Virtuoso® Layout Suite Frequently Asked Questions

Product Version ICADVM20.1
October 2020

Printed in the United States of America.

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

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.

Contents

Preface 5

Scope 6

Licensing Requirements 6

Related Documents 6

What's New and KPNS 6

Installation, Environment, and Infrastructure 6

Technology Information 7

Virtuoso Layout Suite Tools 7

Relative Object Design and Inherited Connections 8

SKILL Documents 8

Additional Learning Resources 9

Video Library 9

Virtuoso Videos Book 9

Rapid Adoption Kits 9

Help and Support Facilities 10

Customer Support 10

Feedback about Documentation 10

Typographic and Syntax Conventions 11

Virtuoso Layout Suite XL: Basic Editing Frequently Asked Questions 13

How do I enable the Used option in the Layers panel of the Palette assistant for every layout view that I open? 13

How can I ensure that a specific layer is used to trace the net using the Mark Net functionality? 13

How can I change the number of rows and columns during a subsequent Copy operation? 15

How can I replace cellviews hierarchically in a design? 15

How can I list the hierarchy from top to bottom? 16

When I stretch a path that overlaps a shape, why does the path remain connected to the shape? 16

Why does the Zoom In command take a long time to refresh the layout if the layout has multiple large sized rulers? 16

How can I set the Auto mode in the Create Label command as the default label creation mode? Also, why do we need to press F3 to display the Create Label form? 17
When I create pins, why are the labels not being displayed? 18
How can I create pins by reading data from a text file? 18
How can I set the default via on the Create Via form 19
How can I map the parameters on the System tab while creating a custom via? . . . 21
How can I set the default value for viaRows and viaColumns to two cuts? 21
Why does the ruler disappear from screen when I save the design if the Save Rulers option is not selected? 22
How can I cycle through the snapping targets in the Create Ruler command? 24
How can I find the cds environment variables corresponding to the fields in the Display Options and Layout Editor Options forms? 25
How can I define the layout cellview Selection Box? 25

Preface

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

This user guide describes how to use the Layout tool. 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

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node (for example, ICADVM20.1) releases.

Label	Meaning
(ICADVM20.1 Only)	Features supported only in the ICADVM20.1 release and which require either the Virtuoso Advanced Node Option for Layout Standard license (95512) or the Virtuoso Advanced Node Option for Layout license (95511).
(ICADVM20.1 EXL Only)	Features supported only in the ICADVM20.1 release and which require the Virtuoso Layout Suite EXL license (95800)
(IC6.1.8 Only)	Features supported only in the IC6.1.8 release.

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 Suit L What’s New*
- *Virtuoso Layout Suite L 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*

Technology Information

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

Virtuoso Layout Suite Tools

IC6.1.8 Only

- *Virtuoso Layout Suite L User Guide*
- *Virtuoso Layout Suite XL User Guide*
- *Virtuoso Layout Suite GXL Reference*

ICADVM20.1 Only

- *Virtuoso Layout Viewer User Guide*
- *Virtuoso Layout Suite XL: Basic Editing User Guide*
- *Virtuoso Layout Suite XL: Connectivity Driven Editing Guide*
- *Virtuoso Layout Suite EXL*
- *Virtuoso Design Planner User Guide*
- *Virtuoso Multi-Patterning Technology User Guide*
- *Virtuoso Placer User Guide*
- *Virtuoso Width Spacing Patterns User Guide*

IC6.1.8 and ICADVM20.1

- *Virtuoso Abstract Generator User Guide*
- *Virtuoso Custom Digital Placer User Guide*
- *Virtuoso Design Rule Driven Editing User Guide*
- *Virtuoso Floorplanner User Guide*
- *Virtuoso Fluid Guard Ring User Guide*

- *Virtuoso Interactive and Assisted Routing User Guide*
- *Virtuoso Module Generator User Guide*
- *Virtuoso Space-based Router User Guide*
- *Virtuoso Symbolic Placement of Devices 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*
  - ❑ *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:

- 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

  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.

`[ ]` — 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.

Virtuoso Layout Suite XL: Basic Editing Frequently Asked Questions

This document lists the frequently asked questions and answers related to Virtuoso Layout Suite.

How do I enable the Used option in the Layers panel of the Palette assistant for every layout view that I open?

In the `.cdsenv` file, set the `pteUsedOnlyScope` environment variable to `t`.

```
layout pteUsedOnlyScope boolean t
```

How can I ensure that a specific layer is used to trace the net using the Mark Net functionality?

I am using the Mark Net functionality for tracing a net in Virtuoso Layout Editor. The Mark Net functionality starts marking the nets using *y0 drawing* layer, then *y1 drawing* and so on till it reaches *y9 drawing*. After *y9 drawing*, it again starts marking the nets on *y0 drawing*. This is because by default the *Cycle Through System Colors Automatically* option in the MarkNet Options form is selected. If I disable this option using the `.cdsenv` variable, the net is always marked using the *y0 drawing* layer. Is there a way that the Mark Net functionality uses the *y4 drawing* layer by default for marking the nets?

You can use the following steps to make the Mark Net functionality use a specific layer to trace the nets:

1. Set the `markNetAutoColorCycle` and `markNetThickline` environment variables in the appropriate `.cdsenv` files as shown below:

   ```
   layout markNetAutoColorCycle boolean nil
   ```

   The above environment variable disables the *Cycle Through System Colors Automatically* option.

   ```
   layout markNetThickline boolean t
   ```

The above environment variable sets the mark net thick option to `t`.

Note: This option has been enabled in IC6.1.6 and above releases.

2. Load the SKILL code listed below in CIW or through the `.cdsinit` file. It defines the bindkey `F10` that can be used to reset the MarkNet Options form so that it sets the default layer in the form.

```
load " CCSFixMarkNetColor.il"

procedure(CCSFixMarkNetColor()

let(()

;;Change the index of nth function to specify the specific layer to use in the
order of y0-y9, like for y0 nth(0..), for y1 nth(1…) etc and so on.

hiRegTimer("le0MarkNetForm->optionTabs->page1->markNetColor->value=nth(4
le0MarkNetForm->optionTabs->page1->markNetColor->choices)" 1)

leHiMarkNet()

t

) ;let

) ;procedure CCSFixMarkNetColor


hiSetBindKey("Layout" "F10" "CCSFixMarkNetColor()")
```

3. Open the layout window and press the `F10` bindkey to invoke the mark net and trace a net. The `y4 drawing` layer will be used to mark the nets.

Note: In IC 6.1.6 and above releases, new SKILL functions `leMarkNet`, `leSaveMarkNet`, `leUnmarkNet` are available. In IC6.1.6.500.1 and above, and ICADV12.1.500.3 and above, the following function can also be used to mark the nets on a specific `y0 drawing - y9 drawing` layers.

```
leMarkNet('(5.7 103.0) ?startLevel 0 ?stopLevel 32 ?thickLine t ?markNetColor
"y2")

procedure(CCSMarkNet()

enterPoint(

?prompts list("Click on a point to Mark the Net")

?doneProc "CCSTraceNet"

);enterPoint

);procedure CCSMarkNet


procedure(CCSTraceNet(win done pt "wgl")

let(()

when(done

geGetWindowCellView(win)

pt=car(pt)

printf("Clicked point is %L\n" pt)

leMarkNet(pt ?startLevel 0 ?stopLevel 32 ?thickLine t ?markNetColor "y2")
```

```
    printf("Marked the Net\n")
    ); when
    );let
    );procedure CCSTraceNet

    hiSetBindKey("Layout" "<Key>F10" "CCSMarkNet()")
    hiSetBindKey("Layout" "<Key>F11" "leUnmarkNet()")
```

How can I change the number of rows and columns during a subsequent Copy operation?

To change the number of rows and columns in the subsequent copy operation, you can set the environment variable `copyResetRowCol`, in the home directory or the `.cdsenv` file.

```
layout copyResetRowCol boolean t
```

If the environment variable `copyResetRowCol` is set to `nil`, the last values used for rows and columns are used for the next Copy operation. However, if the `copyResetRowCol` environment variable is set to `t`, the values of rows and columns in the Enter Points form get reset. The default value of this environment variable is `nil`.

You can also set the `copyResetRowCol` variable in the `.cdsinit` file using the following command:

```
leSetEnv("copyResetRowCol" t)
```

How can I replace cellviews hierarchically in a design?

I need to replace a cellview hierarchically in a design. For example, there is a cell, `MYCAP`, with multiple layout views, `layout_A` and `layout_B`. The `layout_A` of `MYCAP` cell is instantiated in the cell, `MY_CAP_ARRAY`, in the library, `testlib`. How can I swap `layout_A` with `layout_B` in the cell `MY_CAP_ARRAY`?

You can use the sample SKILL API is shown below to replace cellviews hierarchically in a design:

```
CCSTop("tstlib" "MY_CAP_ARRAY" "layout" "MYCAP" "layout_A" "layout_B" '("refLibA"
"refLibB") )
```

In the above SKILl API, `refLibA` and `refLibB` are read-only reference libraries.

How can I list the hierarchy from top to bottom?

I used the command, *Edit – Hierarchy – Tree* with the *Top to bottom* option selected, but the complete hierarchy is not listed.

The *Display Level* settings in the Display Options form limits the number of levels listed by the `Tree` command. If the *Display Level* is set to 0 through 3, even if you select the *Top to bottom* option on the Tree form, the output will be for levels 0 through 3 only. So, before running the `Tree` command with the *Top to bottom* option selected, change the *Display Level* value to 32.

When I stretch a path that overlaps a shape, why does the path remain connected to the shape?

I created a rectangle and then created a path that is on the same layer and overlaps the rectangle. When I try to stretch the path, the end of the path remains connected to the rectangle. I have deselected the *Maintain Connections* option on the Layout Editor Options form.

You should use the *Keep Wires Connected To* option on the Stretch form to specify the connectivity option. If you have not selected this option, the path will not remain connected to the rectangle.

Why does the Zoom In command take a long time to refresh the layout if the layout has multiple large sized rulers?

In IC6.1.5 ISR13 and above releases, set the `useTheNewMeasurementImpl` environment variable to `t` to activate the new drawing code for the rulers. This will reduce the time to refresh.

```
graphic useTheNewMeasurementImpl boolean t
```

Or through the `cdsinit` file or CIW:

```
envSetVal("graphic" "useTheNewMeasurementImpl" 'boolean t)
```

In IC6.1.6 and above releases, the graphic `useTheNewMeasurementImpl` environment variable is not available and has been removed from the `cdsenv` file of the hierarchy. In these releases, the ruler behavior is controlled by the layout `rulerImpl` environment variable that is set to sting `new`, by default. Also, in IC6.1.6 and above releases, OA supports scratch cellviews and can create transient rulers. A transient ruler is a ruler that cannot be saved. However, by default, persistent rulers are created in the layout editor. A persistent ruler can be saved. If you want to create transient rulers, deselect the *Create Ruler as a savable object* option on the *General* tab of the *Create Ruler* form.

How can I set the Auto mode in the Create Label command as the default label creation mode? Also, why do we need to press F3 to display the Create Label form?

You can control default mode for the Create Label command by setting the `labelMode` option environment variable in the home directory or the `.cdsenv` file, as shown below:

```
layout   labelMode   cyclic  "auto"
```

You can also set it through the `.cdsinit` file:

```
envSetVal("layout"  "labelMode"  'cyclic  "auto")
```

Also, the `leHiCreateLabel` SKILL API supports a second argument through which you can pass on the mode in which you want to launch the Create Label command.

```
leHiCreateLabel(hiGetCurrentWindow()  "auto")
```

You can also permanently change the bindkey settings to set the auto mode as the default mode for the Create Label command, as shown below:

```
hiSetBindKey("Layout" "<Key>L"  "leHiCreateLabel(hiGetCurrentWindow()  \"auto\")")
```

Note: In IC614 and above releases, you need to press F3 to open the Create Label form. As a workaround, you can define a SKILL function and a bindkey to display the form by default, as shown below:

```
;---------------
procedure(CCScreateLabel()
envSetVal("ui" "showOptionForms" 'boolean t)
leHiCreateLabel()
envSetVal("ui" "showOptionForms" 'boolean nil)
)
hiSetBindKey("Layout" "<Key>L" "CCScreateLabel")
 ;--------------
```

The `L` bindkey can be used to invoke the Create Label command.

When I create pins, why are the labels not being displayed?

You should enable the *Pin Names* option on the Display Options form to display the labels for pins. You can also enable the display of pin names by setting the `displayPinNames` environment variable in the `.cdsenv` file, as shown below:

```
layout displayPinNames boolean t
```

How can I create pins by reading data from a text file?

I need to automate the pin creation process by reading an ASCII/text file as an input. The input file contains details of pin coordinates, layer (lpp), pin names, and access direction.

You can use the steps mentioned below to create pins by reading data from a text file. A sample input text file and SKILL code are also shown below:

Input text file (`indata`):

```
->layer purpose shape points termName termDirection accessDirection<-

"Poly" "drawing" "rectangle" 1.7 1.5 2.4 2.6 "my_pin1" "inputOutput"
"top"

"Metal1" "drawing" "rectangle" 3.2 3.3 4.3 4.8 "my_pin2" "input"
"bottom"

"Metal2" "drawing" "rectangle" 5.1 5.2 6.4 6.7 "my_pin3" "output"
"none"
```

A sample SKILL code that can be used for pin creation is shown below:

```
;----------------CCSCreatePin.il---------------------------------------------
---------------
procedure(CCSCreatePin(pinListfile libName cellName viewName)
 let((z inPort cvId nextline layer purpose lpp shape p1 p2 p3 p4 points termName
termDir accessDir)
  inPort = infile(pinListfile)
  cvId=dbOpenCellViewByType(libName cellName viewName "" "a")
  when( inPort
   while(z=gets(nextline inPort)
   layer=readstring(nth(0 parseString(z)))
   purpose=readstring(nth(1 parseString(z)))
   lpp=list(layer purpose)
   shape=readstring(nth(2 parseString(z)))
   p1=readstring(nth(3 parseString(z)))
   p2=readstring(nth(4 parseString(z)))
   p3=readstring(nth(5 parseString(z)))
   p4=readstring(nth(6 parseString(z)))
   points=list(p1:p2 p3:p4)
   termName=readstring(nth(7 parseString(z)))
   termDir=readstring(nth(8 parseString(z)))
   accessDir=list(readstring(nth(9 parseString(z))))
   leCreatePin(cvId lpp shape points termName termDir accessDir)
   )
  close(inPort)
  );when
 );let
);procedure
```

You can use the following steps to create pins by reading data from a text file:

1. Load the SKILL script `CCSCreatePin.il` through CIW or `.cdsinit` file. Ensure that the layout cellview exists.
2. Call the below mentioned SKILL procedure:

   ```
   CCSCreatePin("./indata" "library" "cell" "layout")
   ```

   You should specify the first argument as the input file name, followed the library name, cell name, and view name of the layout cellview where the pins need to be created.

How can I set the default via on the Create Via form

I need to change the default value of Via Definition in the Create Via form. I need to set the via `M5_M4v` as the default via. Whenever I open a new layout window, by default, the `M5_M4v` via should appear in the *Via Definition* field.

The Create Via form reads the constraint group set in *Create Via* section of Layout Editor Options form. The list of valid vias for this constraint group are populated in `Via Definition` field.

You can use the below mentioned SKILL code to change the default via in *Via Definition* field.

```
procedure(CCSRegProc(args)
    hiRegTimer("CCSSetDefVia(args->window)" 5)
    leHiCreateVia()
) ;proc
```

```
procedure(CCSSetDefVia(window)
    let((cv cstG currCG techId validVias)
      cancelEnterFun('le0ViaForm)
        cv=geGetWindowCellView(window)
        if(envGetVal("layout" "viaConstraintGroup")=="Same as Wire" then
               currCG=envGetVal("layout" "wireConstraintGroup")
              else
              currCG=envGetVal("layout" "viaConstraintGroup")
          ) ;if
      techId=techGetTechFile(cv)
      cstG=cstFindConstraintGroupIn(techId currCG)
      validVias=car(setof(x cstG~>objects  x~>defName=="validVias"))~>value

       if(member("M5_M4v" validVias) then
          le0ViaForm->viaSingleModeScrollView->viaDefName->value="M5_M4v"
         ) ;if
) ;let
) ;procedure


deRegUserTriggers("maskLayout" nil nil 'CCSRegProc)
deRegUserTriggers("maskLayoutXL" nil nil 'CCSRegProc)


hiSetBindKey("Layout" "<Key>o" "CCSSetDefVia(hiGetCurrentWindow())
leHiCreateVia()")
```

Use the following steps to use the above mentioned SKILL code:

1. Save the above code in a file, for example, `CCSChangeDefaultVia.il`.
2. Add the load command to load this file to your `.cdsinit` file: `load"<path_to_file>"/CCSChangeDefaultVia.il"`.
3. Start Virtuoso and open any layout cellview. If the constraint group set for the Create Via option in the Layout Editor Options form has `M5_M4v` in the list of valid vias then it will become the default via in the *Via Definition* field.
4. Change the constraint group for Create Via in the Layout Editor Options form. Then, open the Create Via form using the bindkey `o`. After this, you may use either the bindkey or the menu option.

How can I map the parameters on the System tab while creating a custom via?

While creating a custom via, to change rows and columns, I need to specify the values in the *Rows* and *Columns* fields on the *User defined* tab instead of the *System* tab of the Create Via form. Is there a way to setup the pcell using the *Rows* and *Columns* options on the *System* tab instead of the *User defined* tab?

You can use the *Rows* and *Columns* values of the System tab if you specify the parameter names in the via pcell code as `row` and `column`. Also, for mapping the pcell via, use the following names to map the system parameters: `row`, `column`, `w`, `l`, `xCutSpacing`, `yCutSpacing`, `xBias`, and `yBias`.

Note: If the pcell via has the enclosure information, it will be displayed under the *User Defined* tab and user can modify the value only on this tab.

How can I set the default value for viaRows and viaColumns to two cuts?

In IC6.1.5 ISR9, I have set the following three variables in the appropriate `.cdsenv` or `.cdsinit` file, to set the default values of rows and columns to two cuts.

`.cdsenv` file

```
layout viaRows int 2
layout viaColumns int 2
layout viaUseRowColDefault boolean nil
```

Or

`.cdsinit`

```
envSetVal("layout" "viaRows" 'int 2)
envSetVal("layout" "viaColumns" 'int 2)
envSetVal("layout" "viaUseRowColDefault" 'boolean nil)
```

After starting the Virtuoso session, the values for these variables are set. However, when the *Create - Via* command is invoked in the layout window, the defined values of `viaRows` and `viaColumns` are reset to `1`, as shown in the `CDS.log` file below:

`CDS.log file`

```
\i envGetVal("layout" "viaRows" )
\t 2
\i envGetVal("layout" "viaColumns" )
\t 2
\p >
```

```
\a leHiCreateVia()
\o Loading via.cxt
\o Loading layers.cxt
\p >
\i envGetVal("layout" "viaRows" )
\t 1
\i envGetVal("layout" "viaColumns" )
\t 1
\p >
```

In the above log file, the value for `viaRows` and `viaColumns` changed to `1` after invoking the `leHiCreateVia()` API. The first time the `leHiCreateVia()` SKILL API is invoked, the variables `viaRows` and `viaColumns` are reset to `1`. If `envSetVal` is used to define the values after this, the required number of rows and columns are actually used. The above variable settings used to work fine in IC6.1.4 release. How to can I get the default values to be set first time?

The above mentioned issue has been fixed in IC6.1.5 ISR11 and above releases. If you are using IC6.1.5 release or IC6.1.5 ISR1 to IC6.1.5 ISR10, you will get this issue.This is because the default settings of these variables in the `.cdsenv` file in the hierarchy are:

```
viaUseRowColDefault = t, viaRows =1 and viaColumns = 1
```

The above environment variables work for the `Create Via` command in *Single* mode. When the environment variable `viaUseRowColDefault` is set to `t`, the row and column defined in the `viaDef` of the technology file are populated every time `leHiCreateVia()` is invoked. When `viaUseRowColDefault` is set to `nil`, the row and column defined by the environment variables `viaRows` and `viaColumns` are populated in the Create Via form. The row and column values overridden by you get dynamically updated to the environment variables `viaRows` and `viaColumns`.

For IC5.1.4.1 release, the variables names are listed below:

```
layout contactResetRowCol boolean nil
layout contactRows int 2
layout contactColumns int 2
```

Why does the ruler disappear from screen when I save the design if the Save Rulers option is not selected?

I am using the smart ruler functionality of IC6.1.4, where the ruler provides extra capability of snapping and saving rulers into the design database. When I disable the *Save Rulers* option on the Layout Editor Options form, the rulers are not saved in the design. However, it also clears the rulers from the display whenever I try to save the design. As a layout designer, I often use the *Save* command to save my layout design work but I do not want to save rulers in database. Therefore, I do not select the *Save Rulers* option on the Layout Editor Options form. The rulers are not visible in the design which is not the desired behavior. Is there a way to keep the rulers on screen but not save them in the database when the design is saved using the *Save* command?

You can retain the rulers on the screen and not save them in the database with the help of SKILL code mentioned below:

Load the below mentioned SKILL code through the `.cdsinit` file or CIW:

```
load "CCSrestoreRulers.il"
```

This code ensures that the rulers are not saved in the database but are displayed on the screen even if the design is saved. After loading this file, pre-save and post-save triggers are registered. These triggers work for layout views only. The SKILL code enables the `saveRulers` environment variable, by default. This environment variable enables the *Save Rulers* option on the Layout Editor Options form. However, if you want to save the rulers, use the `F9` bindkey. Once the *Save Rulers* option is selected on the Layout Editor Options form, the pre-save trigger is disabled.

If you have enabled the *Save Rulers* option on the Layout Editor Options form, to get back to the pre-save trigger mode, use the `F10` bindkey so that the pre-save trigger is registered once again.

The below mentioned SKILL code uses the global variable `rulerPtListGLOBAL`.

```
*/
/* Presave trigger for maskLayout views */
procedure(CCSgrabRulers(cv)
let((currentRulers)
if(cv~>cellViewType=="maskLayout"
     then
;; Get all the ruler ID's
    currentRulers=setof(sh cv~>markers sh~>objType=="ruler")
;; If rulers are in design then store ruler points in global list
         if(currentRulers then
              rulerPtListGLOBAL=foreach(mapcar rulerId currentRulers
                                                          rulerId~>points
                                                            )
;; Delete all the rulers
               foreach(rulerId currentRulers dbDeleteObject(rulerId))
;; If no rulers then make sure that global list is empty
                        else
                        rulerPtListGLOBAL=nil
                        )
                )
        )
) ;; Pre save proc ends
/* Postsave trigger for maskLayout views */
procedure(CCSrestoreRulers(cv)
if(cv~>cellViewType=="maskLayout" && rulerPtListGLOBAL
then
;; Stop the cellview modification counter otherwise cell will be inconsistent
dbDisableCellViewCounter(cv)
;; Recreate all the rulers from global list
foreach(rulerPt rulerPtListGLOBAL leCreateRuler(cv rulerPt))
                )
;; After creation empty the global list
rulerPtListGLOBAL=nil
;; Enable cellview modification counter for normal working
dbEnableCellViewCounter(cv)
)  ;; Post save proc ends
/* Procedure for truly saving rulers */
procedure(CCStrulySaveRulers()
dbUnregSaveTrigger('CCSgrabRulers)
)
```

Press the *F9* key before the save operation for pre-save trigger mode:

```
hiSetBindKey("Layout" "<Key>F9" "CCStrulySaveRulers()")
```

Press the *F10* key to ensure that the rulers are not saved:

```
hiSetBindKey("Layout" "<Key>F10" "dbRegSaveTrigger('CCSgrabRulers)")
```

Ensure that the `saveRulers` environment variable is set to true:

```
leSetEnv("saveRulers" t)
```

Register the pre and post-save triggers:

```
dbRegSaveTrigger('CCSgrabRulers)
dbRegPostSaveTrigger('CCSrestoreRulers)
```

How can I cycle through the snapping targets in the Create Ruler command?

Press `Spacebar` to cycle through the snapping targets in the *Create Ruler* and *Quick Align* commands.

How can I find the cds environment variables corresponding to the fields in the Display Options and Layout Editor Options forms?

You can find the cds environment variables corresponding to the fields in the Display Options and Layout Editor Options forms by searching the field in the Search Assistant. You access the *Search* assistant by selecting *Window – Assistants – Search.*

How can I define the layout cellview Selection Box?

A cell is selectable when the cursor moves over an invisible layer near the cell boundary. Can this invisible layer be changed so that the selectability of the cell is very tiny? For example, the selectability is only the lower left corner of the cell. When the cursor moves over the cell, it will not be selectable, or dynamically highlighted till the cursor is on the lower left corner of the cell.

You should enable the *Use True BBox* option on the Display Options form to create user-defined instance or drawing shape to display the outline of the instance instead of the cellview bbox.