" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaComputeFromMode

```
layout viaComputeFromMode cyclic { "Row/Col" | "Width/Height" | "Shape(s)" | "Drawn
     Area" }
```

Description

Enables you to create auto vias at the intersection of shapes in the *Single* and *Stack* modes.

In the *Single* and *Stack* modes: You can select a *Via Definition* and specify the *Row* and *Column* Spacing, *Width* and *Height*, *Shape(s)* or, *Drawn Area* for the via. In the *Stack* mode, you can also specify the *Start Layer* and the *End Layer* of the via stack. So you can pre-define the viaDefs that will be used for each stacked via. The default is `Row/Col`.

GUI Equivalent

Command	*Create – Via*
Field	*Compute From* (Create Via Form)

Examples

```
envGetVal("layout" "viaComputeFromMode")
envSetVal("layout" "viaComputeFromMode" 'cyclic "Row/Col")
envSetVal("layout" "viaComputeFromMode" 'cyclic "Width/Height")
envSetVal("layout" "viaComputeFromMode" 'cyclic "Shape(s)")
envSetVal("layout" "viaComputeFromMode" 'cyclic "Drawn Area")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaConstraintGroup

`layout viaConstraintGroup string { name_of_constraint_group }`

Description

Specifies the constraint group from which to use valid vias for creating and editing vias.

GUI Equivalent

Command	*Options – Editor*
Field	*Wire Editing: Create Via: Via* (Layout Editor Options Form)

Examples

```
envGetVal("layout" "viaConstraintGroup")
envSetVal("layout" "viaConstraintGroup" 'string "Same as Wire")
envSetVal("layout" "viaConstraintGroup" 'string "foundry")
envSetVal("layout" "viaConstraintGroup" 'string "dsn:default")
envSetVal("layout" "viaConstraintGroup" 'string "wspRegions")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCreateInRoute

```
layout viaCreateInRoute boolean { t | nil }
```

Description

Enables the vias to be created in a new route in the *Single* (in Auto mode), *Stack* (in Auto mode), and *Auto* via creation modes. The default value is `t`.

GUI Equivalent

Command	*Create – Via – Options*
Field	*Create In Route* (Create Via Options Form)

Examples

```
envGetVal("layout" "viaCreateInRoute")
envSetVal("layout" "viaCreateInRoute" 'boolean t)
envSetVal("layout" "viaCreateInRoute" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCutSpacingMethod

```
layout viaCutSpacingMethod cyclic { "distribute" | "minimum" }
```

Description

Enables selection of the cut spacing method: either distribute (spread) or minimum (uses the minimum spacing rule constraints). The default value is `distribute`.

GUI Equivalent

Command	*Create – Via – Options*
Field	*Cut Spacing Method*(Create Via Options Form)

Examples

```
envGetVal("layout" "viaCutSpacingMethod")
envSetVal("layout" "viaCutSpacingMethod" 'cyclic "distribute")
envSetVal("layout" "viaCutSpacingMethod" 'cyclic "minimum")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaCutClassOrientation

```
layout viaCutClassOrientation cyclic { SameOrientation | Vertical | Horizontal |
     Auto }
```

Description

Enables you to specify the preferred orientation for the cut class as `SameOrientation`, `Vertical`, `Horizontal`, or `Auto`. The default is `Auto`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Cut Class Orientation* (Create Via Form)

Examples

```
envGetVal("layout" "viaCutClassOrientation")
envSetVal("layout" "viaCutClassOrientation" 'cyclic "SameOrientation")
envSetVal("layout" "viaCutClassOrientation" 'cyclic "Vertical")
envSetVal("layout" "viaCutClassOrientation" 'cyclic "Horizontal")
envSetVal("layout" "viaCutClassOrientation" 'cyclic "Auto")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaEnableUserSkillCallbacks

```
layout viaEnableUserSkillCallbacks boolean { t | nil }
```

Description

Controls the creation of a via based on SKILL callbacks. If the user callback returns `t`, the via is created. If it returns `nil`, the via is not created. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaEnableUserSkillCallbacks")
envSetVal("layout" "viaEnableUserSkillCallbacks" 'boolean t)
envSetVal("layout" "viaEnableUserSkillCallbacks" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaEnableValidPurposes

```
layout viaEnableValidPurposes boolean { t | nil }
```

Description

Enables the functionality to specify a list of valid purposes for via creation in automated modes: auto, fast edit, compute from shape: single, compute from shape: stack.

GUI Equivalent

Command	*Create – Via; Mode: Auto, Fast Edit; Options*
Field	*Valid Purposes* (Create Via Form)

Examples

```
envGetVal("layout" "viaEnableValidPurposes")
envSetVal("layout" "viaEnableValidPurposes" 'boolean t)
envSetVal("layout" "viaEnableValidPurposes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaExtendEnclosureBeyondOverlap

```
layout viaExtendEnclosureBeyondOverlap boolean { t | nil }
```

Description

Enables the via enclosures to be extended beyond the overlap region of the intersecting shapes in the *Single* (in Auto mode), *Stack* (in Auto mode), and *Auto* via creation modes. The default value is `t`.

GUI Equivalent

Command	*Create – Via – Options*
Field	*Extend Enclosure Beyond Overlap* (Create Via Options Form)

Examples

```
envGetVal("layout" "viaExtendEnclosureBeyondOverlap")
envSetVal("layout" "viaExtendEnclosureBeyondOverlap" 'boolean t)
envSetVal("layout" "viaExtendEnclosureBeyondOverlap" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaFastEditFromSelectedVias

```
layout viaFastEditFromSelectedVias boolean { t | nil }
```

Description

Enables you to use fast edit command from a set of selected vias.The default value is `t`.

GUI Equivalent

Command	(In Layout XL) *Create – Via; Mode: Fast Edit*
Field	*From Selected Vias* (Create Via Form)

Examples

```
envGetVal("layout" "viaFastEditFromSelectedVias")
envSetVal("layout" "viaFastEditFromSelectedVias" 'boolean t)
envSetVal("layout" "viaFastEditFromSelectedVias" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaFillEnclosureOverlap

```
layout viaFillEnclosureOverlap boolean { t | nil }
```

Description

Enables you to extend via enclosure such that it covers the complete overlap area between the two intersecting shapes. The default value is `t`.

GUI Equivalent

Command	*Create – Via – Options*
Field	*Fill Overlap with Enclosure* (Create Via Options Form)

Examples

```
envGetVal("layout" "viaFillEnclosureOverlap")
envSetVal("layout" "viaFillEnclosureOverlap" 'boolean t)
envSetVal("layout" "viaFillEnclosureOverlap" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaHideEnclosures

```
layout viaHideEnclosures boolean { t | nil }
```

Description

Toggles the display of the section in the form that sets the enclosure values in the single mode on the Create Via form. When set to `t`, the form is expanded and displays the enclosure values. When set ot `nil`, the enclosure values form is collapsed. The default is `t`.

GUI Equivalent

Command	*Create – Via; Mode: Single*
Field	*Show Enclosures / Hide Enclosures* (Create Via Form)

Examples

```
envGetVal("layout" "viaHideEnclosures")
envSetVal("layout" "viaHideEnclosures" 'boolean t)
envSetVal("layout" "viaHideEnclosures" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaIgnorePurposes

`layout viaIgnorePurposes string` *`purposes_to_ignore`*

Description

Enables you to specify the list of purposes to be ignored for via creation in *Auto* and *Fast Edit* modes, and when single or stack is selected in the *Compute From* field. The default is `""`.

GUI Equivalent

Command	*Create – Via; Mode: Auto, Fast Edit; Options*
Field	*Ignore Purposes* (Create Via Form)

Examples

```
envGetVal("layout" "viaIgnorePurposes")
envSetVal("layout" "viaIgnorePurposes" 'string "marker blockage")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaMaximizeCuts

```
layout viaMaximizeCuts boolean { t | nil }
```

Description

Lets you get a bigger cut size at the cost of more added metal outside the overlap.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaMaximizeCuts")
envSetVal("layout" "viaMaximizeCuts" 'boolean t)
envSetVal("layout" "viaMaximizeCuts" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaMode

```
layout viaMode cyclic { Single | Stack | Auto }
```

Description

Specifies the mode of via creation. `Single` creates a via. `Stack` creates a multi-layered via. `Auto` creates a via and automatically places it on the intersection of two paths. The default is `Single`.

GUI Equivalent

Command	*Create – Via*
Field	*Mode* (Create Via Form)

Examples

```
envGetVal("layout" "viaMode")
envSetVal("layout" "viaMode" 'cyclic "Single")
envSetVal("layout" "viaMode" 'cyclic "Stack")
envSetVal("layout" "viaMode" 'cyclic "Auto")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaParamCalcMode

```
layout viaParamCalcMode cyclic { minRulesAndViaDef | minRules | viaDefDefaults }
```

Description

Populates the form fields for the session after a viaDef is selected. The default is `minRulesAndViaDef`.

GUI Equivalent

Command	*Create – Via; Mode: Single*
Field	*Reset parameters to* (Create Via Form)

Examples

```
envGetVal("layout" "viaParamCalcMode")
envSetVal("layout" "viaParamCalcMode" 'cyclic "minRulesAndViaDef")
envSetVal("layout" "viaParamCalcMode" 'cyclic "minRules")
envSetVal("layout" "viaParamCalcMode" 'cyclic "viaDefDefaults")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaPreventDRCOnOverlap

```
layout viaPreventDRCOnOverlap boolean { t | nil}
```

Description

Prevents the creation of auto via on overlap if it introduces DRC violations. When `t`, auto via is not created on the overlap if DRC violations are introduced. When `nil`, auto via is created even if DRC violations are introduced on the overlap.

The default is `t`.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Prevent DRC Violations On The Overlap* (Create Via Form)

Examples

```
envGetVal("layout" "viaPreventDRCOnOverlap")
envSetVal("layout" "viaPreventDRCOnOverlap" 'boolean t)
envSetVal("layout" "viaPreventDRCOnOverlap" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaPreventDRCWithNeighbors

```
layout viaPreventDRCWithNeighbors boolean { t | nil}
```

Description

Prevents the creation of auto via if it introduces DRC violations with neighboring shapes. When `t`, auto via is not created if DRC violations are introduced with neighboring shapes. When `nil`, auto via is created even if DRC violations are introduced with the neighboring shapes.

The default is `nil`.

GUI Equivalent

Command	(In Layout XL) *Create – Via; Mode: Auto*
Field	*Prevent DRC Violations With Neighbors* (Create Via Form)

Examples

```
envGetVal("layout" "viaPreventDRCWithNeighbors")
envSetVal("layout" "viaPreventDRCWithNeighbors" 'boolean t)
envSetVal("layout" "viaPreventDRCWithNeighbors" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaPrintStatisticsReport

```
layout viaPrintStatisticsReport boolean { t | nil}
```

Description

Prints a statistics report for auto via in the form of a table in the CIW. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaPrintStatisticsReport")
envSetVal("layout" "viaPrintStatisticsReport" 'boolean t)
envSetVal("layout" "viaPrintStatisticsReport" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaPrintStatisticsReportByStackDepth

```
layout viaPrintStatisticsReportByStackDepth boolean { t | nil}
```

Description

Prints a statistics report sorted by stack depth for auto via in the form of a table in the CIW. The `viaPrintStatisticsReport` should be set to `t` for this environment variable to print the statistics report. The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaPrintStatisticsReportByStackDepth")
envSetVal("layout" "viaPrintStatisticsReportByStackDepth" 'boolean t)
envSetVal("layout" "viaPrintStatisticsReportByStackDepth" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaRows

`layout viaRows int` *`any_integer_greater_than_one`*

Description

Specifies the default number of rows of via to be cut in a via array. The *Via Rows* field on the Create Via form is updated based on the value you specify in this environment variable. The default number is `1`.

GUI Equivalent

Command	*Create – Via; Mode: Single*
Field	*Rows* (Create Via Form)

Examples

```
envGetVal("layout" "viaRows")
envSetVal("layout" "viaRows" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaSelectionMode

```
layout viaSelectionMode cyclic { unknown | ignoreLayers | useCutLayers |
     useNonCutLayers }
```

Description

Controls the selection of vias. You can set the environment variable to one of the following values:

- `unknown`: Prevents the selection of vias.
- `ignoreLayers`: Enables the selection of vias if the *Selectability* check box for *Vias* in the *Objects* assistant is selected.
- `useCutLayers`: Enables the selection of vias if the *Selectability* check boxes for *Vias* in the *Objects* assistant and for the respective cut layer in the *Layers* assistant are selected.
- `useNonCutLayers`: Enables the selection of vias if the *Selectability* check boxes for *Vias* in the Objects assistant are selected. Also, both top and bottom layer should be set to visible and at least one bottom or top layer should be selected in the *Layers* assistant.

The default is `ignoreLayers`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaSelectionMode ")
envSetVal("layout" "viaSelectionMode " 'cyclic "unknown")
envSetVal("layout" "viaSelectionMode " 'cyclic "ignoreLayers")
envSetVal("layout" "viaSelectionMode " 'cyclic "useCutLayers")
envSetVal("layout" "viaSelectionMode " 'cyclic "useNonCutLayers")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaStackColumns

`layout viaStackColumns int` *`any_integer_greater_than_one`*

Description

Specifies the number of columns in the top via of the via stack. The default number is `1`.

GUI Equivalent

Command	*Create – Via; Mode: Stack*
Field	*Top Columns* (Create Via Form)

Examples

```
envGetVal("layout" "viaStackColumns")
envSetVal("layout" "viaStackColumns" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaStackRows

`layout viaStackRows int` *`any_integer_greater_than_one`*

Description

Specifies the number of rows in the top via of the via stack. The default number is `1`.

GUI Equivalent

Command	*Create – Via; Mode: Stack*
Field	*Top Rows* (Create Via Form)

Examples

```
envGetVal("layout" "viaStackRows")
envSetVal("layout" "viaStackRows" 'int 10)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaStackSelection

```
graphic viaStackSelection boolean { t | nil }
```

Description

Toggles the via stack selection mode. When `t`, allows the via stack to be selected with a single click. The default is `t`.

GUI Equivalent

Command	*Options – Selection*
Field	*Via Stack* (Selection Options Form)

Examples

```
envGetVal("graphic" "viaStackSelection")
envSetVal("graphic" "viaStackSelection" 'boolean t)
envSetVal("graphic" "viaStackSelection" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaStackSelectionMatchingSizes

```
graphic viaStackSelectionMatchingSizes boolean { t | nil }
```

Description

Allows to select vias belonging to the same via stack from a set of multiple super-imposed via stacks that share the origin and adjacent layers. The default is `nil`.

GUI Equivalent

Command	*Options – Selection*
Field	*Vias with matching sizes between common layers* (Selection Options Form)

Examples

```
envGetVal("graphic" "viaStackSelectionMatchingSizes")
envSetVal("graphic" "viaStackSelectionMatchingSizes" 'boolean t)
envSetVal("graphic" "viaStackSelectionMatchingSizes" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaUseEffectiveCuts

```
layout viaUseEffectiveCuts boolean { t | nil }
```

Description

Disables the use of the `numCuts` parameter on the `cutClass` definition to optimize the total number of cuts for a via. When this environment variable is set to `t`, the via engine uses the total cut area for optimization. The default value is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaUseEffectiveCuts")
envSetVal("layout" "viaUseEffectiveCuts" 'boolean t)
envSetVal("layout" "viaUseEffectiveCuts" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaUseEntireOverlap

```
layout viaUseEntireOverlap boolean { t | nil}
```

Description

Enables you to use the entire overlap to create the via. If `t`, the entire overlap is used to create the via. If `nil`, an intersection between the user area and the overlap box is used to create the via.

GUI Equivalent

Command	*Create – Via; Mode: Auto*
Field	*Use Entire Overlap* (Create Via Form)

Examples

```
envGetVal("layout" "viaUseEntireOverlap")
envSetVal("layout" "viaUseEntireOverlap" 'boolean t)
envSetVal("layout" "viaUseEntireOverlap" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaUseRowColDefault

```
layout viaUseRowColDefault boolean { t | nil}
```

Description

Enables you to manage how to set the values in the *Rows* and *Columns* fields in the Create Via form. When `t`, lets the Create Via form use the edited values. When `nil`, you can force the rows and columns values by setting values in the environment variables, `viaRows` and `viaColumns`.

The default is `t`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaUseRowColDefault")
envSetVal("layout" "viaUseRowColDefault" 'boolean t)
envSetVal("layout" "viaUseRowColDefault" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaValidateDRCStandalone

```
layout viaValidateDRCStandalone boolean { t | nil }
```

Description

Specifies if the Create Via - Single and Create Via - Stack commands in the Create Via form can create a via with DRC violations. In general, not all metal shapes are present when creating a standalone via, so not all design rules can be checked. The valid values are `t` and `nil`. The default value is `nil`.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaValidateDRCStandalone")
envSetVal("layout" "viaValidateDRCStandalone" 'boolean t)
envSetVal("layout" "viaValidateDRCStandalone" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaValidPurposes

`layout viaValidPurposes string` *`any_valid_purposes`*

Description

Enables you to specify a list of valid purposes for via creation in automated modes: auto, fast edit, compute from shape: single, compute from shape: stack. The via is not be created even if one purpose of the overlap shapes does not belong to the list of valid purposes specified in this field.

GUI Equivalent

Command	*Create – Via; Mode: Auto, Fast Edit; Options*
Field	*Valid Purposes* (Create Via Form)

Examples

```
envGetVal("layout" "viaValidPurposes")
envSetVal("layout" "viaValidPurposes" 'string "drawing")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaWallFromSelectedShapesEnabled

```
layout viaWallFromSelectedShapesEnabled boolean { t | nil }
```

Description

Lets you use the wall functionality from the selected shapes.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaWallFromSelectedShapesEnabled")
envSetVal("layout" "viaWallFromSelectedShapesEnabled" 'boolean t)
envSetVal("layout" "viaWallFromSelectedShapesEnabled" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaWallKeepViasEnabled

```
layout viaWallKeepViasEnabled boolean { t | nil }
```

Description

Retains the existing vias when a wall is created.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaWallKeepViasEnabled")
envSetVal("layout" "viaWallKeepViasEnabled" 'boolean t)
envSetVal("layout" "viaWallKeepViasEnabled" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viaWallPreviewEnabled

```
layout viaWallPreviewEnabled boolean { t | nil }
```

Description

Enables the wall capability.

GUI Equivalent

None

Examples

```
envGetVal("layout" "viaWallPreviewEnabled")
envSetVal("layout" "viaWallPreviewEnabled" 'boolean t)
envSetVal("layout" "viaWallPreviewEnabled" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

viewNameToTypeList

`graphic viewNameToTypeList string` *`any_valid_view_name`*

Description

Specifies the correct view name associated with the default tool name in the Create New File form.

GUI Equivalent

Command	(CIW) File - New - Cellview
Field	View (New File Form)

Examples

```
envGetVal("graphic" "viewNameToTypeList")
envSetVal("graphic" "viewNameToTypeList" 'string "schematic")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

xSnapSpacing

`layout xSnapSpacing float` *`float_number`*

`graphic xSnapSpacing float` *`float_number`*

Description

Controls the minimum distance the pointer moves in the X direction. The layout default is `0.1` user units, the graphic default is `1.0` user units.

GUI Equivalent

Command	*Options – Display*
Field	*Grid Controls*: *X Snap Spacing* (Display Options Form)

Examples

```
envGetVal("layout" "xSnapSpacing")
envSetVal("layout" "xSnapSpacing" 'float 0.3)
envGetVal("graphic" "xSnapSpacing")
envSetVal("graphic" "xSnapSpacing" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

ySnapSpacing

`layout ySnapSpacing float` *`float_number`*

`graphic ySnapSpacing float` *`float_number`*

Description

Controls the minimum distance the pointer moves in the Y direction. The layout default is `0.1` user units, the graphic default is `1.0` user units.

GUI Equivalent

Command	*Options – Display*
Field	*Grid Controls*: *Y Snap Spacing* (Display Options Form)

Examples

```
envGetVal("layout" "ySnapSpacing")
envSetVal("layout" "ySnapSpacing" 'float 0.3)
envGetVal("graphic" "ySnapSpacing")
envSetVal("graphic" "ySnapSpacing" 'float 3.0)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

yankShape

```
layout yankShape cyclic { rectangle | polygon }
```

Description

Specifies the default shape used by `leHiYank`. The default is `rectangle`.

GUI Equivalent

Command	*Edit – Basic – Yank*
Field	*Yank Shape* (Yank Form)

Examples

```
envGetVal("layout" "yankShape")
envSetVal("layout" "yankShape" 'cyclic "rectangle")
envSetVal("layout" "yankShape" 'cyclic "polygon")
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

zoomToFig

```
layout zoomToFig boolean { t | nil }
```

Description

Specifies whether the cellview zooms to objects found in a search. The default is `nil`.

GUI Equivalent

Command	*Tools – Find/Replace*
Field	*Search Results: Zoom To Figure* (Find/Replace Form)

Examples

```
envGetVal("layout" "zoomToFig")
envSetVal("layout" "zoomToFig" 'boolean t)
envSetVal("layout" "zoomToFig" 'boolean nil)
```

Related Topics

List of Layout Editor Environment Variables

© 2020 All Rights Reserved.

Environment Variables Used Internally

The following graphic and layout editor environment variables are used internally by the software or are reserved for future use. Changing the values of these variables may cause the layout editor or other Cadence software to behave improperly or quit. Many of these variables are read-only and cannot be changed.

`coalesceLimit`

`coalesceRatio`

`confirmDelete`

`connStatus`

`criticality`

`cycleSnap`

`disableLayerFuncSnapToMfgGrid`

`eipRedraw`

`enterFunGravityTypeStr`

`extractShowDanglingMustConnectInstTerms`

`extractShowDanglingMustConnectInstTermsOnLayer`

`figOrder`

`filteredNetName`

`grMode`

`iterNum`

`jumpersOn`

`leFigGroupNamePrefix`

`markerShowIgnore`

`markNetDumpNetObj`

`markNetConnectedLayer`

`markNetStopLayer`

`markNetViaLayer`

`markNetHilightObj`

`midPointOf2TargetsMode`

© 2020 All Rights Reserved.

`minLabelSize`

`minRefreshArea`

`minStippleSize`

`moreDialogs`

`netName`

`outlineOnly`

`pickNCreate`

`pinNumBits`

`probeTextOn`

`propClass`

`pruneSize`

`routeMenuOn`

`toggle`

`transformMarkersOn`

`viewNameList`

© 2020 All Rights Reserved.

C

Bindkeys in Virtuoso Layout Suite

To view the bindkeys commands in Virtuoso Layout Suite, see Bindkeys.

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

D

Objects Criteria Table

Objects Criteria Table

Objects	Properties (Find)	Properties (Replace)	Condition/Logic	Values
PR boundary	no Add Criteria			
any conic	any radius		Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant/, Any Techlib Layer and list of Layers assistant/ .
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.

© 2020 All Rights Reserved.

any microstrip	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.
any shape	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	ROD name		Cyclic field. Choices are: == and !=	text

© 2020 All Rights Reserved.

	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.
area boundary	boundary name		Cyclic field. Choices are: == and !=	text
	property		Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST	text
		none	None	No input required.
array	cell name	cell name	Cyclic field. Choices are: == and !=	text
	lib name	lib name	Cyclic field. Choices are: == and !=	text
	rows	rows	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	columns	columns	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	deltaX	deltaX	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number

© 2020 All Rights Reserved.

	deltaY	deltaY	cyclic field. choices are: <, <=, >, >=, ==, and !=	number
	array name	array name	Cyclic field. Choices are: ==, and !=	text
		none	None	No input required.
bend	bend style	bend style	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: bend, chamfer, and radial.
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.
blockage	blockage type		Cyclic field. Choices are: == and !=	Cyclic field. Choices are: routing, placement, via, fill, slot, pin, feed, and screen.

© 2020 All Rights Reserved.

	layer name		Cyclic field. Choices are: == and !=	Combination field. Choices are: none, Unrouted, Canplace, Cannotoccupy, Group, Row, marker, changedLayer, designFlow, and unset.
	owner name		Cyclic field. Choices are: == and !=	text
	property		Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	halo		Cyclic field. Choices are: == and !=	Cyclic field. Choices are: True and False.
	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, mask4Color, blackColor, multiColor
		none	None	No input required.
cluster boundary	boundary name		Cyclic field. Choices are: == and !=	text
	cluster name		Cyclic field. Choices are: == and !=	text

© 2020 All Rights Reserved.

	property		Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
		none	None	No input required.
donut	any radius		Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	inner radius	inner radius	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	outer radius	outer radius	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant/, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.

© 2020 All Rights Reserved.

	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
ellipse	any radius		Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	radiusX	radiusX	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	radiusY	radiusY	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.

© 2020 All Rights Reserved.

	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
group	group name		Cyclic field. Choices are: == and !=	text
	property		Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
		none	None	No input required.
inst	cell name	cell name	Cyclic field. Choices are: == and !=	text
	lib name	lib name	Cyclic field. Choices are: == and !=	text
	inst name	inst name	Cyclic field. Choices are: == and !=	text
	view name	view name	Cyclic field. Choices are: == and !=	text
	orient	orient	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: R0, R90, R180, R270, MY, MYR90, MX, and MXR90.

© 2020 All Rights Reserved.

	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	cell type	cell type	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: none, block, blockRing, cover, coverBump, pad, padSpacer, core, coreSpacer, coreAntenna, corner, softMacro, via, blockBlackBox, and padAreaIO.
	cluster name	cluster name	Cyclic field. Choices are: == and !=	text
	physical only		Cyclic field. Choices are: == and !=	Cyclic field. Choices are: TRUE and FALSE.
	placement status	placement status	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Unplaced, None, Placed, Fixed, and Locked/Cover..
		none	None	No input required.
inst or array	cell name	cell name	Cyclic field. Choices are: == and !=	text
	lib name	lib name	Cyclic field. Choices are: == and !=	text
	view name	view name	Cyclic field. Choices are: == and !=	text
		none	None	No input required.

© 2020 All Rights Reserved.

label	text	text	Cyclic field. Choices are: == and !=	text
	font	font	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: stick, gothic, math, roman, script, euroStyle, swedish, fixed, and milspec.
	VDR	none	none	Choices are: Created by VDR, Created by user, Both
	justify	justify	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: centerCenter, lowerLeft, centerLeft, upperLeft, lowerCenter, upperCenter, lowerRight, centerRight, and upperRight.
	height	height	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	orient	orient	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: R0, R90, R180, R270, MY, MYR90, MX, and MXR90.
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: any valid layer, any Layers assistant, any techlib layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text

© 2020 All Rights Reserved.

	ROD name		Cyclic field. Choices are: == and !=	text
		none	None	No input required.
label or text display	text		Cyclic field. Choices are: == and !=	text
	font	font	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: stick, gothic, math, roman, script, euroStyle, swedish, fixed, and milspec.
	justify	justify	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: centerCenter, lowerLeft, centerLeft, upperLeft, lowerCenter, upperCenter, lowerRight, centerRight, and upperRight.
	height	height	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	orient	orient	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: R0, R90, R180, R270, MY, MYR90, MX, and MXR90.
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: any valid layer, any Layers assistant, any techlib layer, and list of Layers assistant.

© 2020 All Rights Reserved.

	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	ROD name		Cyclic field. Choices are: == and !=	text
		none	None	No input required.
path	path width	path width	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	path style	path style	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: truncate, extend, round, variable. **(ICADVM20.1 Virtuoso RF Solution Only)** curve
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	begin ext	begin ext	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	end ext	end ext	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number

© 2020 All Rights Reserved.

	ROD name		Cyclic field. Choices are: == and !=	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.
	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
pathseg	width	width	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	begin style	begin style	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: truncate, extend, variable, chamfer, and custom.
	end style	end style	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: truncate, extend, variable, chamfer, and custom.
	layer	layer	Cyclic field. Choices are: == and !=	cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.

© 2020 All Rights Reserved.

	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name		None	No input required.
	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
		none	None	No input required.
path or pathseg	width	width	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	layer	layer	Cyclic field. Choices are: == and !=	cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		Cyclic field. Choices are: == and !=	text

© 2020 All Rights Reserved.

	no net name	none	None	No input required.
	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
pin	term name	term name	Cyclic field. Choices are: == and !=	text
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	pin type		Cyclic field. Choices are: == and !=	Cyclic field. Choices are: input, output, inputOutput, switch, jumper, and unused.
	connected to net		Cyclic field. Choices are: == and !=	Cyclic field. Choices are: signal, ground, supply, clock, testLatch, analog, tieOff, tieHi, tieLo, scan, reset, optical, singleModeOptical, and multiModeOptical.

© 2020 All Rights Reserved.

	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
		none	None	No input required.
polygon	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	ROD name		Cyclic field. Choices are: == and !=	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.
	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color

© 2020 All Rights Reserved.

	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock
	(ICADVM20.1 Virtuoso RF Solution Only) curved	none	Cyclic field. Choices are: == and !=	TRUE
rectangle	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	ROD name		Cyclic field. Choices are: == and !=	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name	none	None	No input required.
	color	color	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: grayColor, mask1Color, mask2Color, mask3Color, and mask4Color
	color state	color state	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: unlock, lock

© 2020 All Rights Reserved.

row	row name		Cyclic field. Choices are: == and !=	text
	site def name		Cyclic field. Choices are: == and !=	text
	site def type		Cyclic field. Choices are: == and !=	Cyclic field. Choices are: pad and core.
	property		Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
		none	None	No input required.
snap boundary	no Add Criteria	none		
taper	taper style	taper style	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: linear and exponential.
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		net name	Cyclic field. Choices are: == and !=.
	no net name	none	None	No input required.

© 2020 All Rights Reserved.

text display	text		Cyclic field. Choices are: ==, and !=	text
	font	font	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: euroStyle, gothic, math, roman, script, stick, swedish, fixed, and milspec.
	justify	justify	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: centerCenter, lowerLeft, centerLeft, upperLeft, lowerCenter, upperCenter, lowerRight, centerRight, and upperRight.
	height	height	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	orient	orient	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: R0, R90, R180, R270, MY, MYR90, MX, and MXR90.
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property		Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	ROD name		Cyclic field. Choices are: == and !=	text

© 2020 All Rights Reserved.

		none	None	No input required.
trl	trl width	trl width	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	bend style	bend style	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: bend, chamfer, and radial.
	chamfer fac	chamfer fac	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	radius fac	radius fac	Cyclic field. Choices are: <, <=, >, >=, ==, and !=	number
	layer	layer	Cyclic field. Choices are: == and !=	Cyclic field. Choices are: Any Valid Layer, Any Layers assistant, Any Techlib Layer, and list of Layers assistant.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		net name	Cyclic field. Choices are: == and !=.
	no net name	none	None	No input required.

© 2020 All Rights Reserved.

via	via definition	via definition	Cyclic field. Choices are: == and !=	Cyclic field. Choices are populated from the technology file. **Note:** If you select *others* from the cyclic field, the text field next to it gets enabled. You can then enter a via name and combine it with a wild card character to get multiple matches. For example, if you enter `M2_M1*` as the search criterion, all vias with `M2_M1` in their names are displayed.
	property	property	Cyclic field. Choices are: <, <=, >, >=, ==, !=, EXIST, and !EXIST.	text
	net name		Cyclic field. Choices are: == and !=	text
	no net name		None	No input required.
	width	width	Cyclic field. Choices are: <, <=, ==, >=, >, and !=	number
	length	length	Cyclic field. Choices are: <, <=, ==, >=, >, and !=	number

© 2020 All Rights Reserved.

	rows	rows	Cyclic field. Choices are: <, <=, ==, >=, >, and !=	number
	columns	columns	Cyclic field. Choices are: <, <=, ==, >=, >, and !=	number
	row spacing	rowspacing	Cyclic field. Choices are: <, <=, ==, >=, >, and !=	number
	column spacing	column spacing	Cyclic field. Choices are: <, <=, ==, >=, >, and !=	number
	cutClass	cutClass	Cyclic field. Choices are: ==, !=, Others	text

© 2020 All Rights Reserved.

E

Layout Editor Assistants

This topic describes the following assistants available in the Virtuoso® Layout Suite L tool:

- Annotation Browser
- Dynamic Selection
- Palette Assistant
- Search
- Property Editor
- Navigator
- World View

© 2020 All Rights Reserved.

Annotation Browser

You access the *Annotation Browser* assistant by selecting *Window – Assistants – Annotation Browser.*

The *Annotation Browser* assistant enables you to view and manage the violation markers generated for the current design by different Virtuoso applications. It operates on the design window's current environment; if you modify this environment by editing a cellview in place or descending into a hierarchical cellview, the *Annotation Browser* is updated accordingly.

For more information about the *Annotation Browser* user interface, see Annotation Browser in the Virtuoso Layout Suite documentation.

© 2020 All Rights Reserved.

Dynamic Selection

Use the *Dynamic Selection* assistant to identify and select specific objects of interest from a densely-populated design window with many overlapping or hierarchical objects.

The assistant lists all the objects currently under the mouse pointer in the design display area, ordered by the layer-purpose pair on which a particular object is drawn. The display is updated whenever the mouse pointer is moved in the design display area.

You can freeze (and unfreeze) the display by pressing `Ctrl+h` when the pointer is positioned in the design display area. When the display is frozen, if you select an object in the *Dynamic Selection* assistant, the object is automatically cross-selected in the layout window. If you move the pointer over an object in the assistant, that object is dynamically highlighted in the layout window.

For hierarchical instances, groups, and ROD objects, the assistant displays the hierarchical path of the objects under the mouse pointer. It does not necessarily list all the objects contained in a particular hierarchical object, only those that were under the pointer when the display was frozen. The depth of the hierarchy shown is governed by the number of display levels set for the layout window in the Display Options Form.

Video

For a video demonstration of the *Dynamic Selection* assistant, see Virtuoso Dynamic Selection Assistant.

This section covers the following topics:

- Dynamic Selection Assistant Display
- Selecting an Object using the Dynamic Selection Assistant

© 2020 All Rights Reserved.

Dynamic Selection Assistant Display

For each object, the assistant indicates the layer-purpose pair on which the object is drawn and provides a short description of the object itself. Different fonts and background colors are used to display an object depending on its selection state.

The information presented depends on the object type.

Object Type	Information Provided
Instance	Name and master name
Instance array	Name, master name, and the number of rows and columns in the array
Via	Layer 1, layer 2 and the name of the net to which the via is attached
Via array	Layer 1, layer 2, the name of the net to which the via array is attached, and the number of rows and columns in the array

© 2020 All Rights Reserved.

Object Type	Information Provided
Shape	The layer-purpose on which the shape is drawn and the net to which it is attached
Label	The text attached to the label
Row	Site name and the number of sites used in the row
Group	Name and type
Guide	Layer 1, layer 2, and the name of the net to which the guide is attached.
Pin	Information on associated objects and the name of the net to which the pin is attached.
Blockages	Placement band routing blockages, and other related objects such as placement halos, routing, fill, and slot objects.
Routing	Transmission lines, bends and tapers, steiners, multipart paths
Boundary	All types of boundary; for example, area, snap, cluster, and so on.
Row Region	
Row Region	Name of the row region and row template
Row	Name and orientation of the row, site name, and the number of sites used in the row
Custom Row	Name, orientation, and height of the row
Rail shape/ Background shape	The layer-purpose on which the shape is drawn

Click the [+] icon to reveal the hierarchical objects under a particular node and the [-] icon to hide them again. This happens automatically if the *Auto expand* option is switched on in the

© 2020 All Rights Reserved.

Dynamic Selection Assistant Options Form. For hierarchical objects, *LPP*, *Data*, and *Hierarchy* columns are displayed in the Dynamic Selection assistant.

Note: The assistant does not necessarily list all the objects contained in a particular hierarchical object, only those under the mouse pointer.

To ensure that order of fields that you choose for the Dynamic Selection assistant and the show or hide state are retained even after you close the current Virtuoso session, save your preferences in the `*.ini` file, as shown below:

1. Right-click in the Dynamic Selection assistant.

© 2020 All Rights Reserved.

2. Select *Settings – Save*.

Note: You can save your settings in the `DSA.ini` file which gets loaded automatically when you launch the Dynamic Selection assistant. The `DSA.ini` file is saved by default in the `./.cadence/dfII/DSA` directory, unless referenced at some other location as defined in the `setup.loc` file.

Selecting an Object using the Dynamic Selection Assistant

To use the Dynamic Selection assistant:

1. From the layout window menu bar, choose *Windows – Assistants – Dynamic Selection*.
2. In the layout window, hover your pointer over the design objects you want to examine.

   The assistant lists all the objects under the pointer, provides a short description of each object and indicates the layer-purpose pair on which it is drawn.

   For more information on how object information is displayed, see Dynamic Selection Assistant Display.

© 2020 All Rights Reserved.

3. With pointer in the design display area, right-click to display the *Layout* context-sensitive menu and choose *Freeze DSA* or press `Ctrl+h` to freeze the display.

4. In the *Dynamic Selection* assistant, choose the objects you are interested in.

   The objects in question are cross-selected in the layout window, allowing you to perform the required operations only on those specific objects.

   Note: You can also expand and collapse hierarchy and use the commands in the context menu when the display is frozen. For more information, see Dynamic Selection Context-Sensitive Menu.

5. From the *Layout* context-sensitive menu, choose *UnFreeze DSA* or press `Ctrl+h` to unfreeze the display.

   The display now updates dynamically when you move your pointer in the layout window.

Viewing Color, State, and HCL Depth

For designs with color data, in addition to *LPP* and *Data*, *Color*, *State*, and *HCL Depth* columns are displayed in the Dynamic Selection assistant. In the *Color* column, the maskColor information of the shape is displayed. In the *State* column, the lock state of the shape is displayed. In case of vias in a design with color data, the *Color* and *State* columns are displayed for each shape within the via.

© 2020 All Rights Reserved.

You can also view the hierarchy depth of the color lock. The *Hierarchical Depth* column in the *Dynamic Selection* assistant represents the depth from the current cellview to the color.

The value 0 indicates that the color and lock of the shape can be edited at this level. The value 1 in the *Hierarchical Depth* column indicates that you have to descend one level below to edit the color and lock for the selected shape.

In the example below, the cellview has two instances, `I8 - CellB` and `I7 - CellA`. `CellB` has M2 shape with `mask1/locked` color and state. MPT coloring for this instance is set at the current cellview as the value of the *HCL Depth* column is `0`. `CellA` has M1 shape with `mask1/locked` color and state. MPT coloring for `CellA` is set one level below the current cellview as the value of the *HCL Depth* column is `1`.

© 2020 All Rights Reserved.

You can select the *Edit In Place Color Lock* option in the context-sensitive menu on the *Dynamic Selection* assistant to edit in place color and lock for the selected shape.

In the example below, there are three colors and three cellviews. The rectangle on the left is inside `test_bot` instance, instantiated in `test_mid`, which is instantiated in `test_top`. The lock color is set on this shape in `test_top`, then in `test_mid`, and then in `test_bot`, with different colors. The *HCL Depth* column displays values `0`, `1`, and `2`. The value of the *Color* column is `various`. The context-sensitive menu option, Edit in Place Color Lock displays a sub- menu so that you can edit in place in level `1` (`test_mid`) or `2` (`test_bot`).

© 2020 All Rights Reserved.

If you edit in place in level `1`, the HCL depths are `-1`, `0`, and `1` to reflect the lock states from the current edit level.

© 2020 All Rights Reserved.

Palette Assistant

For *Palette Assistant*, see Chapter 3, “Palette Assistant”.

© 2020 All Rights Reserved.

Search

You access the *Search* assistant by selecting *Window – Assistants – Search.*

The *Search* assistant provides a wide range of design search facilities, including dynamic context search categories, fast, keyword-directed searching with immediate feedback, and an intuitive user interface that makes searching for design data simple and productive. The *Search* assistant also provides a fast way to find which form contains which option.

For detailed information about the *Search* assistant user interface, see The Search Assistant and Toolbar in the *Virtuoso Schematic Editor User Guide.*

© 2020 All Rights Reserved.

Search Categories

The *Search Categories* pull-down is used to refine the scope of your search. For example, you can choose to only include Instances in your search query.

From the pull-down, you can also choose to:

- Click *Selected* at the bottom of the pull-down list to restrict a search query to only those objects that are selected in the current view.

© 2020 All Rights Reserved.

- Click *Options* at the bottom of the pull-down list to display the *Search Options* form. From here, you can select and deselect the categories of object that are to be displayed in the *Search Categories* pull-down.

© 2020 All Rights Reserved.

Property Editor

You access the *Property Editor* assistant by selecting *Window – Assistants – Property Editor.*

Note: This section describes only the extended *Property Editor* functionality in the layout editor. For detailed information about the *Property Editor* user interface, how to work with it and how to edit property values, see The Property Editor Assistant in the *Virtuoso Schematic Editor User Guide.*

The *Property Editor* assistant enables you to view and edit object property values on the components in your design. By default, each property is displayed in a separate table row in the *Property Editor*.

This section covers the following topics:

- Setting Nets on Instance Terminals
- Editing a cdsVia
- Editing Via Stacks
- Determining Objects in Route
- Editing Voltage Information

© 2020 All Rights Reserved.

Setting Nets on Instance Terminals

You can use the *Property Editor* assistant to set up the nets on the terminals of a selected instance. Select *Connectivity* from the *Property Category Selector* list and specify the nets in the *Property Value* column for each terminal on the instance.

By using the *Property Editor* assistant, you can set up multiple terminal of different instances at the same time.

© 2020 All Rights Reserved.

Editing a cdsVia

By using the *Property Editor* assistant, you can modify cdsVia parameters, including the metal layer and via layer purposes.

For more information about cdsVias, see Creating a cdsVia.

Editing Via Stacks

The *Property Editor* assistant displays a selected via stack as a single entity. To select a via stack, see Selecting Vias in a Via Stack.

© 2020 All Rights Reserved.

From the *Property Editor* assistant, you can edit the start or end layers of the via stack as well as update viaDefs used for each of the layer pairs. You can also modify the number of top rows and columns in the via stack, the net name, and the rotation of the via stack.

Via Stack of 4 Rows and 6 Columns

Note: Right-clicking a via stack displays the *Via* context-sensitive menu.

To edit individual vias of a via stack, select *Via* from the *Object Category Selector* list. This displays properties of all the vias in the via stack. Expand the required property node and update properties on any of the individual vias.

Determining Objects in Route

You can use the following two attributes in the *Property Editor* assistant to determine the route status of vias, paths, and pathsegs:

- *Route Status*

  It is editable. It can have the following values:

  Routed indicates normal routing status.

  Fixed indicates a route that should not be edited automatically, though it can be edited manually.

© 2020 All Rights Reserved.

Locked/Cover indicates a route that should not be edited either by automatic tools or manually. The route forms a part of the base array. To edit the route, change the routing status to *Routed* or *Fixed*.

- *Part of a Route Status*

  It is a non-editable attribute. It is set to FALSE if the selected via, path, or pathseg is not part of a route and set to TRUE if the via, path, or pathseg is part of a route.

The *Create Wire* command in VLS XL creates vias and pathsegs that are in route.

Editing Voltage Information

In Virtuoso, the voltage range information on a net is used to verify or enforce voltage dependent constraints, such as voltage dependent spacings (`minVoltageSpacing`). By using the *Property Editor* assistant, you can view and edit the voltage range information (`minVoltage`, `maxVoltage`) of net(s).

© 2020 All Rights Reserved.

Select one or more nets in the *Navigator* assistant as shown in the following figure.

Selected Nets

In the *Property Editor* assistant, the *Min Voltage* and *Max Voltage* are displayed for the selected net(s).

Min Voltage and Max Voltage for Individual nets can be edited

You can edit the *Min Voltage* and *Max Voltage* values for any or all of the nets.

© 2020 All Rights Reserved.

Navigator

You access the *Navigator* assistant by selecting *Window – Assistants – Navigator.*

The *Navigator* assistant is a dockable assistant pane that provides facilities to view objects across the design hierarchy using a tree representation.

The Navigator Tree is a hierarchical expandable/collapsible navigation system, used to display devices, nets, and pins that exist in the design area. It helps you navigate through a hierarchical design to find the components you are looking for. Selecting a component in the *Navigator* selects the corresponding device in the design area. Similarly, selecting a component in the design areas selects the component in *Navigator*.

If you select a net in the *Navigator* assistant, the physical shapes on the net are highlighted/probed in the design canvas. The shapes are only highlighted/probed but not selected. When you delete a net selected by using the *Navigator* assistant, it deletes the associated shapes on the net. The pins on the net are not deleted.

If you select a pin or an instance in the *Navigator* assistant, the design display area automatically pans to or zooms at the selected object if not currently visible. You can control whether to pan or zoom to the selected object by using the autoZoomMode environment

© 2020 All Rights Reserved.

variable. You can turn off the automatic panning or zooming by setting the autoZoomPan environment variable to `nil`.

Note: In the *Navigator* assistant, all nets that connect to an instance within the group are displayed as duplicate nets inside that group.

For detailed information about the *Navigator* user interface and how to work with the Navigator, see The Navigator Assistant in the *Virtuoso Schematic Editor User Guide*.

Video

For a video demonstration of the enahancements in the *Navigator* Assistant, see *Navigator Changes - Overview*.

Video

For a video demonstration on how to create and use new sets in the *Navigator* Assistant, see *How to Create and Use New Sets in Navigator*.

© 2020 All Rights Reserved.

World View

You can access the World View assistant by selecting *Window – Assistants – World View*.

The *World View* assistant is a navigation tool, especially useful in large designs. It shows you a complete picture of your entire design and marks the part of the design that is currently displayed in the drawing area.

For more information, see Using the World View Window.

© 2020 All Rights Reserved.

F

Commands and Toolbars in Read-Only Mode

The following sections list the layout editor commands and toolbar icons that are not available if you open a design in read-only mode:

- Commands
- Toolbar Icons

Commands

The following table lists the commands that are not available in read-only mode in the second and third columns.

Menu	Command/Sub-Menu	Command
File	Save	
	Discard Edits	
Edit	Undo	
	Redo	
	Move	
	Stretch	
	Delete	
	RepeatCopy	
	Rotate	Rotate
		Rotate Left
		Rotate Right
		Flip Horizontal
		Flip Vertical

© 2020 All Rights Reserved.

Menu	Command/Sub-Menu	Command
	Basic	Chop
		Merge
		Yank
		Paste
	Advanced	Reshape
		Split
		Modify Corner
		Size
		Attach/Detach
		Convert To Polygon
		Move Origin
		Align
	Hierarchy	Descend Edit
		Descend Read
		Return
		Return To Level
		Edit In Place
		Make Cell
		Flatten
	Group	Add to Group
		Remove from Group
		Ungroup
Create	Shape	Rectangle
		Polygon
		Path
		Circle
		Ellipse

© 2020 All Rights Reserved.

Menu	Command/Sub-Menu	Command
		Donut
		Geometric Wire
	Instance	
	Pin	
	Label	
	Via	
	Device	Mos
		Resistor
	Multipart Path	
	Guard Ring	
	P&R Objects	Blockage
		Row
		Custom Placement Area
		Clusters
		Track Patterns
		Width Spacing Patterns
		P&R Boundary
		Snap Boundary
		Area Boundary
		Cluster Boundary
	Group	
	Microwave	
Connectivity	Pins	
	Nets	Propagate
		Add Shape
		Remove Shape
Tools	Remaster Instances	

© 2020 All Rights Reserved.

Menu	Command/Sub-Menu	Command
	Create Pins From Labels	
	Layer Generation	

Toolbar Icons

- File Toolbar:

  The following icons are available on the File toolbar:

  - Open
  - Save (Unavailable in read-only mode)

- Edit toolbar:

  The following icons are available on the Edit toolbar:

  - Undo (Unavailable in read-only mode)
  - Redo (Unavailable in read-only mode)
  - Move (Unavailable in read-only mode)
  - Copy (Unavailable in read-only mode)
  - Stretch (Unavailable in read-only mode)
  - Delete (Unavailable in read-only mode)
  - RepeatCopy (Unavailable in read-only mode)
  - Edit Properties
  - Align
  - Rotate

© 2020 All Rights Reserved.

- Rotate Left (Unavailable in read-only mode)
- Rotate Right (Unavailable in read-only mode)
- Flip Horizontal (Unavailable in read-only mode)
- Flip Vertical (Unavailable in read-only mode)
- Create Group (Unavailable in read-only mode)
- Ungroup (Unavailable in read-only mode)

■ Align Toolbar

The following icons are available on the Align toolbar:

- Align Left
- Align Vertical
- Align Right
- Align Top
- Align Horizontal
- Align Bottom
- Align Spacing
- Orthogonal Spacing
- Use Reference Layer

■ View Toolbar

The following icons are available on the View toolbar:

- Zoom In
- Zoom Out
- Zoom to Fit
- Zoom to Selected

© 2020 All Rights Reserved.

- Create Toolbar

The following icons are available on the Create toolbar:

  - Create Instance
  - Create Label
  - Create Pin
  - Create Via
  - Create Wire
  - Create Bus
  - Zoom to Selected

- Optimize Toolbar

The following icons are available on the Optimize toolbar:

  - Execute Flow
  - Preferences
  - Browse Constraints
  - Copy Back to Layout

- Search Toolbar

The following icons are available on the Search toolbar:

  - Search

© 2020 All Rights Reserved.

- Options Toolbar

The following icons are available on the Options toolbar:

- Full Select
- Path Spine
- Via Stack
- Create/Edit Snap Mode: Orthogonal
- Transparent Group
- DRD Enforce
- DRD Notify
- DRD Post-Edit
- Constraint Aware Editing (Unavailable in read-only mode)
- Update connectivity information when design is modified (Unavailable in read-only mode)
- Selection Granularity (Unavailable in read-only mode)
- Info Balloon
- Display Stop Level
- Create Snap Mode: Orthogonal
- Edit Snap Mode: Orthogonal
- Dimming
- Snap Wires to Manufacturing Grid
- Dynamic Measurement
- Highlight Aligned Edges
- Layer Scope

© 2020 All Rights Reserved.

- Status Toolbar

The following icons are available on the Status toolbar:

- Selection Mode
- Number of nets selected (*Sel(N):n*)
- Number of instances selected (*Sel(I):n*)
- Number of objects selected (*Sel(O):n*)
- X and Y
- dX and dY
- Distance

- Guardring Toolbar

The following icons are available on the Guardring toolbar:

- Create Guardring by Wrapping
- Create Guardring from Path
- Create Guardring from Rectangle
- Create Polygon Contact Fill
- Convert to Polygon
- Chop
- Merge
- Tunnel
- Heal
- Reshape
- Split
- Clean Overlapping Contacts

© 2020 All Rights Reserved.

- Clean Guardring Cache

- Show Selection Info Toolbar

The following icons are unavailable on the Show Selection Info toolbar:

- PreSel/Select
- Object Name
- Attributes

- Go Toolbar

The following icons are unavailable on the Go toolbar:

- Back (Unavailable in read-only mode)
- Forward (Unavailable in read-only mode)
- Up (Unavailable in read-only mode)
- Top (Unavailable in read-only mode)

- Workspaces Toolbar

The following icons are available on the Workspaces toolbar:

- Workspace Configuration
- Save Workspace
- Toggle Assistants Visibility

© 2020 All Rights Reserved.

© 2020 All Rights Reserved.

G

Layout Editor Plug-Ins

This topic lists the plug-ins available in the Virtuoso® Layout Suite cockpit. Plug-ins are applications that can be dynamically added to the layout editor user interface, used to perform some specialized design task, and then removed again when no longer required.

The *Launch - Plugins* menu contains all the applications that are available from the layout editor. To use any application listed in the menu, you must first load it.

When you load a plug-in application, it appears in the layout editor, typically as a separate menu on the menu bar, but sometimes as a user interface docked on the left-hand side of the layout window. Some plug-ins open the related applications directly on the layout canvas itself.

© 2020 All Rights Reserved.

The figure below shows the *Plugins* submenu available in the *Launch* menu of Virtuoso Layout Suite.

The following table lists the plug-ins available in layout editor and provides links to further information on the association feature.

Plug-In	Description
Debug Abutment	Lets you debug the abutment using the Pcell IDE window. For more information, see *Debugging Pcells Abutment* in the *Virtuoso Parameterized Cell Reference*.
Debug CDF	Lets you debug CDF callback procedures using simple SKILL expressions for form initialization (`forminit`), form completion (`doneProc`), and any parameter field value modification (parameter callbacks). For more information, see *Debugging CDF* in the *Virtuoso Parameterized Cell Reference*.

© 2020 All Rights Reserved.

Plug-In	Description
Innovus	Adds the *Innovus* menu to the layout editor menu bar. The Innovus Implementation System is used as the design cockpit if a Verilog netlist is used for the top-level design. For more information, see the *Mixed Signal (MS) Interoperability Guide*.
Pcell	Adds the *Pcell* menu to the layout editor menu bar to let you create graphical Pcells. For more information, see *Creating Graphical Parameterized Cells* in the *Virtuoso Parameterized Cell Reference*.
Simulation: NC-Verilog	Opens the Virtuoso Verilog Environment for NC-Verilog Integration application. For more information, see the *Virtuoso NC-Verilog Environment User Guide*.
Simulation: Other	Adds the *Simulation* menu to the layout editor menu bar. For more information, see the *Virtuoso NC-Verilog Environment User Guide*.
Simulation: Spectre	Adds the *Spectre* menu to the layout editor menu bar. This application provides a solution for digital circuit designers to netlist a Virtuoso transistor-level schematic for the Spectre circuit simulator. For more information, see the *Virtuoso Schematic Editor Spectre and Ultrasim Interface User Guide*.
Simulation: UltraSim	Adds the *UltraSim* menu to the layout editor menu bar. This product provides a solution for digital circuit designers to netlist a Virtuoso transistor-level schematic for the Virtuoso UltraSim circuit simulator. For more information, see the *Virtuoso Schematic Editor Spectre and Ultrasim Interface User Guide*.
VPS-L	Adds the *VPS-L* menu to the layout editor. The Virtuoso Power System L application provides a view of IR drop, electromigration (EM), and signal net EM analysis. For more information, see the *Power IR/EM User Guide*.
Voltus-Fi-L	Opens the Voltus-Fi-L application, which uses the extracted view to perform IR drop and Electromigration (EM) analysis. For more information, see the *Voltus-Fi Custom Power Integrity Solution User Guide*.

© 2020 All Rights Reserved.

Plug-In	Description
Voltus-Fi-XL	Opens the Voltus-Fi-XL application, which performs multi-mode simulation (MMSIM), electromigration, and voltage drop (EMIR) analysis for power and signal nets using the simulation database generated by Spectre APS/XPS. For more information, see the *Voltus-Fi Custom Power Integrity Solution User Guide*.

© 2020 All Rights Reserved.