

# **Virtuoso Technology Data User Guide**

**Product Version ICADVM20.1  
October 2020**

© 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.

---

# Contents

---

<u>Preface</u> .....	9
<u>Scope</u> .....	9
<u>Licensing Requirements</u> .....	10
<u>Related Documentation</u> .....	10
<u>What's New and KPNS</u> .....	10
<u>Installation, Environment, and Infrastructure</u> .....	10
<u>Technology Information</u> .....	11
<u>Virtuoso Tools</u> .....	11
<u>Relative Object Design and Inherited Connections</u> .....	12
<u>Additional Learning Resources</u> .....	12
<u>Video Library</u> .....	12
<u>Virtuoso Videos Book</u> .....	13
<u>Rapid Adoption Kits</u> .....	13
<u>Help and Support Facilities</u> .....	13
<u>Customer Support</u> .....	14
<u>Feedback about Documentation</u> .....	14
<u>Typographic and Syntax Conventions</u> .....	14
<u>1</u>	
<u>About Virtuoso Technology Data</u> .....	17
<u>Incremental Technology Databases</u> .....	17
<u>Simple Incremental Technology Database Structures</u> .....	20
<u>User-defined Reserved Layers</u> .....	24
<u>Providing Like Data for Different Design Purposes</u> .....	25
<u>Multiple Design Libraries Using Incremental Technology Databases</u> .....	27
<u>Designing Incremental Technology Database Graphs</u> .....	29
<u>Database References</u> .....	29
<u>Database Reference Ordering</u> .....	29
<u>Database Content Targeting</u> .....	30
<u>Conflict Avoidance</u> .....	30
<u>Conflict Correction</u> .....	32

# Virtuoso Technology Data User Guide

---

<u>Defining and Creating Technology Databases</u>	33
<u>ASCII Technology and Display Resource Files</u>	33
<u>Compiling ASCII Technology Files into Technology Databases and Graphs</u>	34
<u>Setting up Library Access to Technology Libraries</u>	34
<u>Querying Incremental Technology Bases with SKILL</u>	35

## 2

<u>ASCII Technology File Development</u>	37
<u>Methods of Initial ASCII File Creation</u>	37
<u>Technology File Organization</u>	38
<u>Which Technology Data Does Your Application Use?</u>	38
<u>Specifying Data for An Incremental Technology Database</u>	39
<u>Creating a Technology Database from a Technology File</u>	40

## 3

<u>Display Resource File Development</u>	41
<u>Overview of Development and Usage</u>	42
<u>How Cadence Design Software Handles Multiple Display Resource Files</u>	42
<u>Planning Display Resource File Updates for Proper Merging</u>	43
<u>Methods of Initial Display Resource File Creation</u>	45
<u>Display Resource File Organization</u>	45
<u>How Display Packets Control Layer Display</u>	48

## 4

<u>Preparing Files for Use with a Design</u>	51
<u>The Technology File Manager and User Interface</u>	52
<u>Invoking the Technology File Manager</u>	52
<u>Technology File Manager Commands</u>	53
<u>Displaying the Incremental Technology Database Graph</u>	55
<u>Effective and As Defined Graph Displays</u>	56
<u>Conflicts in Graph Displays</u>	58
<u>Generating a New Technology Library</u>	60
<u>Compiling an ASCII Technology File into a Technology Database (Library)</u>	60
<u>Creating a New Technology Library from an Existing Technology Library</u>	63

## Virtuoso Technology Data User Guide

<u>Creating a New Technology Library that References Existing Technology Libraries</u>	64
<u>Referencing or Attaching a Technology Library</u>	67
<u>Referencing a Technology Library from a Design Library</u>	68
<u>Attaching a Technology Library to a Design Library</u>	68
<u>Ensuring Desired Display Resource File Usage</u>	69

## 5

### Editing, Reusing, and Merging Technology File Data..... 71

<u>Updating Technology Files and Technology Libraries</u>	72
<u>The Techfile IDE</u>	73
<u>Starting the Techfile IDE</u>	73
<u>tfEditTechfile</u>	78
<u>Creating a Technology File</u>	78
<u>Navigating through a Technology File</u>	79
<u>Editing a Technology File</u>	81
<u>Working with the Rule Editor</u>	87
<u>Checking a File for Errors</u>	92
<u>Loading a Technology File into a Technology Library</u>	94
<u>Environment Variables for the Techfile IDE</u>	94
<u>Reusing Technology Data to Build a New Technology Library</u>	96
<u>Creating an ASCII File from a Technology Library to Edit and Load Changes</u>	96
<u>Copying a Technology Library to Use As a Basis for Creating a New Technology Library</u>	100
<u>Loading Technology Data into Virtual Memory</u>	101
<u>Merging New ASCII Technology Data into an Existing Technology Library</u>	102
<u>Replacing Existing Technology Data in a Technology Library</u>	104
<u>Discarding an Edited Technology Database from Virtual Memory</u>	106
<u>Saving a Technology Library Edited in Virtual Memory to Disk</u>	107

## 6

### Editing, Reusing, and Merging Display Resources..... 109

<u>Using Display Resource Manager</u>	110
<u>Opening Display Resources Tool Box</u>	110
<u>Using Display Resources Tool Box</u>	110
<u>Editing Display Resources</u>	110

## Virtuoso Technology Data User Guide

---

<u>Merging Display Resource Files</u>	111
<u>Using Display Resource Editor</u>	114
<u>Accessing Display Resource Data through Display Resource Editor</u>	117
<u>Deleting Display Resources</u>	127
<u>Loading a Display Resource File in Virtual Memory</u>	131
<u>Saving Display Resource Data to a File</u>	132
<u>Reloading Source Display Resource Files</u>	134
<u>Setting Selection Display Colors and Dynamic Highlights</u>	136
<u>Editing Display Resource Data with SKILL Functions</u>	137
<u>Editing a Display Resource File</u>	139
<u>Testing a Display Resource File</u>	141

## 7

<u>Defining and Installing Devices</u>	143
<u>Bringing up the Install Device Form</u>	144
<u>Guard Rings</u>	146
<u>About Guard Rings</u>	146
<u>About the Install Device Form for Guard Rings</u>	150
<u>Defining and Installing a New Guard Ring</u>	158
<u>Modifying an Existing Guard Ring</u>	167
<u>Deleting a Guard Ring</u>	168

## A

<u>Form Descriptions</u>	169
<u>New Technology Library Form</u>	170
<u>SKILL Function to Display Form</u>	170
<u>Reference Existing Technology Libraries Form</u>	171
<u>SKILL Function to Display Form</u>	171
<u>Technology Database Graph Form</u>	172
<u>SKILL Function to Display Form</u>	172
<u>Attach Technology Library to Design Library Form</u>	173
<u>SKILL Function to Display Form</u>	173
<u>Load Technology File Form</u>	174
<u>SKILL Functions to Display Form</u>	174
<u>Dump Technology File Form</u>	175

## Virtuoso Technology Data User Guide

---

<u>SKILL Function to Display Form</u>	177
<u>Technology File – Layer Browser Forms</u>	178
<u>Discard Edits To Technology File Form</u>	179
<u>SKILL Function to Display Form</u>	179
<u>Save Technology File Form</u>	180
<u>SKILL Function to Display Form</u>	180
<u>Install Device Form</u>	181
<u>SKILL Function to Display Form</u>	181
<u>Merge Display Resource Files (DRF) Form</u>	182
<u>Display Resource Editor (DRE) Form</u>	183
<u>SKILL Function to Display Form</u>	183
<u>Color Editor Form</u>	184
<u>Stipple Editor Form</u>	185
<u>Line Style Editor Form</u>	186
<u>Load Form</u>	187
<u>Save As Form</u>	188

## B

<u>Environment Variables</u>	189
<u>Working with Environment Variables</u>	189
<u>Checking the Value of an Environment Variable</u>	189
<u>Changing the Default Settings of Environment Variables</u>	189
<u>Copying Environment Variables from the Samples Directory to ~/.cdsenv</u>	190
<u>Technology Data Environment Variables</u>	191
<u>AlternateFoundryCG</u>	191
<u>defaultAttachTech</u>	192
<u>noTechUpRev</u>	193

## C

<u>Data Handling for Incremental Technology Databases</u>	195
<u>Major Data Duplication Rules Summary</u>	195
<u>Data Handling Table</u>	196
<u>CDS_ALLOW_VIRTSYSLAY Environment Variable</u>	200

## D

<u>Incremental Technology File Examples</u> .....	203
<u>Example 1</u> .....	203
<u>Single ASCII Technology File</u> .....	203
<u>ITDB ASCII Technology Files Derived from the Single File</u> .....	206
<u>Example 2</u> .....	212



---

# Preface

---

This manual provides information about how to create and maintain a technology file and display resource file, both of which define the technology information you need to create and view your designs. This manual assumes you are familiar with the development and design of integrated circuits.

This preface contains the following topics:

- [Scope](#)
- [Licensing Requirements](#)
- [Related Documentation](#)
- [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 and methodologies (for example, ICADVM20.1) releases.

Label	Meaning
(ICADVM20.1 Only – 95512)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Adv_Node_Opt_Lay_Std license (95512).
(ICADVM20.1 Only – 95511)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Adv_Node_Opt_Layout license (95511).

# Virtuoso Technology Data User Guide

## Preface

---

(ICADVM20.1 Only – 95511 and 95800)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Adv_Node_Opt_Layout (95511) and Virtuoso_Layout_Suite_EXL (95800) licenses.
(ICADVM20.1 Only – Virtuoso MultiTech Framework)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_MultiTech_Framework (95022) license.
(ICADVM20.1 Photonics Only)	Features supported only in the ICADVM20.1 release and which require the Virtuoso_Photonics_Option license (95550).
(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 Guide\*](#).

## Related Documentation

### What's New and KPNS

- [\*Virtuoso Technology Data What's New\*](#)
- [\*Virtuoso Technology Data 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 ASCII Files Reference\*](#)
- [\*Virtuoso Technology Data Constraints Reference\*](#)
- [\*Virtuoso Technology Data User Guide\*](#)
- [\*Virtuoso Technology Data SKILL Reference\*](#)

## Virtuoso 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 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\*](#)

## **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 Electrically Aware Design Flow Guide\*](#)
- [\*Virtuoso Floorplanner User Guide\*](#)
- [\*Virtuoso Fluid Guard Ring User Guide\*](#)
- [\*Virtuoso Interactive and Assisted Routing User Guide\*](#)
- [\*Virtuoso Layout Suite SKILL Reference\*](#)
- [\*Virtuoso Module Generator User Guide\*](#)
- [\*Virtuoso Parameterized Cell Reference\*](#)
- [\*Virtuoso Pegasus Interactive User Guide\*](#)
- [\*Virtuoso Space-based Router User Guide\*](#)
- [\*Virtuoso Symbolic Placement of Devices User Guide\*](#)
- [\*Virtuoso Voltage Dependent Rules Flow Guide\*](#)

## **Relative Object Design and Inherited Connections**

- [\*Virtuoso Relative Object Design User Guide\*](#)
- [\*Virtuoso Relative Object Design SKILL Reference\*](#)
- [\*Virtuoso Schematic Editor User Guide\*](#)

## **Additional Learning Resources**

### **Video Library**

The [Video Library](#) on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

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

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

## Virtuoso Videos Book

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

## Rapid Adoption Kits

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

To explore the full range of training courses provided by Cadence in your region, visit [Cadence Training](#) or write to [training\\_enroll@cadence.com](mailto: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.

<i>text</i>	Indicates names of manuals, menu commands, buttons, and fields.
-------------	---

# Virtuoso Technology Data User Guide

## Preface

---

<code>text</code>	Indicates text that you must type exactly as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
<code>z_argument</code>	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, <code>z_</code> ) indicates the data type the argument can accept and must not be typed.
<code> </code>	Separates a choice of options.
<code>{ }</code>	Encloses a list of choices, separated by vertical bars, from which you <b>must</b> choose one.
<code>[ ]</code>	Encloses an optional argument or a list of choices separated by vertical bars, from which you <b>may</b> choose one.
<code>[ ?argName t_arg ]</code>	Denotes a <i>key argument</i> . 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.
<code>...</code>	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.
<code>, ...</code>	Indicates that multiple arguments must be separated by commas.
<code>=&gt;</code>	Indicates the values returned by a Cadence® SKILL® language function.
<code>/</code>	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 Technology Data User Guide**

## **Preface**

---



---

# About Virtuoso Technology Data

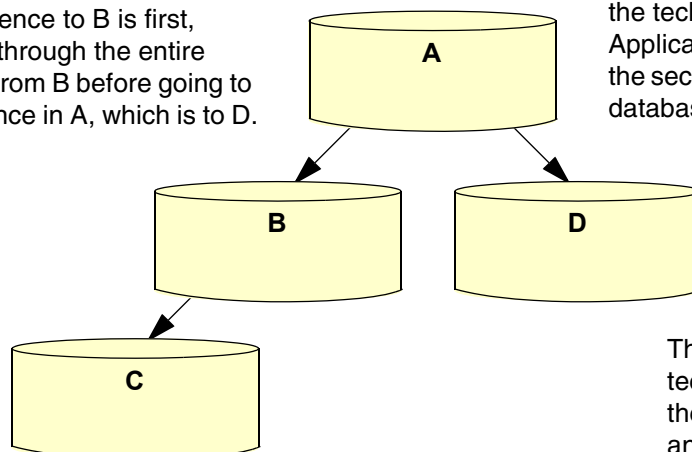
---

Virtuoso<sup>®</sup> technology databases define the process data used throughout a design flow: layer definitions, layer attributes, design constraints, site definitions, via definitions, via specifications, and device definitions—all of the information that defines the framework for creating designs. Most Virtuoso design technology data is initially defined in ASCII technology files which are compiled into technology databases, or libraries. These technology databases can then be set up as incremental technology databases.

## Incremental Technology Databases

The Virtuoso incremental technology database structure enables applications to combine technology data from different sources by establishing references between databases. A database inherits the technology data from any databases it references. A chain of references, called a *graph*, creates an effective technology database that is a combination of the technology data in all of the databases, as illustrated below. Multiple references in a technology database are ordered: The order in which they are listed in the referencing technology database dictates the order in which applications traverse the graph to find technology data.

A first references B, which references C, and so A inherits technology data from B, which in turn inherits from C. Because the reference to B is first, software will look through the entire graph emanating from B before going to the second reference in A, which is to D.



A references D second, and so inherits the technology data from D. Applications go to D last because it is the second reference in A and is the last database in the graph.

The result is that all of the technology data from A, B, C, and then D is effective when A is in use and at the top of the graph.

## Virtuoso Technology Data User Guide

### About Virtuoso Technology Data

---

The ability to establish references among technology databases allows the incremental inclusion of technology databases as needed by various applications throughout the design process. Technology databases can be supplied by multiple sources (for example, the foundry, IP provider, and designer) and can be tailored to meet the requirements of different tasks in the design flow (for example, device-level design as opposed to top-level interconnect creation). Avoiding the limitations placed on design teams when a single read-only technology database is supplied, this structure provides flexibility by allowing designers to

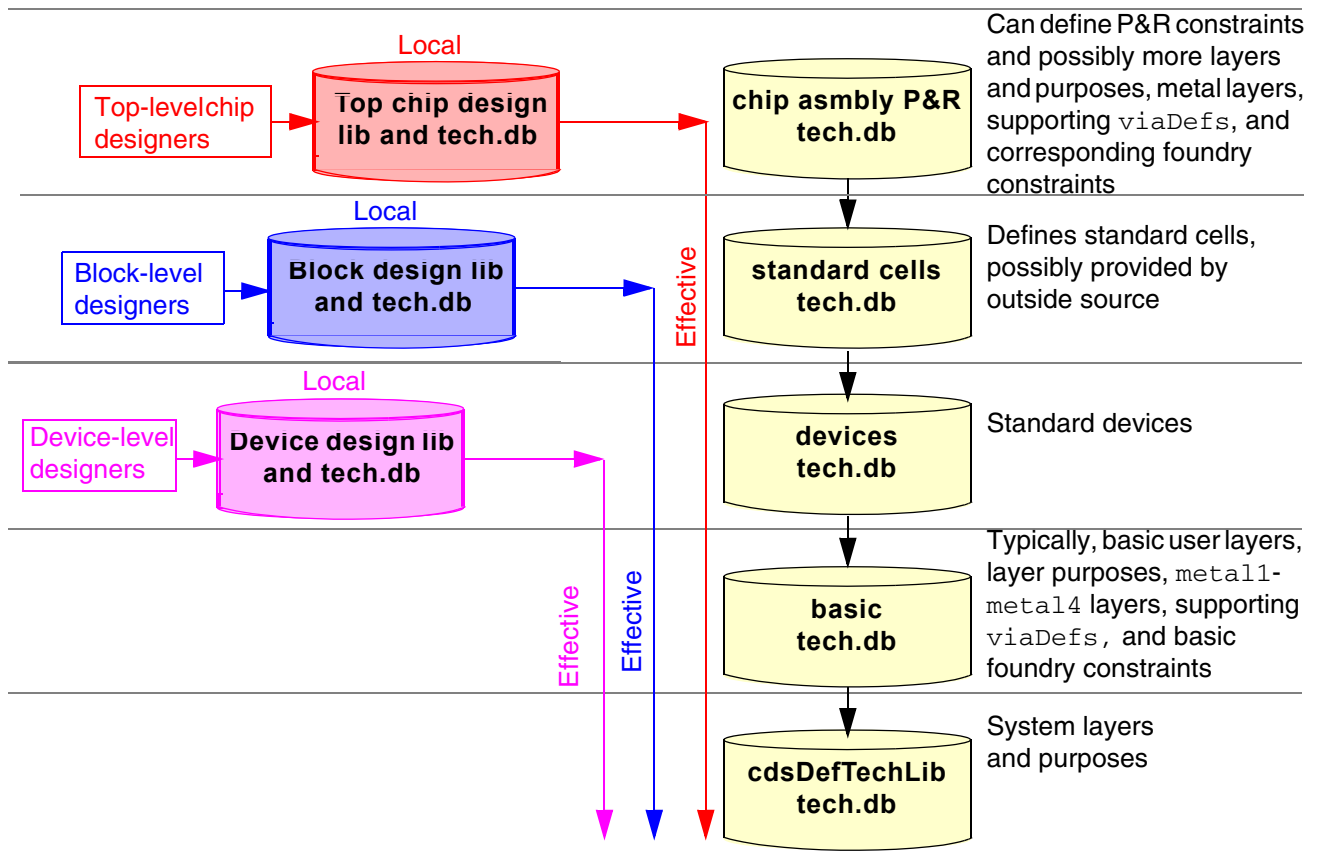
- reference and use only the technology data required for a specific task in the design flow, and
- define their own data in a writable technology database in which they set up references to any predefined read-only technology databases they need

A design library or an application can reference any technology database whose entry point in a graph provides only the technology data it needs. This technology database is the *local* technology database for that design library. The complete graph of databases initially referenced by the local database is the *effective* technology database. For example, in the following simple graph, in which the technology databases contain data grouped by type,

- The design library used by device-level designers can reference the `Devices` database, making `Devices` its local database. This entry point provides the reference chain (graph) containing all of the technology data required for device-level design and none of the data required by tasks later in the design flow (for example, the standard cells database, with data needed for block-level design but not for device-level design).
- The design library or application used for block-level design can reference the `standard cells` database, which is then its local database. This entry point provides the graph containing all of the technology data required for block-design and none of the data required by tasks later in the design flow.
- The design library or application used for chip assembly place and route can reference the `Router` technology database, its local database, which contains specific technology data required for place and route and, again, provides the complete graph of data required for this task.

# Virtuoso Technology Data User Guide

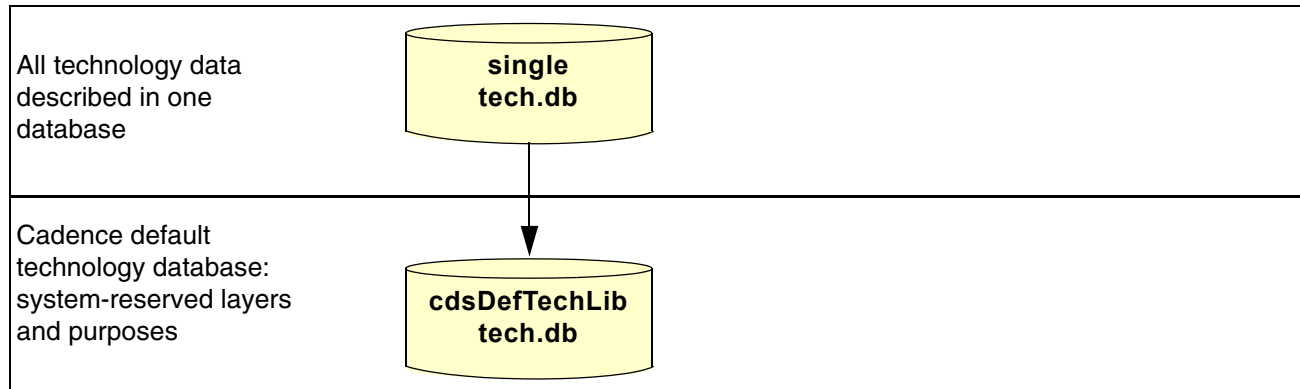
## About Virtuoso Technology Data



## Simple Incremental Technology Database Structures

Incremental technology database graphs can be as simple or as elaborate as required.

The simplest but least flexible is to specify all technology data in a single technology database.



Creation of any technology database automatically creates a reference to the Cadence default technology database, `cdsDefTechLib`. This default technology library defines all system-reserved layers and purposes. If system-reserved layers and purposes are in the compiled technology file, they are removed; defining them in two databases constitutes a conflict.

**Note:** `cdsDefTechLib` must be installed in the standard Design Framework II hierarchy. Failure to install `cdsDefTechLib` in this manner will result in missing layer and purposes in the technology databases.

Additionally, `cdsDefTechLib` specifies layer-purpose pair priorities and `techDisplays` settings, as well as assigning the substrate layer function. This data can be redefined in the referencing technology database, which overrides the definitions in `cdsDefTechLib`. For example, you can

- define other layer-purpose pairs
- redefine layer-purpose pair priorities
- specify different `techDisplays` for layer-purpose pairs to change how they are displayed

A technology database automatically inherits the layer-purpose pair priorities defined in `cdsDefTechLib` unless they are redefined higher in the graph. When inherited from `cdsDefTechLib`, the priorities are appended to the layer-purpose pair priority list specified in the technology database for user-defined layer-purpose pairs. When priorities are

## **Virtuoso Technology Data User Guide**

### About Virtuoso Technology Data

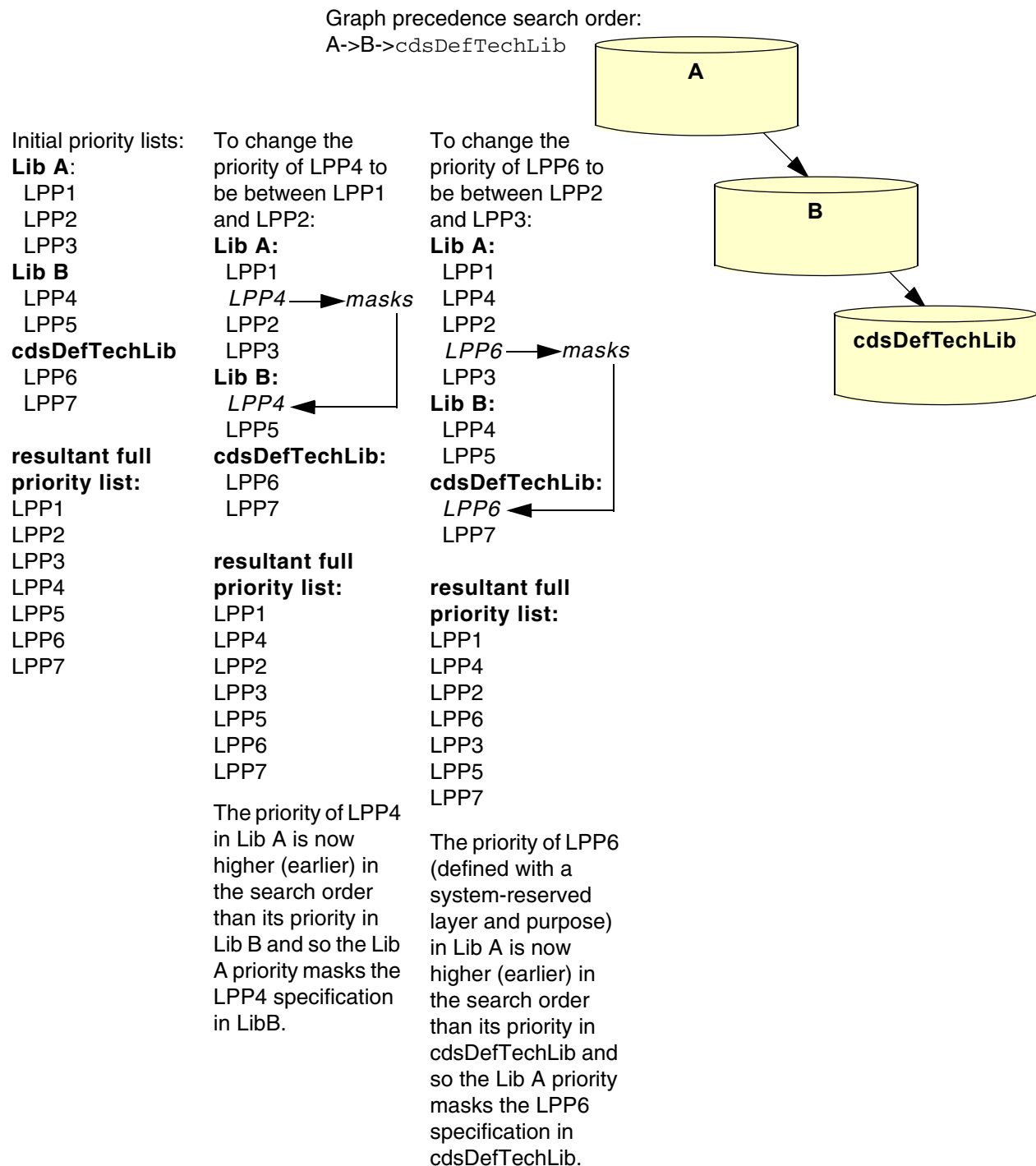
---

redefined, they are inherited based on the graph precedence search order. Any redefined priorities in the technology database override those specified lower in the graph. For example, then, you can mix user-defined layer-purpose pairs and layer-purpose pairs comprised of system-reserved layers and purposes throughout the priority list.

# Virtuoso Technology Data User Guide

## About Virtuoso Technology Data

The following simple illustration shows how layer-purpose pair priorities work in a graph:



The ability to redefine layer-purpose priorities in this way allows you to include lower-level layer-purpose pairs higher in the priority list.

## Virtuoso Technology Data User Guide

### About Virtuoso Technology Data

---



#### *Tip*

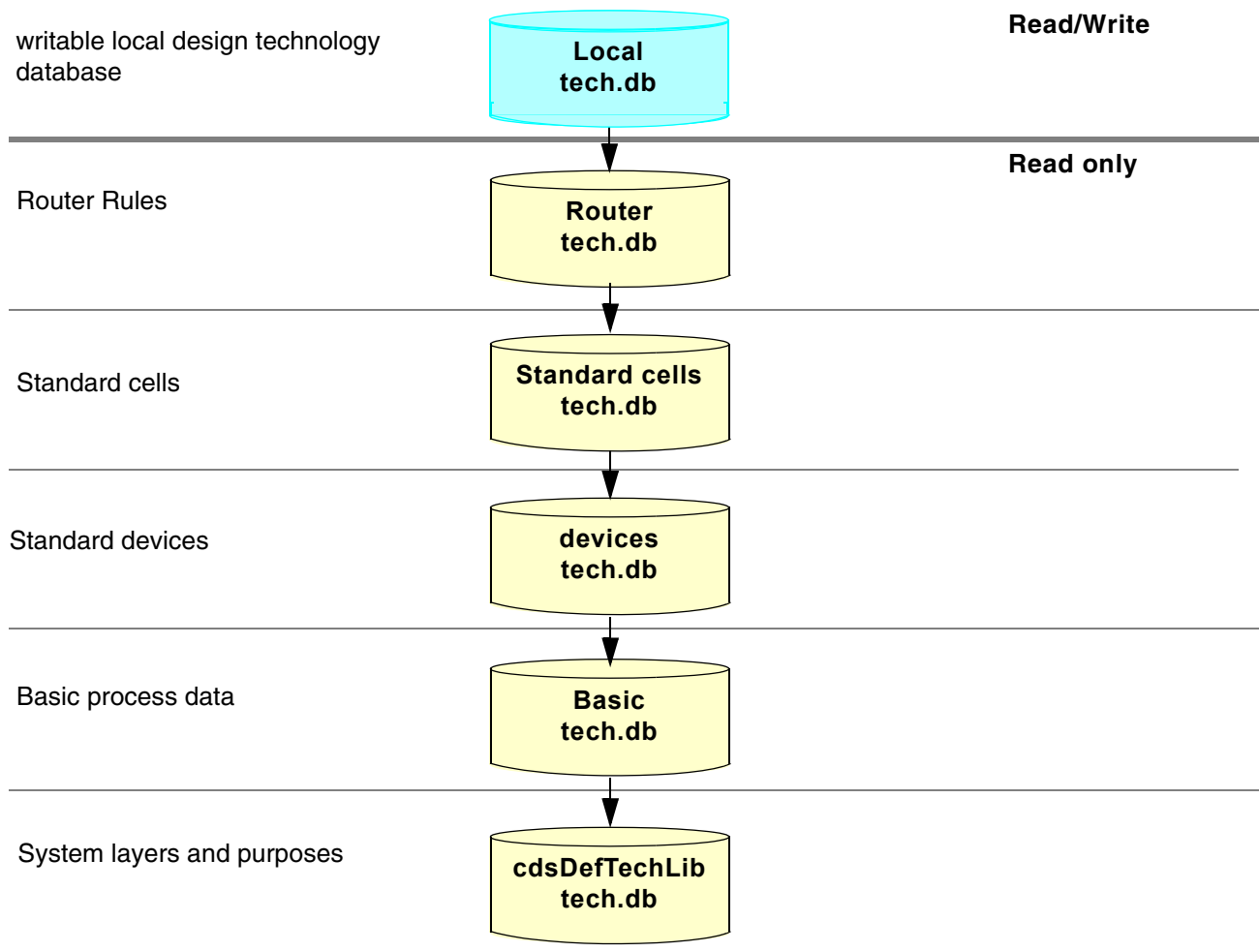
It is a good idea to define a full complement of layer-purpose pairs and their priorities in each technology database.

You can redefine the way layer-purpose pairs are displayed in the same manner, by redefining their `techDisplays` entries in any technology database. By default, a technology database inherits `techDisplays` definitions based on the graph precedence search order. Any redefinition overrides the definition in a lower technology database.

## Virtuoso Technology Data User Guide

### About Virtuoso Technology Data

When designers are provided with established, read-only technology databases, they can create their own local technology database in which to define their own technology data and establish a reference to the read-only technology databases. For example, in this graph, the designer's local technology database can reference the `Router` technology database, as shown below, or any other entry point that provides the data that is required for the designer's task. Multiple local technology databases can be created and added to the graph as needed to be used by designers at different entry points in the graph.



## User-defined Reserved Layers

Earlier, on the DFII, there are certain reserved layers defined in `cdsDefTechLib`. Users don't need to specify these layers, since these layers automatically gets entered in the memory once the database gets read in. In addition, the technology file database saved by the user does not have the information for those system reserved layers.



## Virtuoso Technology Data User Guide

### About Virtuoso Technology Data

---

For example, `<text 230>` is a system reserved layer. This layer will automatically get entered in the technology file database once the database gets read in. However, when the user exits the program, there is no such layer information in the saved database. Therefore, the user is unable to find the `text` layer.

Now, a user can also define reserved layers in the database. If the user specifically defines the layers with correct layer name and layer number, DFII will consider them as user-defined reserved layers. For these layers, the technology file database always saves the layer information, even after the user quits the DFII environment.

Therefore, if the user defines such layer with the name, `text` and number, `230` either through the ASCII technology file or through SKILL API `techCreateLayer`, then this layer will be treated as a user-defined reserved layer. In addition, this layer's information can be found in the database even after the user quits the DFII environment.

## Providing Like Data for Different Design Purposes

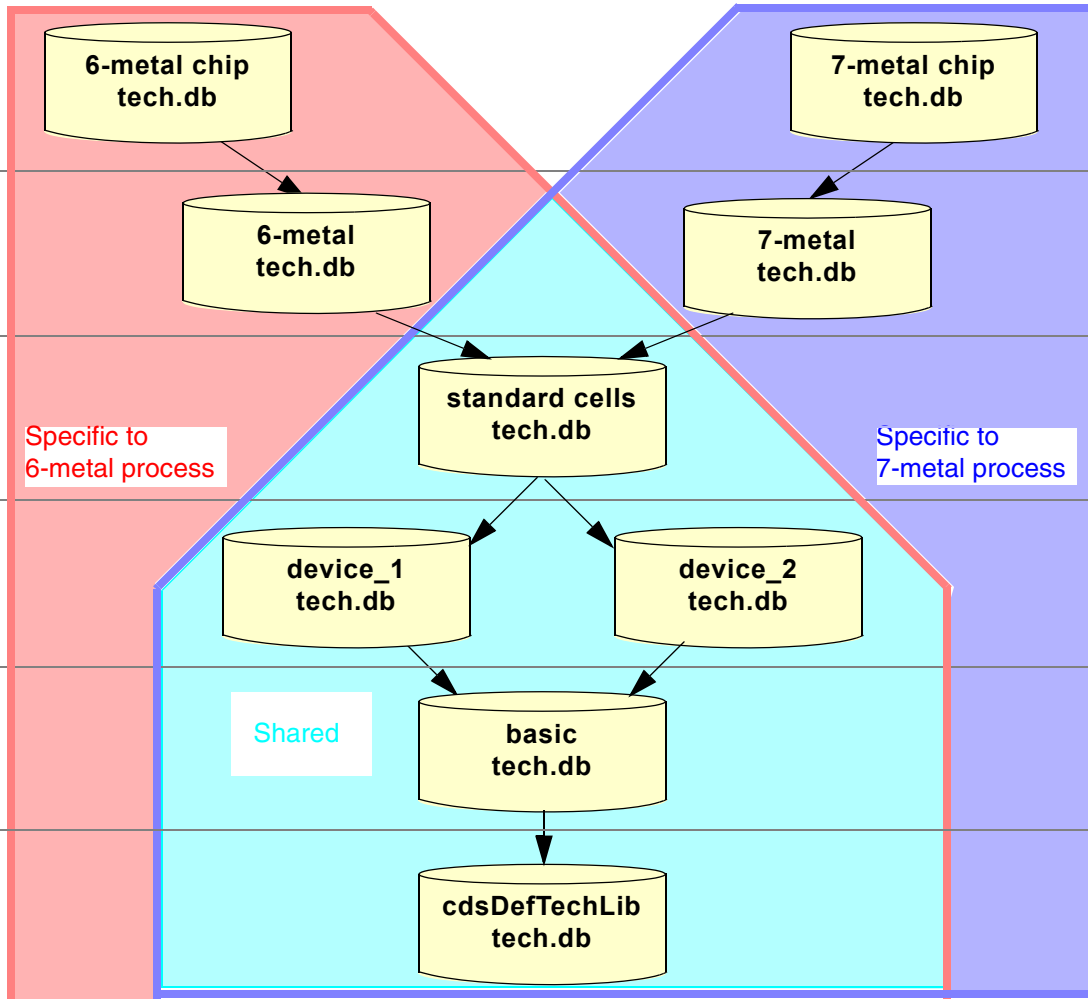
The following sample graph is designed to provide the data required for a 6-metal process and for a 7-metal process:

# Virtuoso Technology Data User Guide

## About Virtuoso Technology Data

Graph followed for 6-metal process

Graph followed for 7-metal process



The following summarizes the technology database contents and relationships in the graph:

**6-metal chip tech.db** is populated with data required for designing a chip with six metal layers: LEFDefaultRouteSpec constraint group, non-default constraint groups, and standard viaDefs for generated vias. It references **6-metal tech.db**.

**6-metal tech.db** defines the metal5-metal6 layers, their supporting `viaDefs`, and the foundry constraints they require.

**7-metal chip tech.db** is populated with data required for designing a chip with seven metal layers: LEFDefaultRouteSpec constraint group, non-default constraint groups, and standard viaDefs for generated vias. It references **7-metal tech.db**.

**7-metal tech.db** defines the metal5-metal7 layers, their supporting `viaDefs`, and the foundry constraints they require.

**6-metal tech.db** and **7-metal tech.db** then both reference the graph that contains the data common to both of them by referencing **standard cells tech.db**.

**standard cells tech.db** defines standard cells, defines siteDefs for standard cells, and references two device libraries, **device\_1 tech.db** and **device\_2 tech.db**.

**device\_1 tech.db** and **device\_2 tech.db** define two sets of custom-design data with additional device definitions and specific constraint groups for custom layout applications such as VLM and VCAR. Both reference **basic tech.db**.

**basic tech.db** defines basic processing data, such as base layers, user purposes, and including defining metal layers 1-4, the `viaDefs` to support them, and the foundry rules for them. It references, by default, **cdsDefTechLib**.

## Multiple Design Libraries Using Incremental Technology Databases

The following example illustrates a complex incremental technology database graph. The graph provides multiple entry points, designed to supply each design library with the data it needs without burdening it with data that it does not use but that is required by other design libraries setting references higher in the graph. It also allows defining technology data required by more than one design library only once in a technology database that can be shared.

**The Top-Level Chip design library** extends itself by referencing the IP design library, `IOPadLib`. All require the 6-metal technology database. Also, the design library can instantiate devices from both of the other design libraries it references.

**The Digital Block 1 design library** extends itself by referencing the IP design library, Memory block. Digital Block 1 requires the `LEFDefaultRouteSpec` constraint group, non-default constraint groups, and standard via definitions for generated vias defined in the 6-mtl P&R rules technology database, which also requires the same 6-metal technology database that is in the effective graphs for the Top-Level Chip design libraries. Additionally, this design library uses site definitions defined in the technology database, standard cells A.

**The Digital Block 2&3 design library** also uses the 6-mtl P&R rules technology database used by the Digital Block 1 design library; however, it uses a different set of site definitions defined in the technology database, standard cells B. It, too, references the IP design library, Memory block.

Technology databases supporting a 7-metal process, although not used by the designs in this graph, are available for 7-metal process designs when needed. The 7-mtl P&R rules technology database contains the `LEFDefaultRouteSpec` constraint group, non-default constraint groups, and standard via definitions for generated vias needed for a 7-metal process. The 7-metal technology database provides the layers, via, and foundry constraints required by 7-metal processes in addition to the shared base data.

# Virtuoso Technology Data User Guide

## About Virtuoso Technology Data

### Design Libraries

In-House  
Design Libraries

Reference IP  
Design Libraries

### Incremental technology databases

P&R

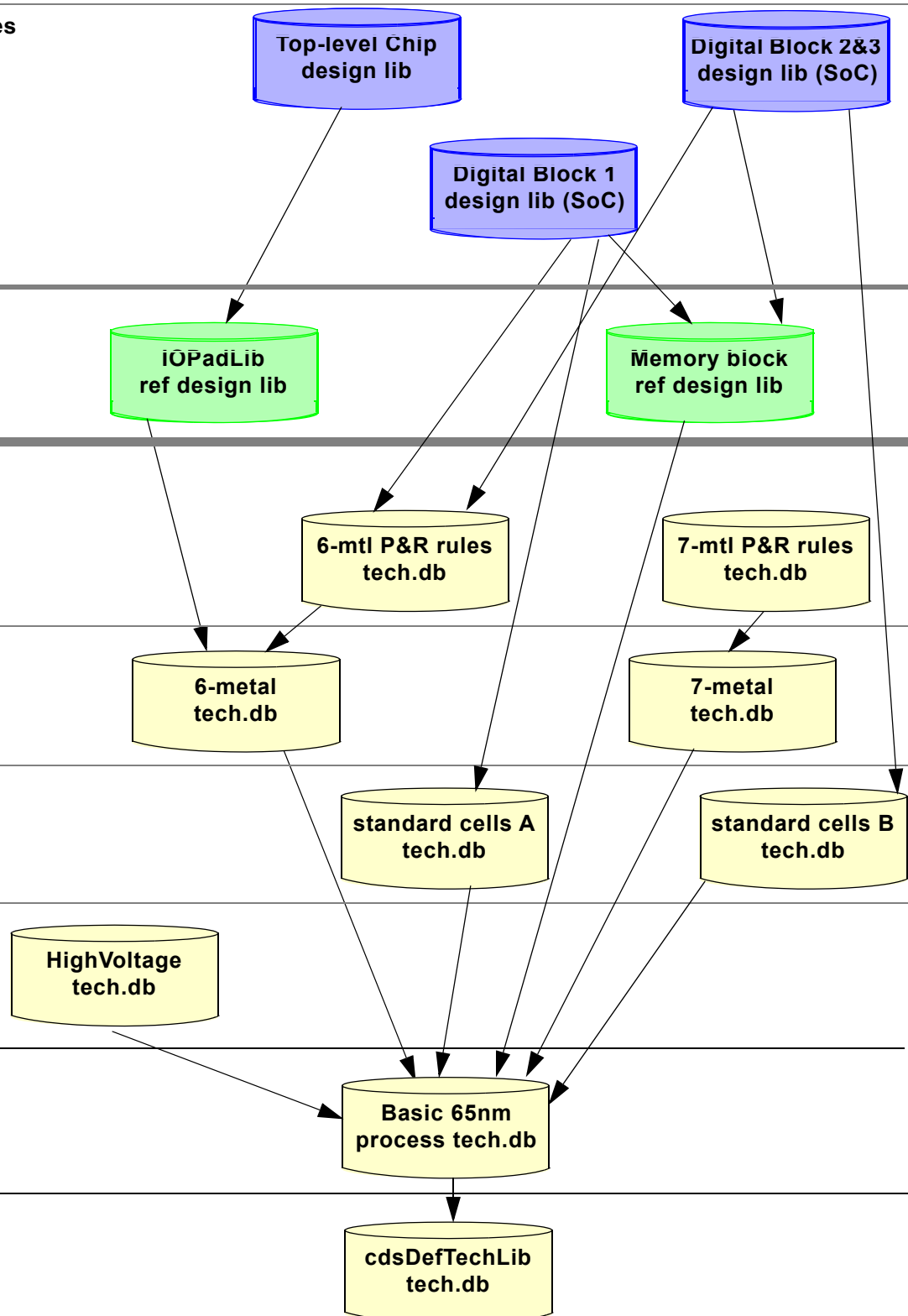
Metal options

Standard cells

Optional  
devices

Basic  
process  
data

System layers and  
purposes



## Designing Incremental Technology Database Graphs

Determining how to structure your technology databases depends primarily upon your design processes and what technology data you must use and cannot change, such as constraints defined and used by your foundry. Technology data should be grouped optimally for your design flow and for the most efficient access and use by the applications you employ. As you define a graph, it is important to

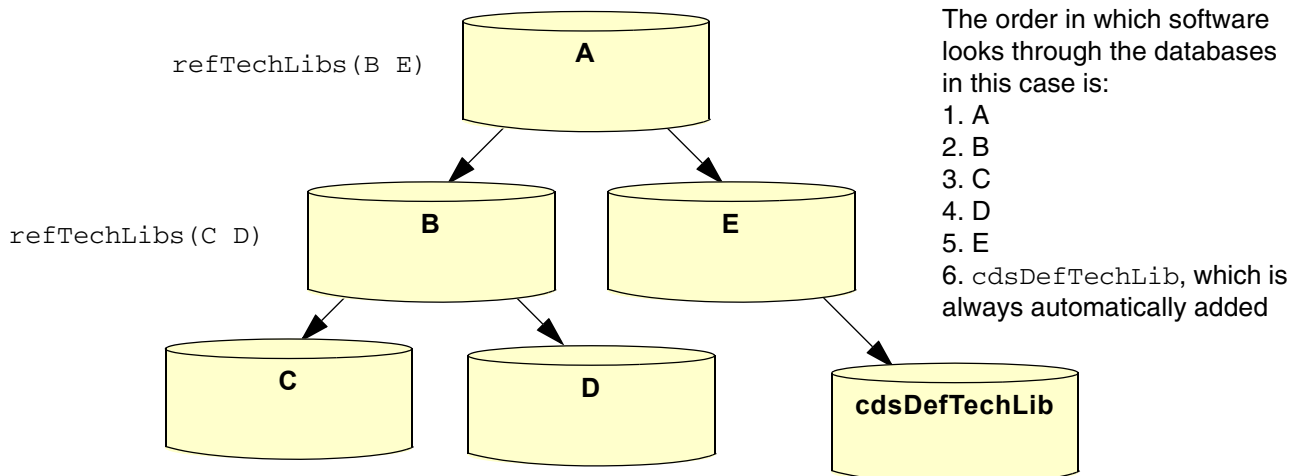
- Ensure that the data required by any database is provided by a database it references.
- Ensure that the order in which software will look through the databases is correct for your purposes.
- Consider where each task in your design flow will enter the graph, and add technology data specific to that task at that entry point. This ensures that only relevant technology data is seen during any design task.
- Avoid defining conflicting data in multiple databases in a graph.

### Database References

Data organization must be planned and defined so that any referenced database contains the data expected and required by the referencing database.

### Database Reference Ordering

The order in which you reference technology databases from other technology databases determines the order in which software looks through the graph to find data. It will go to the first referenced database and then go through its referenced databases before returning to the second referenced database. The following illustrates this order:



No conflict exception warning is displayed when the referenced technology in the graph has a default value attribute, which is different with the top technologies in the graph. For example, if technology A references technology B and technology B's DBUPerUU is the default value then no conflict notification is issued.

## Database Content Targeting

Defining technology data in a graph only when it is required by a task keeps data from being seen in tasks that do not need it. For example, the base technology database typically contains definitions of the layers, constraints, and via definitions required for lower-level metal processes (metal1 through metal4). If your graph must provide data to support higher-level metal processes, such as a 6-metal process, then those ought to be defined in separate databases rather than in, for example, the basic technology database, which is used by every task in the design flow. There would be no reason to provide technology data for a 6-metal process to designers who use only lower-level processes (metal1-metal4).

## Conflict Avoidance

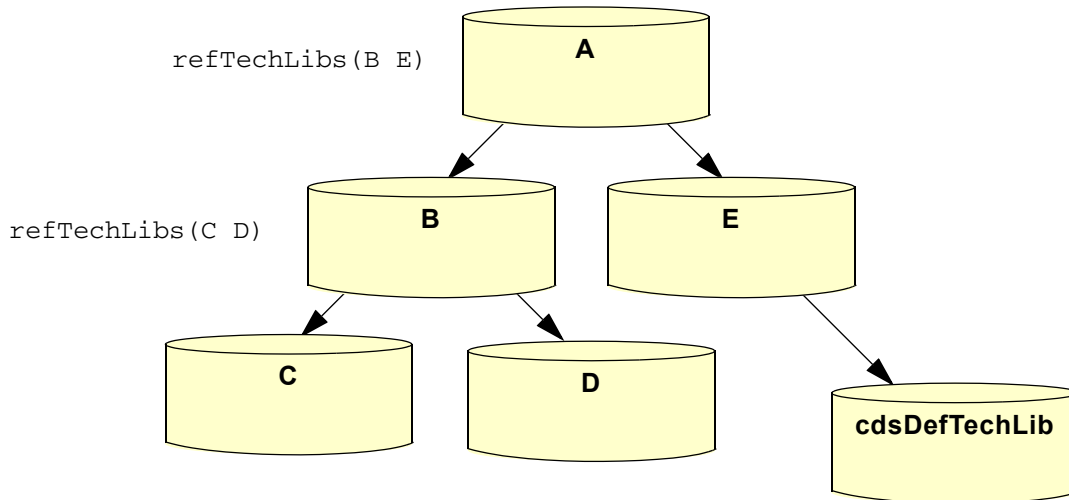
While some technology data can be repeated in more than one database, most cannot and duplicating it will create conflicts. The software checks for conflicts whenever and however you introduce a new technology database or make any additions or changes to an existing technology database in a graph. When it detects a conflict, the software issues a warning or an error and disallows the action creating the conflict. For example, when compiling an ASCII technology file, the compilation will be aborted upon detecting certain conflicts. When setting or creating data that creates a conflict with SKILL, an error will be issued and the requested set or create function will not be completed. See [Appendix C, “Data Handling for Incremental Technology Databases”](#) for complete details about data conflicts and how the software handles conflict resolution.

## Virtuoso Technology Data User Guide

### About Virtuoso Technology Data

---

The following simple illustration is used as a reference for this discussion.



It is possible to introduce *immediate* conflicts or *out-of-context* conflicts.

Immediate conflicts are those that are detected and can be handled as soon as they are introduced. For example, referring to the above illustration,

- During compilation, the software checks for conflicts throughout the compiling and compiled databases in the graph. For example, when compiling the technology databases in this graph, C is compiled first, followed by B. If the software detects a conflict between B and C when compiling B, the compiler immediately issues a warning and aborts the compilation.
- During a design session, changing, adding, or deleting data can create an immediate conflict that is detected as the software checks for conflicts throughout all of the open technology databases in all graphs. For example, with every database in the graph open, adding data to B that conflicts with data in any other database in the graph causes the application to report the conflict immediately.

Out-of-context conflicts result whenever data is changed in a database when the software does not see the entire graph and so cannot check all databases for conflicts. For example,

- If, during a design session, A is not open, and a change is made to B that does not cause a conflict with C or D (the open databases in the graph, which includes B, C, and D when B is the local database), there is no problem detected. However, if the changed data conflicts with data in A, the next time A is opened, the conflict will be detected and reported.

To avoid introducing out-of-context conflicts, it is important to keep in mind that any single technology database can be referenced in many different graphs simultaneously.

## **Conflict Correction**

Whenever the software reports a conflict that it cannot resolve and needs to be corrected by you, you must

1. Dump the technology database to an ASCII file.
2. Edit the ASCII file to eliminate the conflict.
3. Recompile the corrected ASCII technology file.



## Defining and Creating Technology Databases

### ASCII Technology and Display Resource Files

Two basic types of ASCII files contribute to defining a technology database:

- technology files, which define technology data
- display resource files, which specify definitions of how layers appear on display devices

**Note:** Some products require their application-specific rules in separate files. For specific application software requirements, see the documentation for that application.

A technology file can contain any or all of the following:

- statements and controls used in the technology file
- layer definitions
- layer attributes
- design constraints
- site definitions
- via definitions
- via specifications
- device definitions
- Palette Assistant window layer display specifications

A display resource file contains the following

- display device definitions
- definitions of colors, stipple patterns, line styles, and fill styles
- definitions of display packets, which are collections of colors, stipples, and line styles associated with particular display devices. A display packet specifies how you want a layer to be represented on the monitor or by a plotter. The technology file can assign a display packet to any layer it defines. In other words, a display resource file defines display packets and assigns a display packet to a display device or plotter, and the technology file assigns a display packet to any layer it defines.

For detailed information about these ASCII files and the data you can specify in them, see the [Virtuoso Technology Data ASCII Files Reference](#).

## Compiling ASCII Technology Files into Technology Databases and Graphs

Loading, or compiling, an ASCII technology file creates a technology database. To create an incremental technology database graph, you must compile ASCII technology files in the proper order. Essentially, this means that you must compile the technology files of referenced databases before compiling their referencing database; you cannot compile a technology file when any database it references is not already compiled. In other words, you must compile the technology databases from the bottom up as seen in the graph. If you try to compile a technology database out of order, the compiler will see that the referenced database does not exist, issue an error, and abort the compilation.

## Setting up Library Access to Technology Libraries

You can attach to or reference a technology library from any other design or technology library. To use your technology data in a design session, however, a design library must either attach to or reference a technology library. The choice between attaching and referencing determines designers' ability to create and edit their own local technology database.

Attaching a read-only technology library to another library provides access to all of the data defined in the local and effective technology database, but precludes designers from defining their own local technology data. Attachment can be preferable in cases where designers use only predefined technology data.

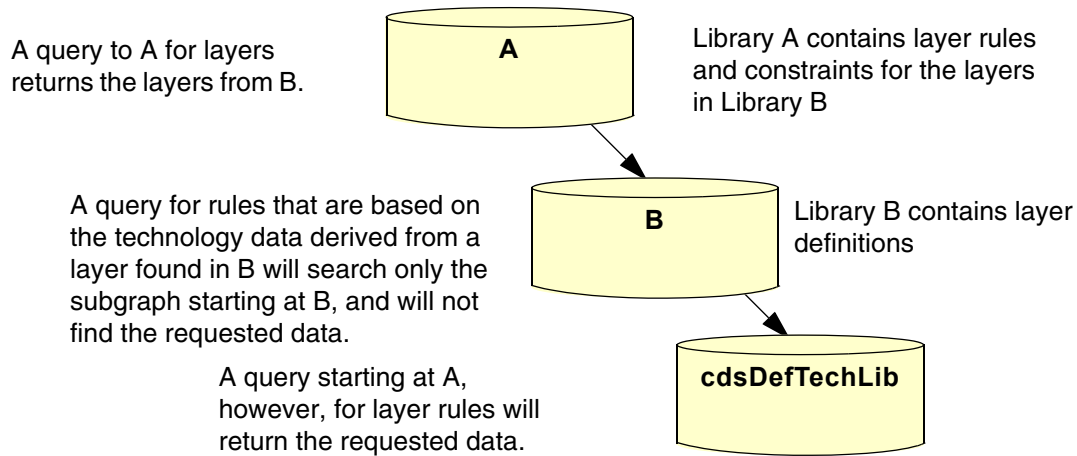
Referencing a read-only technology database from a writable design library or technology library protects the integrity of the data in the read-only technology database, but also provides a local technology database where designers can define their own technology data. Referencing is preferable to attachment when you want to be able to add technology data during the design process. (For example, creating a writable library provides a place to store LEF/DEF generated data.)

**Note:** Libraries attached to technology databases contain the `techLibName` property that is added to them when the attachment takes place; this property identifies the attached technology library. Referenced libraries do not contain this property, which is applied only during attachment.

## Querying Incremental Technology Bases with SKILL

When you use SKILL to retrieve data from an incremental technology database, always query the technology database from the top of the graph, and never query based on derived technology data from an object. If you do, you may get incorrect results, depending upon where the object is defined in the graph. The software searches the graph from the technology library containing the object down; if the data requested is above that technology library, it will not be found. The following is a simple illustration of how queries work:

In response to a query to a database, the software searches the graph from the specified database down.



For information on SKILL functions, refer to the [Virtuoso Technology Data SKILL Reference](#).

# **Virtuoso Technology Data User Guide**

## About Virtuoso Technology Data

---

---

## ASCII Technology File Development

---

This chapter introduces and presents an overview of ASCII technology file development and usage. It also summarizes technology file organization and presents definitions of the various kinds of data defined in a technology file.

This chapter discusses the following:

- [Methods of Initial ASCII File Creation](#) on page 37
- [Technology File Organization](#) on page 38
- [Which Technology Data Does Your Application Use?](#) on page 38
- [Specifying Data for An Incremental Technology Database](#) on page 39
- [Creating a Technology Database from a Technology File](#) on page 40

### Methods of Initial ASCII File Creation

You can create a new ASCII technology file by any of the following methods:

- In a text editor, create a technology file from scratch.
- Copy a sample ASCII technology file from the Cadence<sup>®</sup> installation and edit it in a text editor to produce your own technology file.
- Copy an existing ASCII technology file from your company's files and edit it in a text editor to produce your own ASCII technology file.
- [Dump](#) a technology file from an existing technology library and edit it in a text editor to produce your own ASCII technology file.

Whatever method you use, the structure of and requirements for specifying ASCII technology file data remain the same. For details, see the [Virtuoso Technology Data ASCII Files Reference](#).

## Technology File Organization

An ASCII technology file is organized into sections that define technology data as described in the following paragraphs. You can specify one, some, or all of these sections in an individual technology file, depending upon what is needed to create the technology database you want for your graph.

**Technology file statements** (`include` and `comment`) allow you to include other files in the technology file and put comments in the technology file that remain during compilation and dumping.

**Controls** (`controls`) specify data that can be used throughout the technology file and also specify references to other technology libraries.

**Layer definitions** (`layerDefinitions`) define the layers that can be used to define data throughout the technology file or by other technology databases in design sessions.

**Layer rules** (`layerRules`) specify layer attributes, such as layer materials, manufacturing grid resolutions, and current densities.

**Constraint groups** (`constraintGroups`) specify design constraints.

**Site definitions** (`siteDefs`) define scalar site definitions and arrays of scalar site definitions.

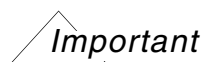
**Via definitions** (`viaDefs`) define standard and custom vias.

**Via specifications** (`viaSpecs`) define arrays of via definitions.

**Devices** (`devices`) define Cadence-predefined and ruleContact devices and multipart path templates.

For complete definitions of technology file sections and details about specifying technology file data, see the [\*Virtuoso Technology Data ASCII Files Reference\*](#).

## Which Technology Data Does Your Application Use?



To provide the greatest flexibility and satisfy the needs of a wide range of applications, the technology file provides a place to define many different data items. Not all of the data you can define is recognized and used by all applications. Before

specifying data in the technology file or adding it to a technology database, make sure that it is used by the application or applications you are employing in your design process.

## Specifying Data for An Incremental Technology Database

When you specify technology data in your ASCII technology file, you must keep in mind how that data is to be used as an incremental technology database. You must specify data that is in accord with its referenced and referencing technology databases and will not conflict with other technology data in a graph.

For detailed information about where you can specify each type of data without creating conflicts and for a detailed chart explaining how the software handles data, including data in conflict, see [Appendix C, “Data Handling for Incremental Technology Databases.”](#) While the following paragraphs present some general guidelines, the reference information in the appendix specifies details for every type of data in every situation.

Duplication of most technology data in multiple databases in a graph is not allowed because it creates irresolvable conflicts. Some technology data, however, can be duplicated in more than one technology database.

1. You can duplicate any `MPPTemplate` or `non-viaDef` device of the same name in multiple technology databases.
2. You can duplicate the following technology file subsections and data in multiple technology databases:
  - ☐ `foundry` constraint groups and their contents, with the requirement that the objects referenced exist somewhere in the subgraph
  - ☐ default constraint groups and their contents, with the requirement that the objects referenced exist somewhere in the subgraph
  - ☐ layer-purpose pairs and their drawing priority order
  - ☐ `techLayerPurposePriorities` for the same layer and purpose names
  - ☐ `equivalentLayers`
  - ☐ `stampLabels`
  - ☐ `labels`
  - ☐ `leLswLayers`

3. You can specify the `dbuPerUU` value and user units in every technology database, but the value must be the same in all of them.

You cannot duplicate most other technology data, such as `viaDefs` or any constraint group with the same name other than a `foundry` or `default` constraint group.

Whereas, in general, objects referenced in a technology data definition can be defined in a referenced database, in certain cases, related data must be specified in the same technology database:

- Any `cdsViaDevice` and its `viaDef` must be in the same technology database.
- For any `techDisplays` definition, the specified layer-purpose pair must be defined or redefined in the same technology database.

## Creating a Technology Database from a Technology File

Compiling an ASCII technology file creates a technology database, or library, as defined by the ASCII technology file. The technology library consists of the binary technology database (always named `tech.db`) and the device cellviews defined in the ASCII technology file.

### Notes:

1. Do not rename the binary technology file or rearrange the file structure.
2. If your application has rules files separate from the technology file, you must file those in the technology library subdirectory. For information about application-specific rules files, see the documentation for your application.
3. Do not hard-code the binary technology database name into applications. Instead, retrieve the name with the SKILL function `techGetDefaultTechName` or, for C and C++, use the `# define techDefaultTechFileName` constant.

You must compile technology files for incremental technology bases in the order in which they are referenced; if you try to compile a technology file when its referenced database or any referenced objects, such as layers or purposes, does not already exist, the compilation aborts and issues an error message.

For details on compilation, see [Generating a New Technology Library](#) on page 60.



---

## Display Resource File Development

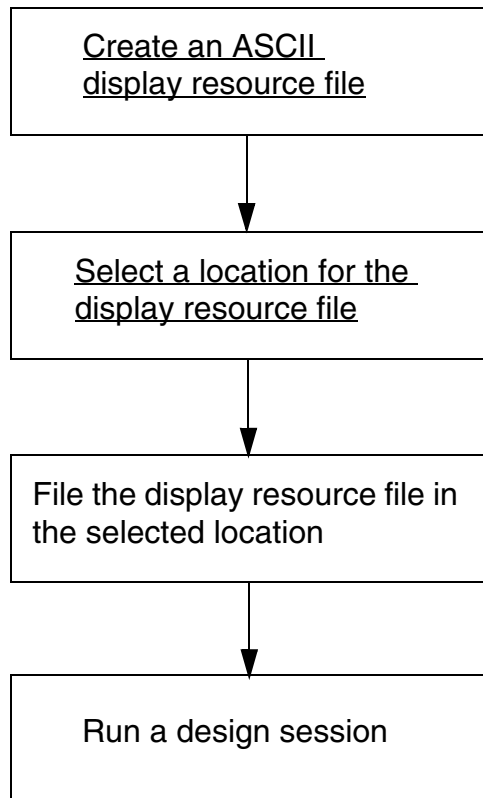
---

This chapter discusses the following:

- [“Overview of Development and Usage”](#) on page 42
- [“How Cadence Design Software Handles Multiple Display Resource Files”](#) on page 42
- [“Methods of Initial Display Resource File Creation”](#) on page 45
- [“Display Resource File Organization”](#) on page 45
- [“How Display Packets Control Layer Display”](#) on page 48

## Overview of Development and Usage

The following illustrates the major steps for display resource file development and usage:



Creating an ASCII display resource file with a text editor, you define the display data you want to use with specific display devices. The display resource file groups display data in display packets that it assigns to display devices. (The technology file can assign these display packets to layers to control layer display.) You can have multiple display resource files filed in various locations. Each, however, must be named `display.drf`.

When selecting where to file your `display.drf` file, you must consider that Cadence design software loads and merges up to six display resource files from predefined locations. See [“How Cadence Design Software Handles Multiple Display Resource Files”](#) on page 42 for details.

Upon opening a design library, Cadence design software automatically loads up to six display resource files into virtual memory. During a design session, you can [manipulate and edit the display resource data in virtual memory](#) with the Display Resource Editor or DFII SKILL functions.

## How Cadence Design Software Handles Multiple Display Resource Files

At startup, Cadence design software creates a display resource file in virtual memory for use during a design session. The virtual memory display resource data is based on five `display.drf` files present at standard locations and the files loaded by explicit usage of the `drLoadDrf` API.

Because the files are merged in sequence during reinitialization, files loaded later in the sequence can redefine display packets, colors, line styles, stipples, and display devices defined by files loaded earlier.

For information about reinitialization, see [“Reloading Source Display Resource Files”](#) on page 134.

The *File – Reinitialize* command loads only the files located in the standard paths mentioned in the following list:

- The Cadence-supplied default display resource file

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

This file is used with the Virtuoso Schematic Composer.

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

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

This is an optional file you can use to provide required display resource definitions.

- Optional site and project display resource files

These are optional files your system administrator can place in the site and project directories, if those directories are set up at your site. These files must be called `display.drf`.

For more information about these directories, refer to the [\*Cadence Application Infrastructure User Guide\*](#).

- Personal display resource file

`~/display.drf`

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

- The current directory

`./display.drf`

This is an optional file that you can customize and place in the directory from which you start the software. This file must be called `display.drf`.

If multiple technology libraries are used in your Cadence design environment, you can create a `display.drf` file with display requirements specific to each technology. Additional display resource files can be loaded during a session through explicit `drLoadDrf` calls from `.cdsinit`. However, such definitions are lost after the session and are not loaded during reinitialization. If you want such a file to be loaded during reinitialization, merge the technology-specific `display.drf` file with an existing `display.drf` at a standard location.

## Planning Display Resource File Updates for Proper Merging

Because the system merges several files to create the display resource data you use to create your designs, you will need to plan updates to the data. There will be times when you

## **Virtuoso Technology Data User Guide**

### **Display Resource File Development**

---

will use the Display Resource Editor and save your changes to a new `display.drf` file. There will be other times when you will need to edit a source display resource file in a text editor. For further information, refer to Chapter 6, “Editing, Reusing, and Merging Display Resources.”

## Methods of Initial Display Resource File Creation

You can create a new display resource file by any of the following methods:

- In a text editor, create a display resource file from scratch
- Copy a sample display resource file from the Cadence® installation and edit it in a text editor to produce your own display resource file
- Copy an existing display resource file from your company's files and edit it in a text editor to produce your own display resource file
- Dump the display resource data from virtual memory to a new display resource file
- Edit display resource data in virtual memory with the Display Resource Editor and save the edited data to a display resource file

Whatever method you use, the structure of and requirements for specifying display resources in a display resource file remain the same. This chapter defines how to specify display resources.

## Display Resource File Organization

A display resource file is organized into sections that define display resources as described in the following paragraphs.

**The display devices section** (`drDefineDisplay`) lists the names of the display devices for which display information is defined in the display resource file.

**The color definitions section** (`drDefineColor`) defines the colors used with various display devices. This section applies specific color definitions to color names and associates them with specific display devices.

**The stipple definitions section** (`drDefineStipple`) defines the stipple patterns used with various display devices. This section applies specific stipple pattern bitmaps to stipple names and associates them with specific display devices.

**The line style definitions section** (`drDefineLineStyle`) defines the line styles used with various display devices. This section applies specific line style sizes and patterns to line style names and associates them with specific display devices.

**The display packet definitions section** (`drDefinePacket`) defines the display packets used with various display devices. This section applies specific stipple patterns, line styles,

fill colors, outline colors, and fill styles to display packet names and associates them with specific display devices.

You can define two different packets for the fins and the tracks of a `snapPattern` shapes. For example, if the packet name of the `snapPattern` layer-purpose is `M1_spPurpose`, then you can add the following packets in the `drDefinePacket` section of `display.drf`:

`M1_spPurpose_snapPatternFull` to define the color of the `snapPattern` fins

`M1_spPurpose_snapPatternTracks` to define the color of the `snapPattern` tracks

If the color of `snapPattern` fins and tracks is not defined, then the color of these will be same as that of `M1_spPurpose`.

**The display packet alias definitions section** (`drDefinePacketAlias`) applies alias names to display packet names and associates them with specific display devices.

For details about specifying display resource file data, see the [\*Virtuoso Technology Data ASCII Files Reference\*](#).

## Virtuoso Technology Data User Guide

### Display Resource File Development

---

The following illustrates the sections of this file, along with the display resources they define.

```
drDefineDisplay(  
; ( displayName )  
  ( display      )  
  ( display2     )  
  .  
  .  
  .  
) ; end of drDefineDisplay
```

**Define Display section.** Lists the names of the display devices for which display information is defined in this file.

```
drDefineColor(  
; ( DisplayName  ColorName  Red   Green  Blue  Blink )  
  ( display      white      255   255   255      )  
  ( display      whiteB     255   255   255   t   )  
  ( display      silver     217   230   255      )  
  .  
  .  
  .  
) ; end of drDefineColor
```

**Define Color section.** Defines the colors used with various display devices.

```
drDefineStipple(  
; ( DisplayName  StippleName  Bitmap )  
  ( display      blank        (  
                                (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)  
                                (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)  
                                (0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)  
                                .  
                                .  
                                .  
                                )  
) ; end of drDefineStipple
```

**Define Stipple section.** Defines the stipple patterns used with various display devices.

## Virtuoso Technology Data User Guide

### Display Resource File Development

---

#### drDefineLineStyle(

```
; ( DisplayName   LineStyle   Size   Pattern )
( display        solid        1      (1 1 1) )
( display        dashed       1      (1 1 1 1 0 0) )
( display        dots         1      (1 0 0) )
.
.
.
) ; end of drDefineLineStyle
```

#### **Define Line Style**

**section.** Defines the line styles used with various display devices.

#### drDefinePacket(

```
; ( Display   Packet           Stip   Line   Fill   Outline [FillStyle])
( display    blacksolid_S     solid  solid  black  black   solid   )
( display    blue             blank  solid  blue   blue    )
( display    bluedashed_L     blank  dashed blue   blue    )
.
.
.
) ; end of drDefinePacket
```

#### **Define Display Packet**

**section.** Defines the display packets used with various display devices.

#### drDefinePacketAlias(

```
; ( displayName  packetAlias      packetName )
( display       blsol1           blacksolid_S )
( display       blue2            blue         )
( display       b3               blue         )
.
.
.
) ; end of drDefinePacketAlias
```

#### **Define Display Packet**

**Alias section.** Defines alias names for the specified display packet names.

## How Display Packets Control Layer Display

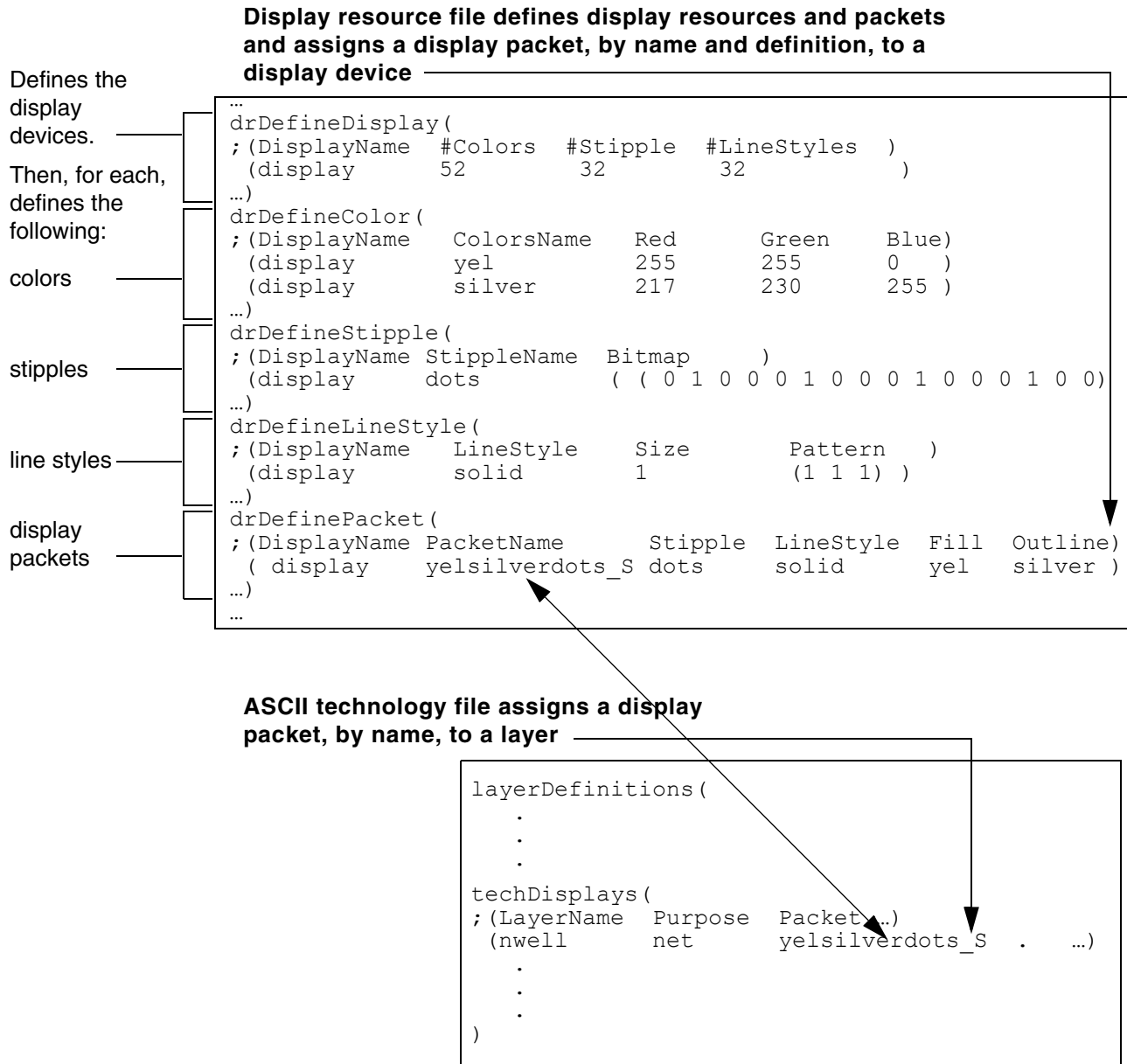
A technology database can contain assignments of display packets to layers to control how layers are displayed. This information can be specified in the `techDisplays` subsection in the `layerDefinitions` section of the ASCII technology file. You can also apply, change, and redefine display packets assigned to layers with the design software graphical user interface.



## Virtuoso Technology Data User Guide

### Display Resource File Development

The following sample illustrates the display resource file display packet definition and assignment to a display device and the technology file assignment to a layer:



# **Virtuoso Technology Data User Guide**

## **Display Resource File Development**

---

---

## Preparing Files for Use with a Design

---

This chapter discusses the following:

- The Technology File Manager and User Interface on page 52
  - Invoking the Technology File Manager on page 52
  - Technology File Manager Commands on page 53
- Displaying the Incremental Technology Database Graph on page 55
- Generating a New Technology Library on page 60
  - Compiling an ASCII Technology File into a Technology Database (Library) on page 60
  - Creating a New Technology Library from an Existing Technology Library on page 63
  - Creating a New Technology Library that References Existing Technology Libraries on page 64
- Referencing or Attaching a Technology Library on page 67
  - Referencing a Technology Library from a Design Library on page 68
  - Attaching a Technology Library to a Design Library on page 68
- Ensuring Desired Display Resource File Usage on page 69

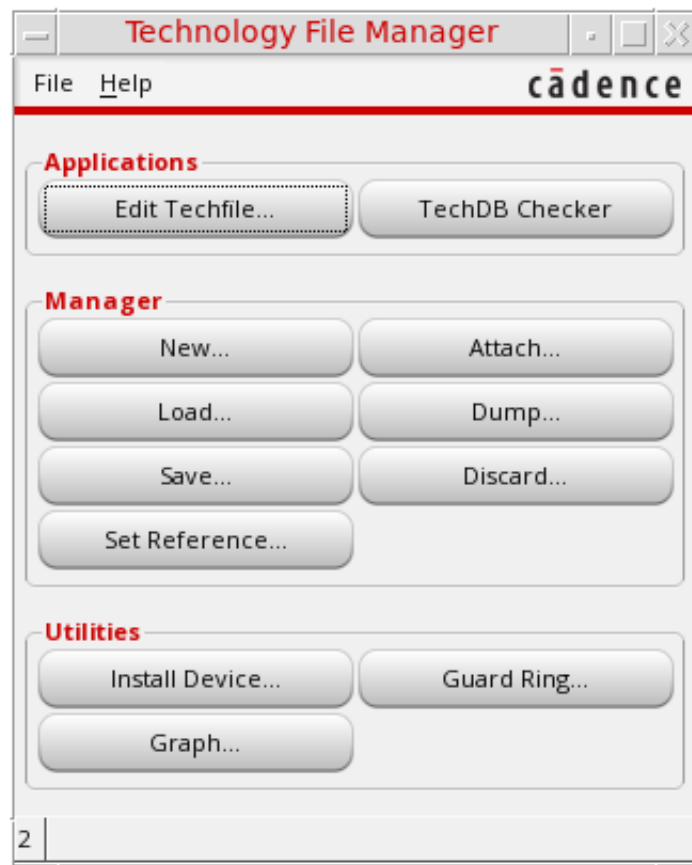
## The Technology File Manager and User Interface

The technology file manager provides a user interface that allows you to create technology databases and manipulate technology data in virtual memory during a design session.

### Invoking the Technology File Manager

To start the technology file manager from the CIW, choose *Tools – Technology File Manager*.

The technology file manager displays the Technology File Manager.



### SKILL Functions to Display Form

`techManagerOpenTechToolBox()`

## Technology File Manager Commands

The Technology File Manager commands let you compile, dump, and edit technology data. This section introduces the Technology File Manager commands.

*Edit Techfile* opens the Techfile IDE in which you can create, edit, and view technology files.

*TechDB Checker* inspects an OpenAccess technical database that was created by compiling a Virtuoso ASCII technology file using the Virtuoso Technology File Manager. It helps you check if the technology database has all necessary data to function with Cadence tools. For details, see the *Virtuoso Technology Database Checker User Guide*.

*New* creates a new technology library by compiling an ASCII technology file, copying an existing binary technology library, or referencing other technology libraries. It also loads the new technology library into virtual memory.

*Attach* assigns a technology library to another technology or design library.



Be sure you do not attach a library that you need to reference. For information on referencing as opposed to attaching, see Setting up Library Access to Technology Libraries on page 34 and Referencing or Attaching a Technology Library on page 67.

*Load* compiles an ASCII technology file into an existing technology library and loads it into virtual memory.

*Dump* writes a technology database from virtual memory to an ASCII technology file and opens the ASCII file in an editor window for you to view and edit.

*Save* writes a technology database in virtual memory to the technology database on disk. You can make whatever changes you want to the technology database in virtual memory without changing the permanent copy on disk by not saving. If you want the changes preserved in the technology database, you must save to disk.

*Discard* deletes the current technology data from virtual memory and reloads technology data to virtual memory from disk.

*Set Reference* lets you establish a library containing only references to other libraries that you select and order.

*Install Device* invokes the Install Device form.

## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

---

Graph displays the incremental technology database graph for the current local technology database.

Guard Ring is a special type of ROD multipart path that are used to encircle one or more objects.

## Displaying the Incremental Technology Database Graph

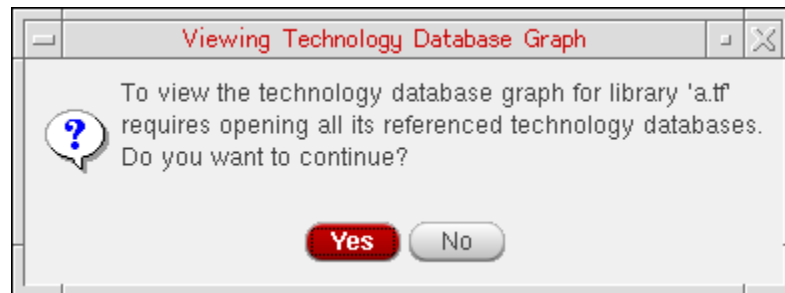
You can display the incremental technology database graph for the selected technology database at any time. To do so,

1. From the Technology File Toolbox, choose *Graph*.

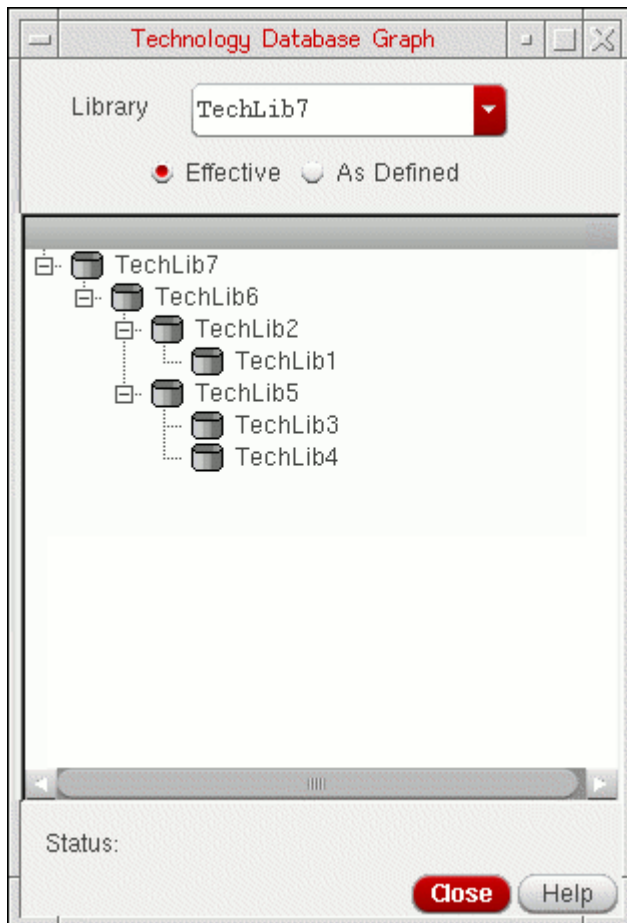
The Technology Database Graph form is displayed.

2. From the Library pull-down menu, choose the technology library whose graph you want to display.

**Note:** All technology databases in the graph must be open for the software to display the entire graph. If they are not open, the software displays the following message box:



Clicking **Yes** opens the referenced databases and displays the Technology Database Graph form.



For a description of this form, see [Appendix A](#).

## Effective and As Defined Graph Displays

The *Effective* radio button is selected by default and displays the effective technology database graph.

Any technology database that is referenced multiple times in a graph shows up only once in the effective graph. Its location is the first place in the graph where it will be encountered by software using the graph. (See [Database Reference Ordering](#) on page 29.)

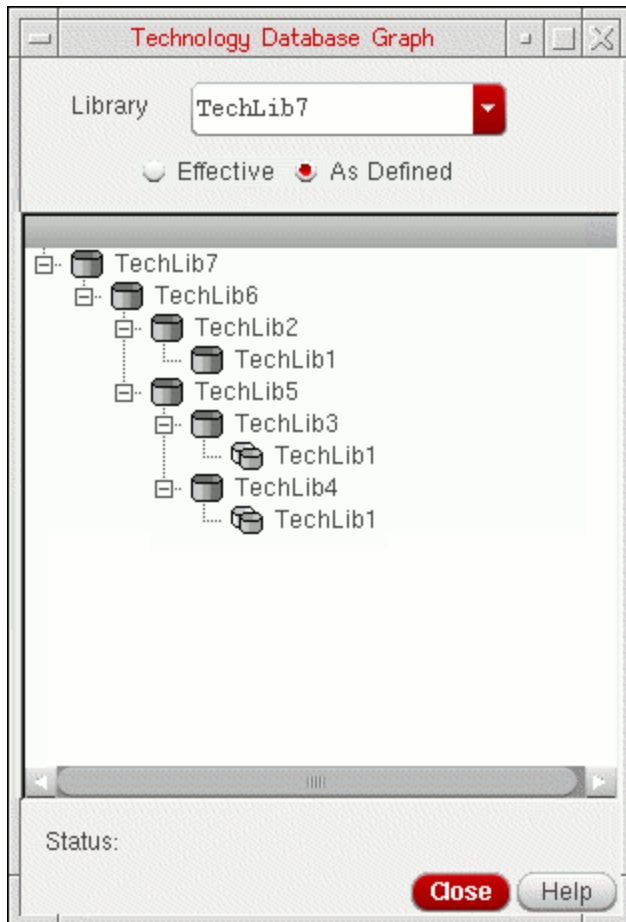


## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

---

Turning on the *As Defined* radio button displays the graph showing every reference originally defined in the technology databases.



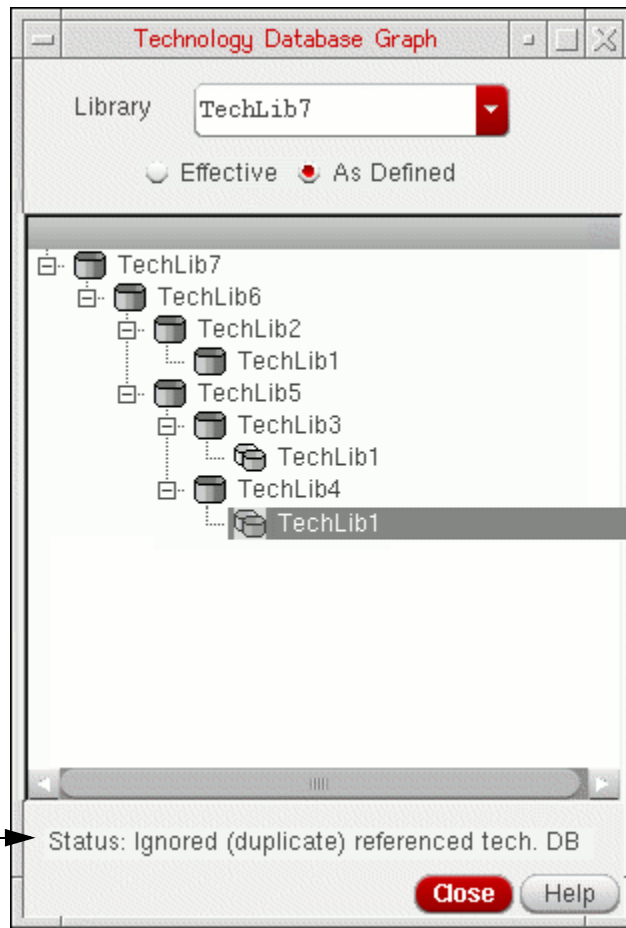
In this example, TechLib1 is referenced by TechLib2, TechLib3, and TechLib4. In the *Effective* display, TechLib1 is shown only once. Since TechLib6 references TechLib2 first, the TechLib2 reference to TechLib1 is displayed in the graph. In the *As Defined* display, its reference from each of the three other technology databases is shown, but represented by a different icon at the two extra references to indicate that it is repeated.

## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

When you click an icon, the *Status* line displays information about the selected database, as shown in the following example:

The status line displays information about the selected technology database. In the case of this *As Defined* display, the selected database, TechLib1, is referenced by multiple databases, and this is not the first reference in the graph.



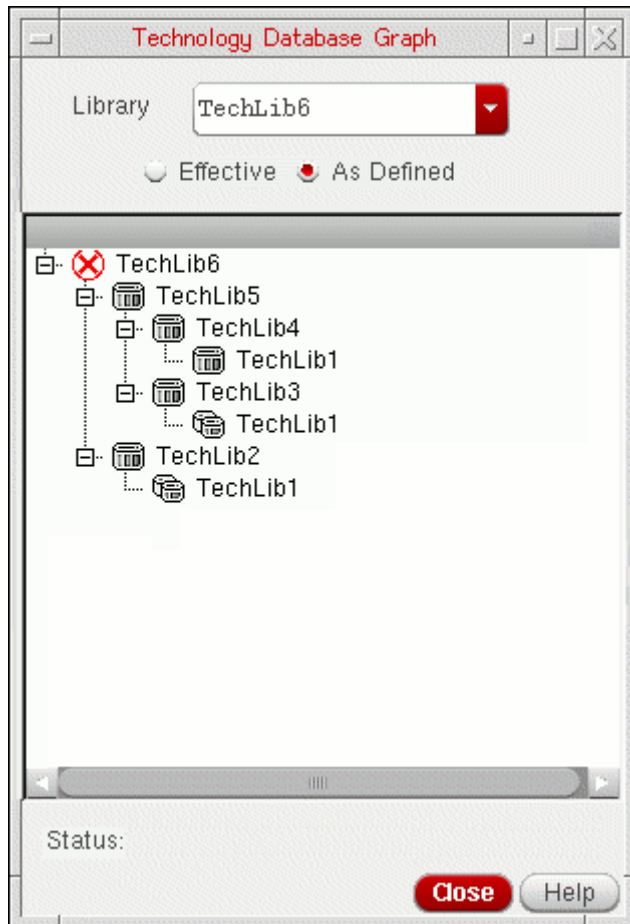
## Conflicts in Graph Displays

Conflicts detected at compilation time are never allowed to exist because the compilation process displays a message and aborts. It is, however, possible to introduce conflicts with closed databases higher in the graph when making changes to technology databases when only some of the databases are open. Once the affected databases are open, any conflicts then become apparent. In the Technology Database Graph, such conflicts are identified by a red X next to the highest database in the graph affected by the conflict (which is not necessarily the database containing the conflicting data). For example, in the following graph, TechLib6 is flagged as having a conflict. In this case, with only TechLib1 and TechLib2 open (that is, working with TechLib2 as the entry point in the graph with all other databases closed), merging a technology file containing technology data already defined in TechLib5, for example, is accomplished without any problem. However, when subsequently working with

## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

TechLib6, the conflict between TechLib2 and TechLib5 is identified. TechLib6 is flagged with the X because the conflict is applicable through the top of the graph due to the technology database references.



## Generating a New Technology Library

The Technology File Manager provides the following methods for generating a new technology database (library):

- compiling an ASCII technology file into a technology library
- copying an existing technology library to a technology library of a different name
- creating a technology library containing references to other technology libraries

### Compiling an ASCII Technology File into a Technology Database (Library)



Always compile technology databases in the order in which they are referenced by each other. In other words, when creating the technology databases in a graph, compile a referenced file before compiling its referencing file. Whenever you create a new technology library to add to a incremental technology database graph, first verify that any technology databases it references are already created. If you try to compile a technology file that references a technology database that does not exist, the software will issue an error message and abort the compilation.

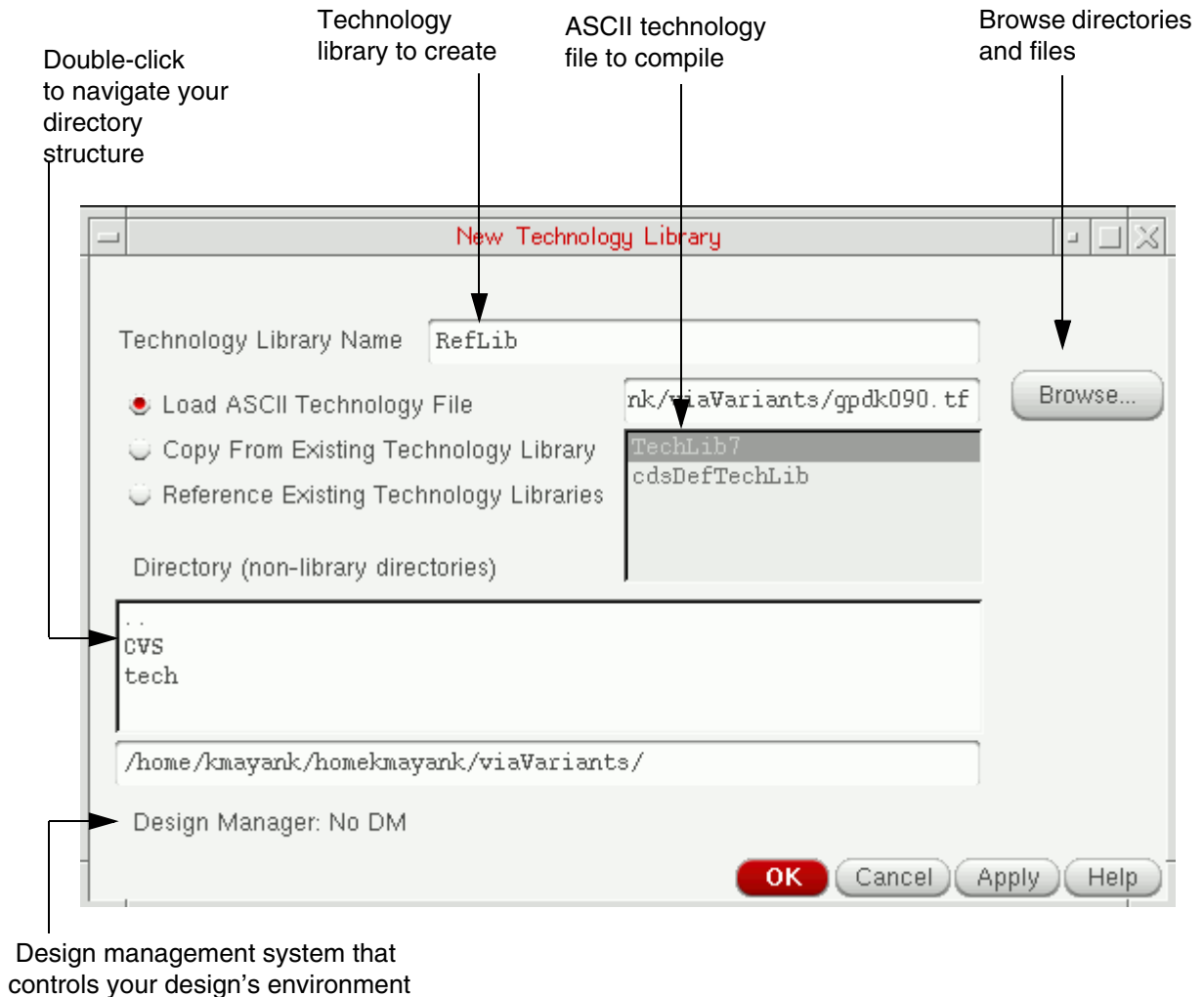
To create a technology library from an ASCII technology file, do the following:

1. From the Technology File Manager, choose *New*.

## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

The New Technology Library form appears.



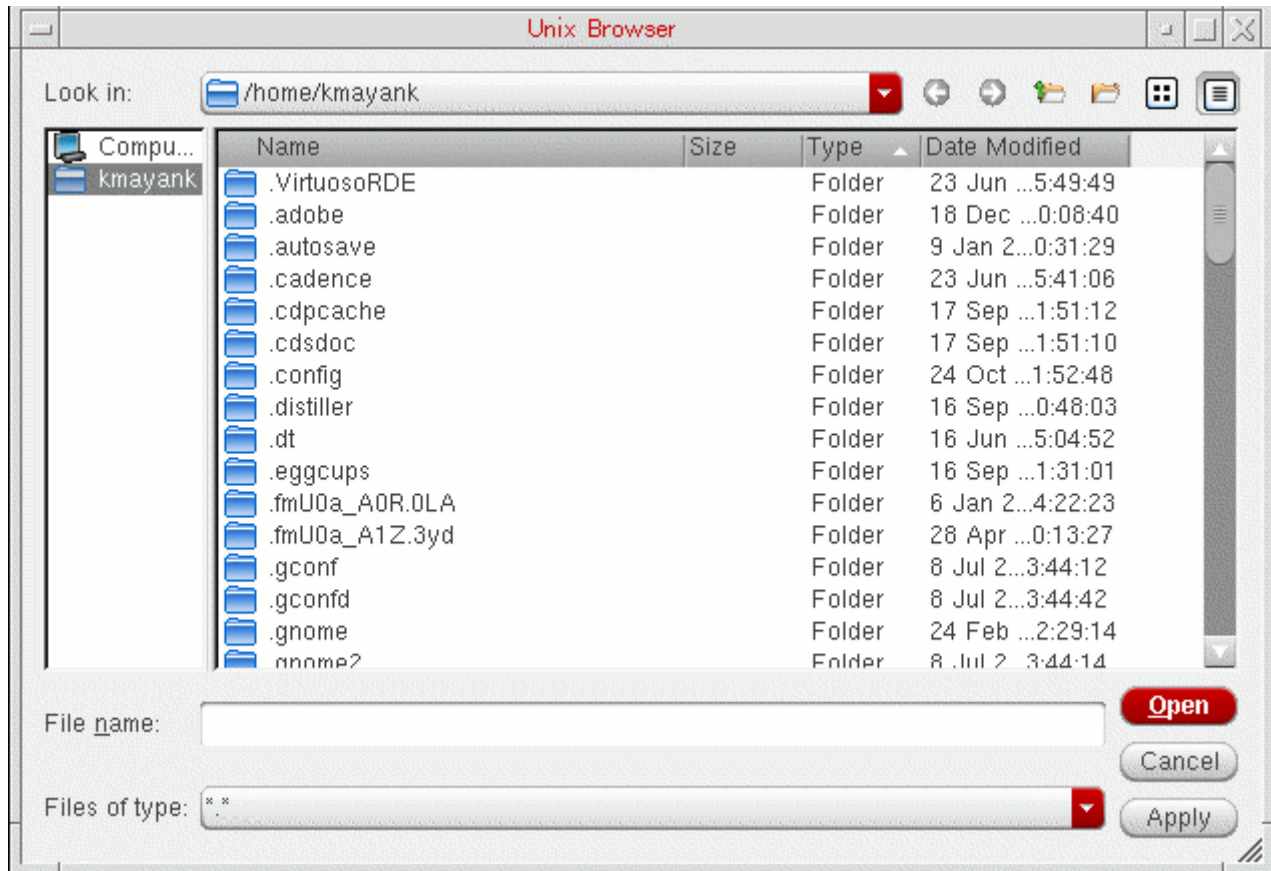
For a description of this form, see [Appendix A](#).

2. In the *Technology Library Name* field, type the name of the new technology library you want to create.
3. Click *Load ASCII Technology File*.

## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

4. Type the name of the ASCII technology file to compile or select it by using the browser. The UNIX browser dialog box is displayed.



5. Choose the directory in which to create the new technology library.
  - ☐ To descend into the directory hierarchy, double-click a directory.
  - ☐ To ascend the directory hierarchy, double-click the “.” directory.
  - ☐ To choose a directory, click the directory name.
6. If you want to use a design management system, choose the design manager from the *Design Manager* cyclic field.
7. Click *OK*.

The system compiles the ASCII file and creates the new technology library. The new library directory contains the binary technology database file called `tech.db` and cellviews of the devices defined in the ASCII technology file.

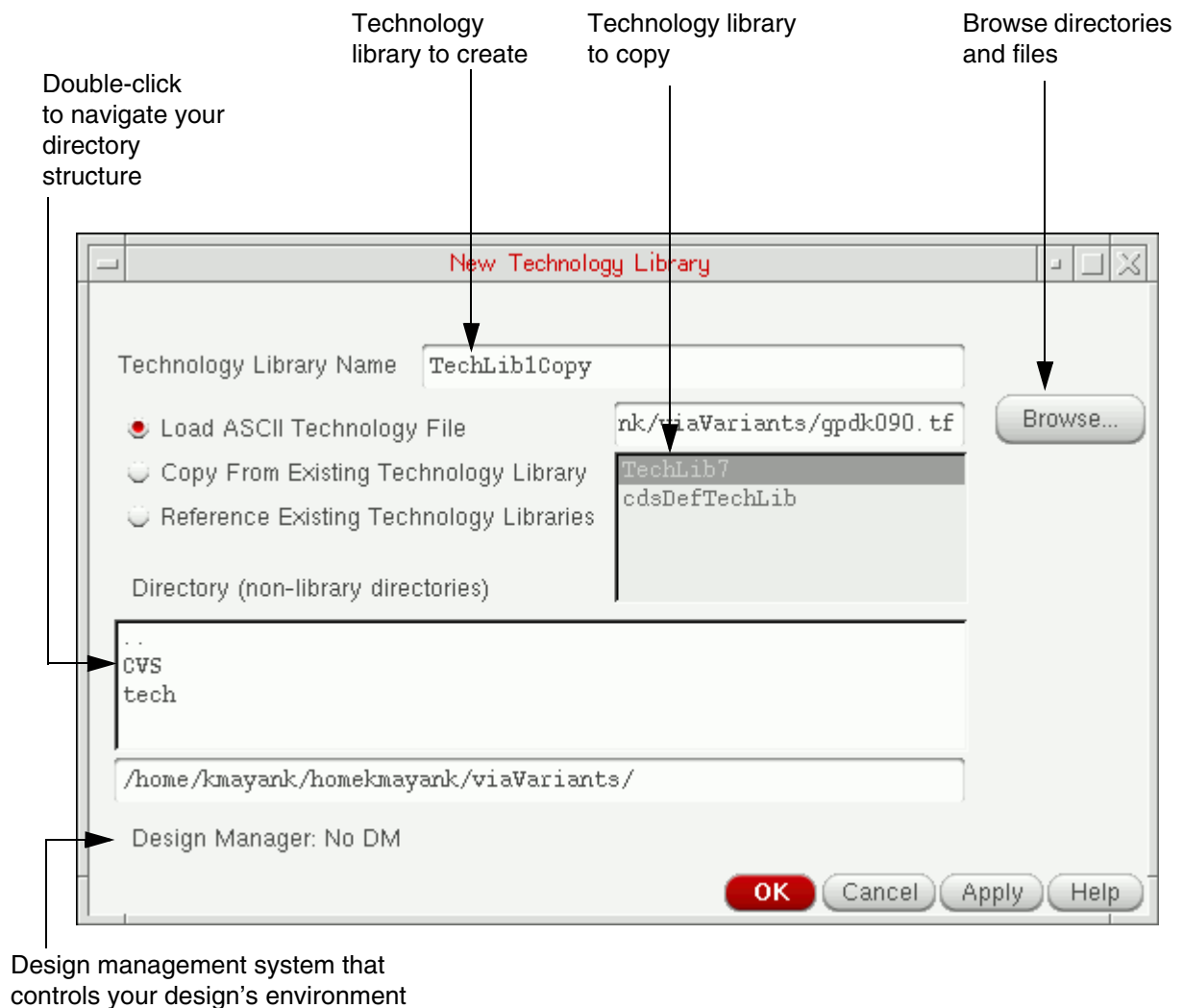
**Note:** A technology library can contain only one binary technology database, and it must always be named `tech.db`. Do not change its name.

## Creating a New Technology Library from an Existing Technology Library

You can create a new technology library by copying an existing technology library to a new library of a different name. You can then edit the new library for other purposes or to add to a different incremental technology database graph.

1. From the Technology File Manager, choose *New*.

The New Technology Library form appears.



2. In the *Technology Library Name* field, type the name of the new technology library to create.
3. Click *Copy From Existing Technology Library*.

**Note:** If an OA library contains CDB files, the library will not appear on the list of libraries for the attached technology form.

4. In the list box, highlight the technology library you want to copy.

For a description of this form, see [Appendix A](#).

5. Choose the directory in which you want to create the new technology library.

- ☐ To descend the directory hierarchy, double-click a directory name.
- ☐ To ascend the directory hierarchy, double-click the “..” directory.
- ☐ To choose a directory, click once to highlight the directory name.

**Note:** If you do not highlight a directory, the software selects the current directory by default.

6. Click *OK*.

The system copies the existing technology library to the new technology library.

**Note:** The name of the binary technology database must be `tech.db`. Do not change it. A technology library can contain only one binary technology file.

## Creating a New Technology Library that References Existing Technology Libraries

You can create a new technology library that references one or more other technology libraries. If you are creating a new library that reference existing technology libraries, the technology database of the library is kept in the read-only mode.

When copying libraries within an ITDB graph, if one or more of the libraries that is referenced is missing, the copy is still completed. A warning displays in the CIW indicating that a referenced library is missing.

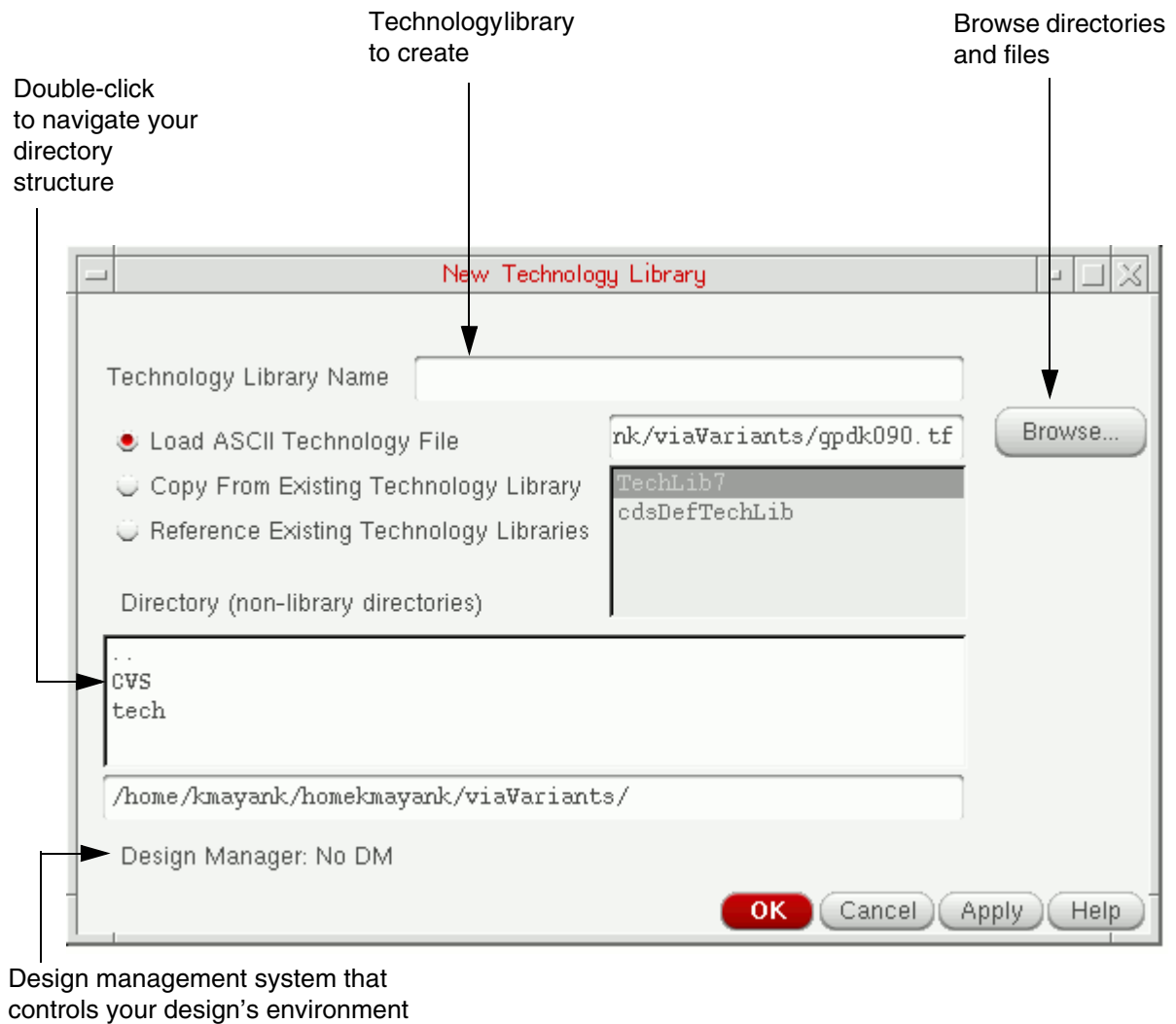
1. From the Technology File Manager, choose *New*.



## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

The New Technology Library form appears.



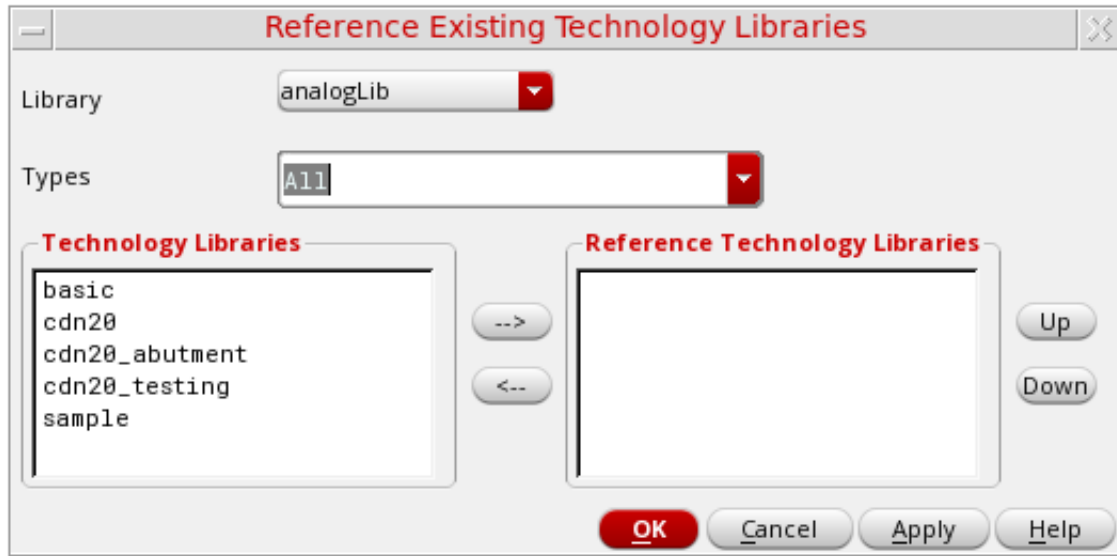
2. In the *Technology Library Name* field, type the name of the new technology library to create.
3. Click *Reference Existing Technology Libraries*.  
**Note:** You can use the Browse button to load the library from a disk.
4. Click *Apply* or *OK*.

## Virtuoso Technology Data User Guide

### Preparing Files for Use with a Design

---

The Reference Existing Technology Libraries form appears.



For a description of this form, see [Appendix A](#).

5. (ICADVM20.1 Only – Virtuoso MultiTech Framework) In the *Types* field, select the kinds of libraries to show. By default, all libraries are shown and the value is set to `All`. You can select `Package` or `IC` to see only the relevant set of libraries.
6. For each technology library you want to reference, in the order in which you want them referenced (top to bottom):
  - a. Select the library name from the *Technology Libraries* list.
  - b. Click the right arrow button to move the selected technology library name into the *Reference Technology Libraries* list box.

**Note:** If you want to change the order of the Reference Technology Libraries, you can highlight a library name and click *Up* or *Down* to move it up or down in the list.

7. Click *OK*.

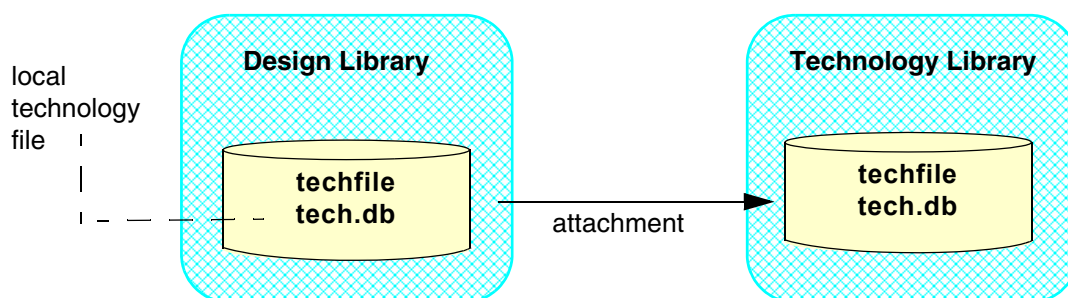
The system creates the new library, which contains only references to the selected reference libraries, whose reference libraries then also come along with their existing graph structure. You can add technology data to this library.

## Referencing or Attaching a Technology Library

Any library can either reference a technology library or attach to a technology library. A design library must do one or the other to have access to technology data during a design session. A technology library can also reference or attach to another technology library; alternatively, it can be standalone by doing neither, using only the `cdsDefTechLib` default technology data. Choosing whether to reference or attach a technology library depends upon whether or not designers need to specify technology data during their design sessions.

- Referencing a technology database from a design library or another technology library protects the integrity of any read-only data in the effective technology library while providing a writable local library where designers can define technology data. Referencing is preferable to attachment when designers need a writable local technology database in which to add technology data (such as data output by LEFIN).
- Attaching is preferable in cases where designers use only predefined technology data, which is typically read-only.

**Note:** Virtuoso does not allow a technology library that has a local technology file to be attached to another technology library. When you try to do this, it issues error `TECH-2000203`. You need to either detach the second library by using `techUnattachTechFile(ddGetObj(<otherTechLibraryName>))` or remove the local technology file by using `techDeleteTechFile(techGetTechFile(ddGetObj(<techLibraryName>)))`. For more information about these functions, see [\*Virtuoso Technology Data SKILL Reference\*](#).



The design library has an ambiguous technology setup because it contains a local techfile but is also attached to another technology library. The local techfile takes precedence, and the attachment will be ignored.

## Referencing a Technology Library from a Design Library

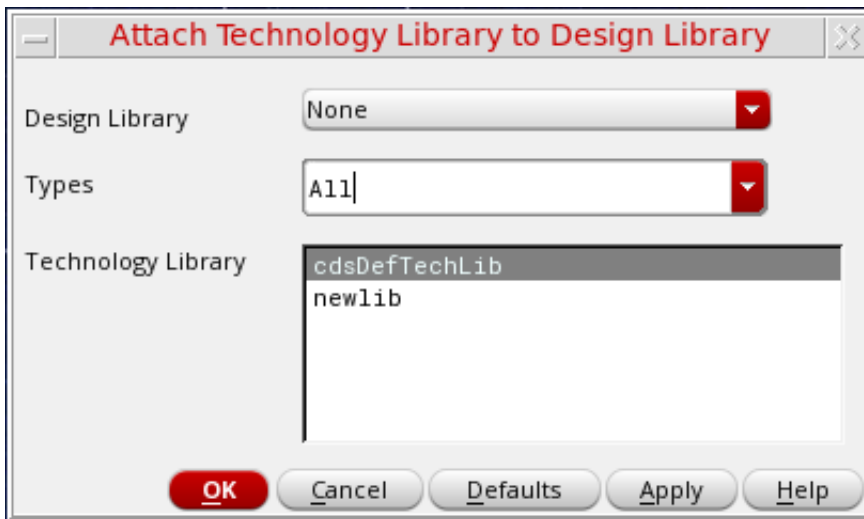
You can set up a reference from a design library to a technology library with the Library Manager or from the CIW when you create a new library. For details, refer to the [Cadence Library Manager User Guide](#) and the [Virtuoso Design Environment User Guide](#).

## Attaching a Technology Library to a Design Library

To attach a technology library to a design library:

1. From the Technology File Manager, choose *Attach*.

The Attach Technology Library to Design Library form appears.



For a description of this form, see [Appendix A](#).

2. In the *Design Library* field, select the design library to which you want to attach the technology library.
3. (ICADVM20.1 Only – Virtuoso MultiTech Framework) In the *Types* field, select the kinds of libraries to show. By default, all libraries are shown and the value is set to `All`. You can select `Package` or `IC` to see only the relevant set of libraries.
4. From the *Technology Library* field, choose the library containing the technology library you want to attach to the design library.
5. Click *OK*.

The technology library is attached to the design library.

## **Ensuring Desired Display Resource File Usage**

Keep in mind how the software loads and uses display resource files upon initialization to ensure that you are using display resources in the way you want. For details, refer to [“How Cadence Design Software Handles Multiple Display Resource Files”](#) on page 42.

# **Virtuoso Technology Data User Guide**

## **Preparing Files for Use with a Design**

---

---

## Editing, Reusing, and Merging Technology File Data

---

This chapter discusses the following:

- [Updating Technology Files and Technology Libraries](#) on page 72
- [The Techfile IDE](#) on page 73
- [Reusing Technology Data to Build a New Technology Library](#) on page 96
- [Loading Technology Data into Virtual Memory](#) on page 101
- [Discarding an Edited Technology Database from Virtual Memory](#) on page 106
- [Saving a Technology Library Edited in Virtual Memory to Disk](#) on page 107

## Updating Technology Files and Technology Libraries

You can create and update technology files in one of the following ways:

- By using the Techfile IDE, which is recommended.
- By using a text editor.
- By dumping an ASCII technology file from an existing technology library.

You can use the following methods to edit an existing technology database:

- Edit an ASCII file in a text editor and use the *Technology File – Load* command to compile and load it into virtual memory, either merging it with the technology data already in virtual memory or replacing the technology data already in virtual memory with it.
- Use SKILL functions to load a binary technology file and update it in memory.

Technology data changed in virtual memory can be used during a design session without changing the original technology database saved on disk. If you want to save changes permanently, save the edited technology data to the current technology library on disk.

### *Important*

In the process of introducing data into a incremental technology database structure, any conflicts existing with other *open* technology databases

### *Important*

Conflict checking applies only to open technology databases in a graph.

### *Important*

When you load an edited technology file or merge or replace data from another technology file, the operation checks for conflicts in all open technology databases in the graph, both up and down the graph. Conflicts are reported and loading aborted as applicable. If, however, you introduce a conflict with a database that is higher in the complete graph but not open, the conflict is not relevant to the open databases and is not reported. When you do open the higher-level database, the conflict becomes apparent and is reported in the graph.



## The Techfile IDE

The Virtuoso platform relies on proper technology information. At advanced nodes, more and more constraints need to be captured in technology files. More importantly, they need to be grouped correctly within constraint groups. When done manually, capturing this data is a cumbersome, error-prone, and time-consuming process. Moreover, the amount of text involved cannot be handled efficiently by conventional text editors, especially when complex tables are involved.

The Techfile IDE makes it easier for you to understand, write, and modify ASCII technology files, particularly constraints. It also helps you visualize the organization of constraints in constraint groups in a compiled technology database.



### Video

For a quick overview, see the video [Virtuoso Techfile IDE](#) on Cadence Online Support.

For more information about the syntax of the ASCII technology file, see [Virtuoso Technology Data ASCII Files Reference](#).

For generic information about the IDE, which is based on the Cadence SKILL IDE, see [Cadence SKILL IDE User Guide](#).

This section presents specific information about the Techfile IDE and contains the following topics:

- [Starting the Techfile IDE](#)
- [Creating a Technology File](#)
- [Navigating through a Technology File](#)
- [Editing a Technology File](#)
- [Working with the Rule Editor](#)
- [Checking a File for Errors](#)
- [Loading a Technology File into a Technology Library](#)

## Starting the Techfile IDE

To start the Techfile IDE, use one of the following options:

- At a terminal, type the following command:

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

---

```
$> virtuoso -techhide technology_file_name
```

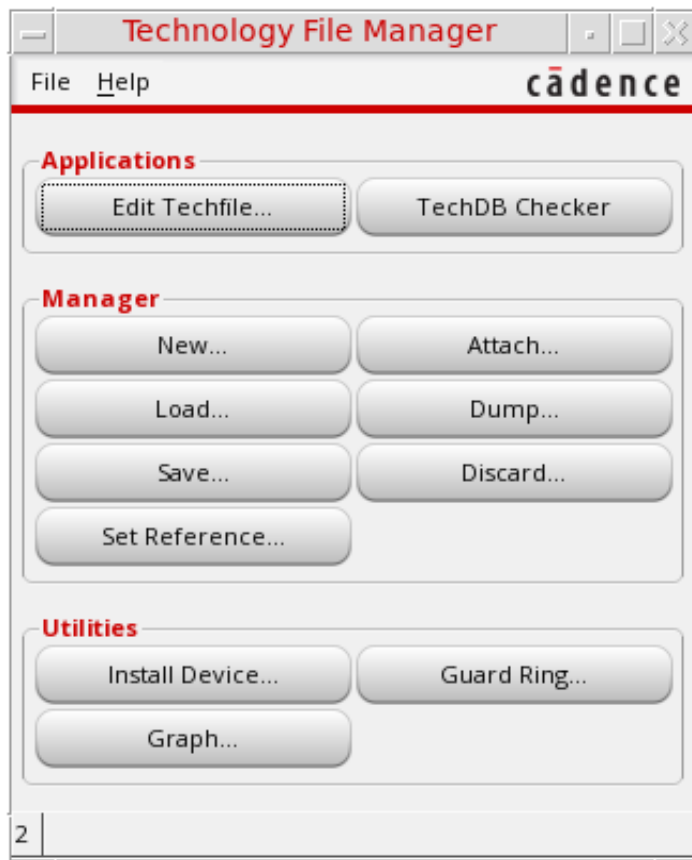
You can also open multiple technology files together, for example, as follows:

```
$> virtuoso -techhide file1.tf file2.tf file3.tf &
```

■ In the CIW:

a. Choose *Tools – Technology File Manager*.

The Technology File Manager window appears.



b. Click *Edit Techfile*.

Alternatively, to create a technology file from a technology library, click Dump.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

---

In the Dump Technology File form that appears, select the *Open in Techfile IDE* check box.

**Dump Technology File**

Technology Library: **cdsDefTechLib**

☐ Select All ☐ Dump Empty Section Headers

- ☐ controls
- ☐ layerDefinitions
- ☐ layerRules
- ☐ siteDefs
- ☐ viaDefs
- ☐ packaging
- ☐ viaSpecs
- ☐ constraintGroups
- ☐ leRules
- ☒ devices
  - ☐ cdsVia
  - ☐ cdsGuardRing
  - ☐ User Defined Device
  - ☐ MPP
  - ☐ extractDevice
  - ☐ waveguide
  - ☐ ruleContact

ASCII Technology File:  **Browse...**

Open in Techfile IDE ☐

**OK** **Cancel** **Defaults** **Apply** **Help**

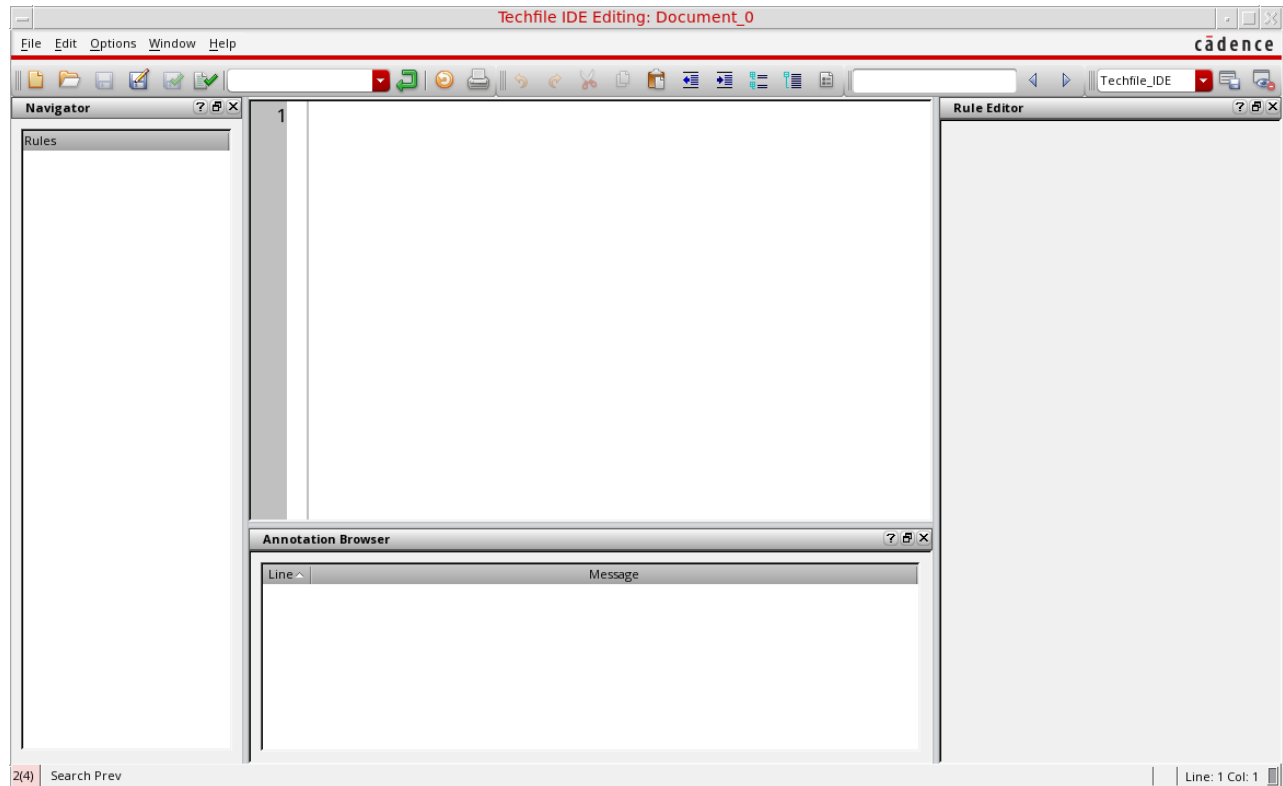
- Use the `tfEditTechfile` SKILL function.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

---

The Techfile IDE window opens, showing either a blank technology file or a specified one.

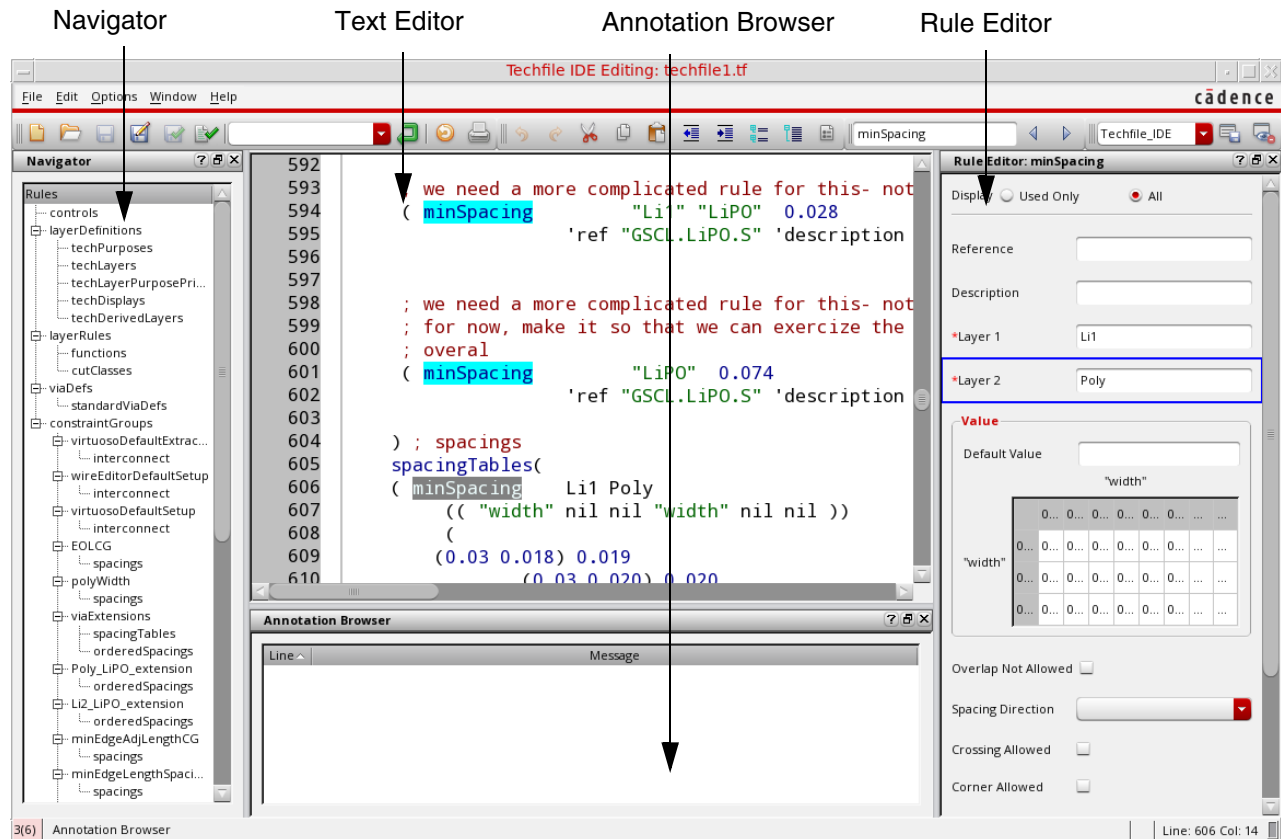


For detailed information about various elements of the UI and how to customize them, see *Cadence SKILL IDE User Guide*. This section presents only features that are specific to technology files.

# Virtuoso Technology Data User Guide

## Editing, Reusing, and Merging Technology File Data

When a technology file is open, the Techfile IDE window appears as follows.



The most important elements of the interface are briefly described as follows:

- **Navigator:** An assistant that shows a hierarchical list of the contents of the technology file. The entries at the primary level are constraint groups and those at the secondary level are types of constraints. You can click a node or leaf to jump to it in the editor. See [“Navigating through a Technology File”](#) on page 79.
- **Text Editor:** The main window in the center. You can work with it by typing in it or using the Edit menu or the contextual menu, which appears when you right-click the window. It has several additional features. See [“Editing a Technology File”](#) on page 81.
- **Rule Editor:** An assistant that presents parameters and values for a selected constraint in a form. You can update the rule in this window without having to consider the nuances of syntax and formatting. See [“Working with the Rule Editor”](#) on page 87.
- **Annotation Browser:** An assistant that lists syntax errors identified in the active technology file after it is checked for errors. See [“Checking a File for Errors”](#) on page 92.

## tfEditTechfile

```
tfEditTechfile(  
    l_fileList  
    t_application  
=> t / nil
```

### Description

Starts the Techfile IDE in a standalone mode.

### Arguments

<i>l_fileList</i>	A list of ASCII technology files to be opened in the Techfile IDE.
<i>t_application</i>	The name of the application to be opened in standalone mode. The value nil can be used to indicate the default application, which is the Techfile IDE.

### Value Returned

<i>t</i>	The files were opened in the Techfile IDE.
<i>nil</i>	The files could not be opened in the Techfile IDE.

### Example

Opens the three technology files specified in separate tabs in the Techfile IDE.

```
tfEditTechfile(  
    ?fileList list("test1.tf" "test2.tf" "test3.tf")  
    ?application nil)  
=> t
```

## Creating a Technology File

To create a new ASCII technology file:

### 1. Choose *File – New*.

A new file named `Document_n` is opened. Here, *n* is a number that is incremented each time you create a new file using this option.

Optionally, choose *File – Open*. Then, select a technology file that you want to modify. You can select any other valid technology file or the sample ASCII technology file in the Cadence installation folder.

**Note:** To open a file only to view it, choose *File – Open for Read*.

2. Enter text or modify it, as required. See the remaining topics in this section for more details.

Ensure that you follow the structure of and requirements for specifying ASCII technology file data as described in [Virtuoso Technology Data ASCII Files Reference](#).

See “[Navigating through a Technology File](#)” on page 79 for information about viewing various sections of the file.

While entering or modifying constraints, the Rule Editor provides an easy way to enter details and visualize them. See “[Working with the Rule Editor](#)” on page 87.

**Note:** The Navigator panel appears disabled as you start modifying the file.

3. Check the file for errors.

See “[Checking a File for Errors](#)” on page 92.

4. Optionally, to undo changes edit by edit, choose *Edit – Undo*. To discard all edits, choose *File – Discard Edits*.
5. Choose *File – Refresh* to re-parse the file.

The Navigator panel appears enabled again with its contents refreshed.

6. Choose *File – Save As* to save the data into a filename of your choice.

Alternatively, choose *Save* to save the file by the same name.

**Note:** Technology files are saved with the `.tff` extension.

7. Optionally, to make the file read-only, choose *File – Make Read Only*.
8. Choose *File – Close*.

**Note:** Shortcut keys are mentioned next to command names on menus. Most menu options are also available on the toolbar.

## Navigating through a Technology File

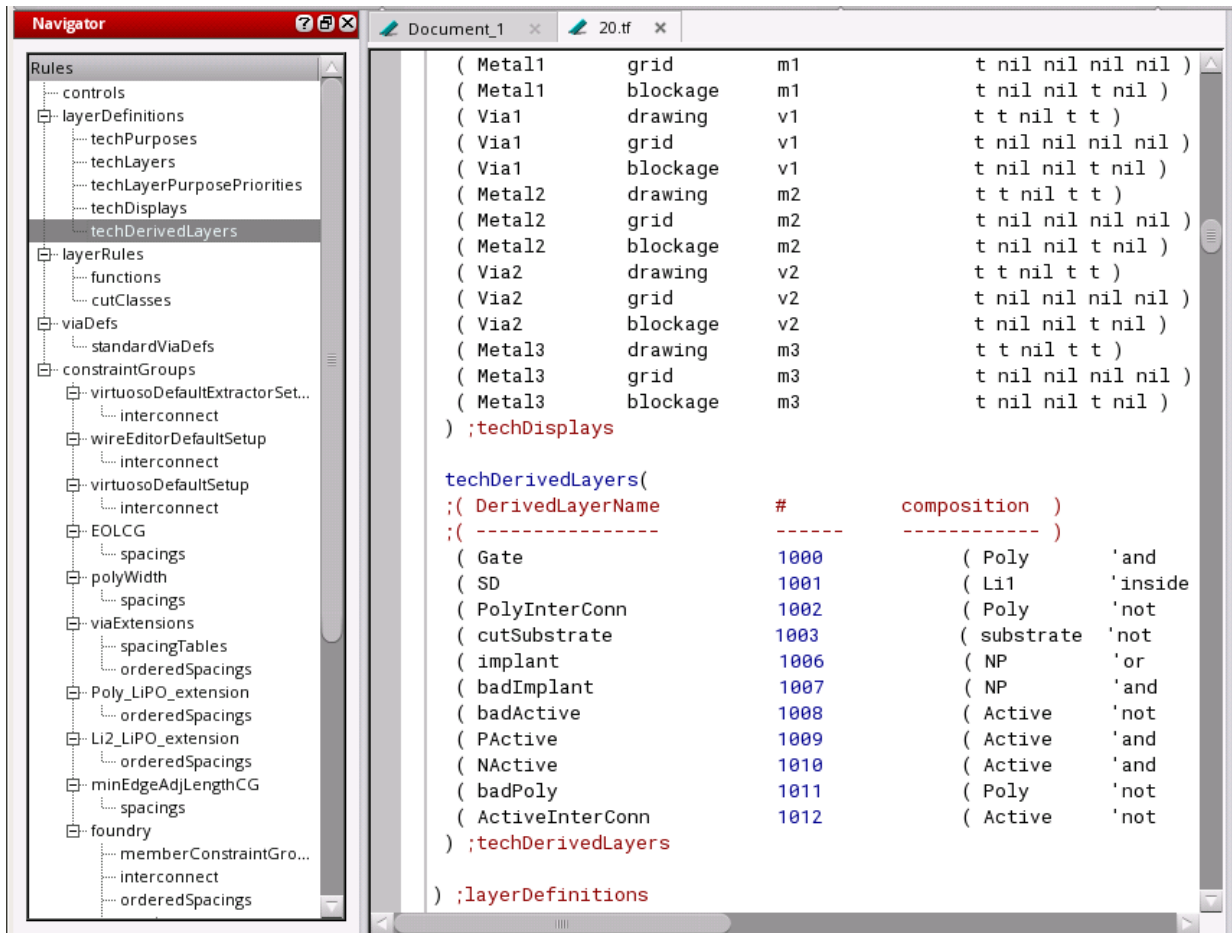
A technology file is usually a very large document. Navigating back and forth through it can be tedious with simple text editors. The Techfile IDE offers several ways to navigate efficiently through technology files.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

To navigate through a technology file, use one of these options:

- Choose a node in the Navigator assistant to jump to the related part of the file in the text editor.



Every header section in the technology file and constraint groups are represented by a node in the Navigator assistant. The tree does not include details such as individual layers, vias, and constraints. The section where the pointer is currently positioned is highlighted.

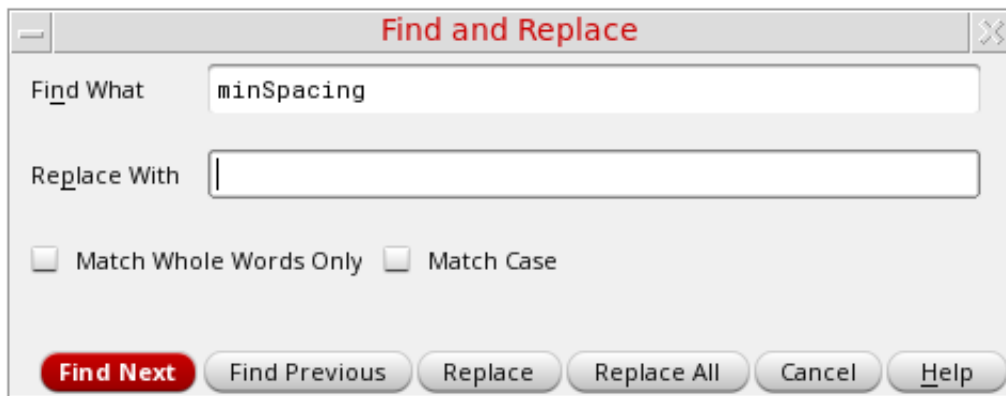
**Note:** If the Navigator is not visible, choose *Window – Assistants – Navigator*. The Navigator appears docked at the right edge of the application window.

- Find text quickly by typing it in the search window on the toolbar and clicking the *Search* buttons next to it.





- Choose *Edit – Find/Replace* to open the Find and Replace form and use the options available to find and replace text.



- When a parenthesis is selected, the matching parenthesis also appears selected.

```
techParams(  
;( parameter          value          )  
;( -----          -----          )  
 ( minFingerCountInGroup  7          )  
)| ;techParams
```

For a large segment, where both parentheses might not be in the same view, select a parenthesis and choose *Edit – Go to Matching Parenthesis* to jump to the matching parenthesis.

To jump to the preceding outer parenthesis, press `Ctrl+]`. To jump to the next inner parenthesis, press `Ctrl+[`.

## Editing a Technology File

The *Edit* menu offers standard options to work with the text in a technology file. These include *Cut*, *Copy*, *Paste*, *Select All*, *Undo*, and *Redo*.

Unlike conventional text editors, the Techfile IDE offers several other useful features that help you work more efficiently with technology files. This section describes these features and how to work with them.

## Setting Options

The Options form helps you set generic editor options and color options in the text editor.

To set options:

1. Choose *Options – Editor or Color Settings*.



2. Set options as required.

These options are described in subsequent topics.

3. Close the form.

The changes that you make persist across sessions.

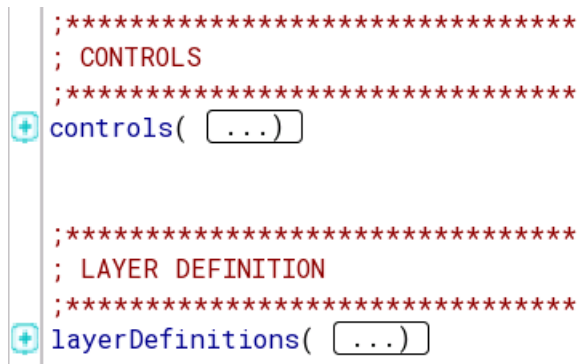
## Working with the Outline View

The Outline view helps you collapse and expand parts of a technology file. It helps you focus on the segments that you want to work with.

To enable the Outline view:

- ➔ Click the *Outline View* button () on the toolbar.

The outline is marked in turquoise and the contents are collapsed by default, as indicated by the plus icons. You can expand a segment by clicking the plus icon or the ellipsis in the box next to the visible word of the segment.




```

;*****
; CONTROLS
;*****
controls( ...)

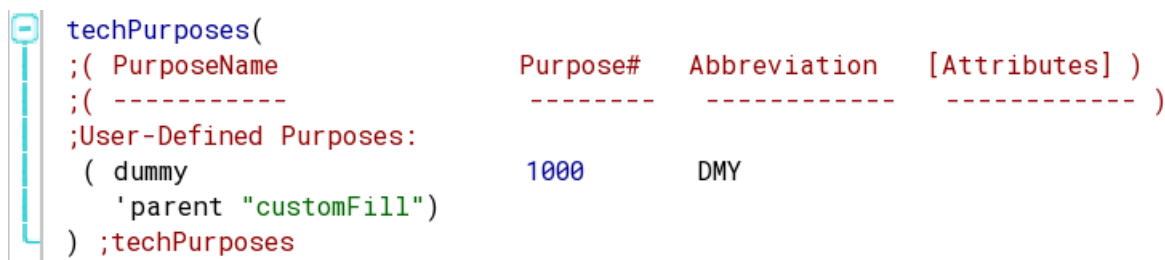
;*****
; LAYER DEFINITION
;*****
layerDefinitions( ...)
  
```

To expand all segments of the file with one action:

- ➔ Click the *Expand All* button () on the toolbar.

By default, the Outline view does not collapse or expand individual constraints or groups. To enable the outline for a single constraint:

- ➔ Double-click the text just before the opening parenthesis, for example, `techPurposes`, in the following screenshot.



```

techPurposes(
;( PurposeName           Purpose#   Abbreviation   [Attributes] )
;( -----             -----   -----   )
;User-Defined Purposes:
( dummy                1000      DMY
  'parent "customFill")
) ;techPurposes
  
```

You can collapse a segment by clicking the toggle minus icon in the margin.

## Showing Line Numbers

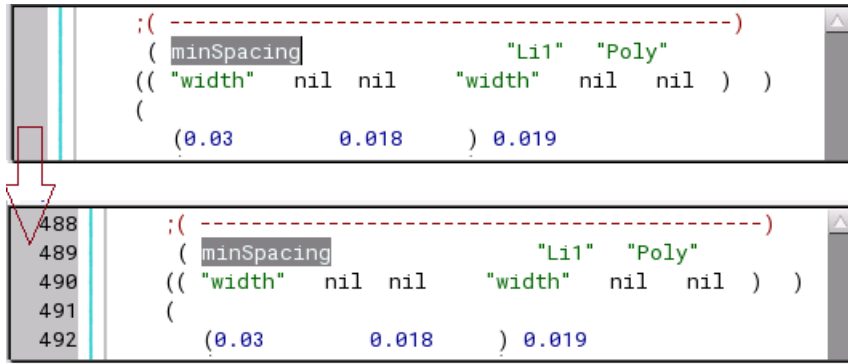
To display line numbers in the editor:

1. Choose *Options – Editor*.

The Options form opens with the Editor tab displayed.

## 2. Select *Show Line Number*.

Line numbers appear in the margin to the left.



## Highlighting Matching Text

To highlight matching text in a technology file:

1. Choose *Options – Editor*.
2. Select *Highlight Matching Text*.

When a string is selected in the text editor, matching strings appear highlighted.

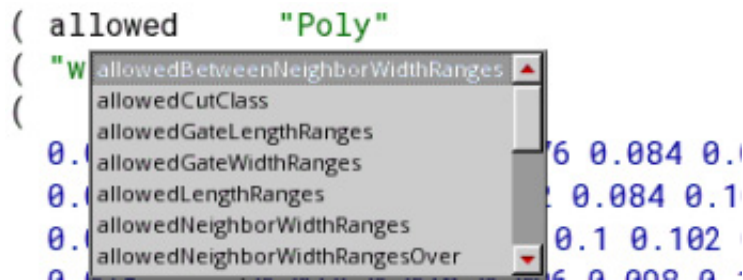


## Enabling Name Completion

To enable name completion while typing in the editor:

1. Choose *Options – Editor*.
2. Select *Enable Name Completion*.

While typing the name of a constraint, a list appears with matching names. You can select a name rather than type it out.



## Applying Indentation and Tab Spaces

To apply indentation automatically:

1. Choose *Options – Editor*.
2. Select *Auto Indent*.

To insert white spaces for each Tab press:

- ➔ Select *Emulate Tabs*.

To specify the number of characters by which the cursor moves with each Tab press:

- ➔ Specify a value in the *Tab Size* field.

## Modifying Fonts

To change fonts:

1. Choose *Options – Editor*.
2. Specify a value in the *Font Size* field.
3. Select a value in the *Font Style* field.

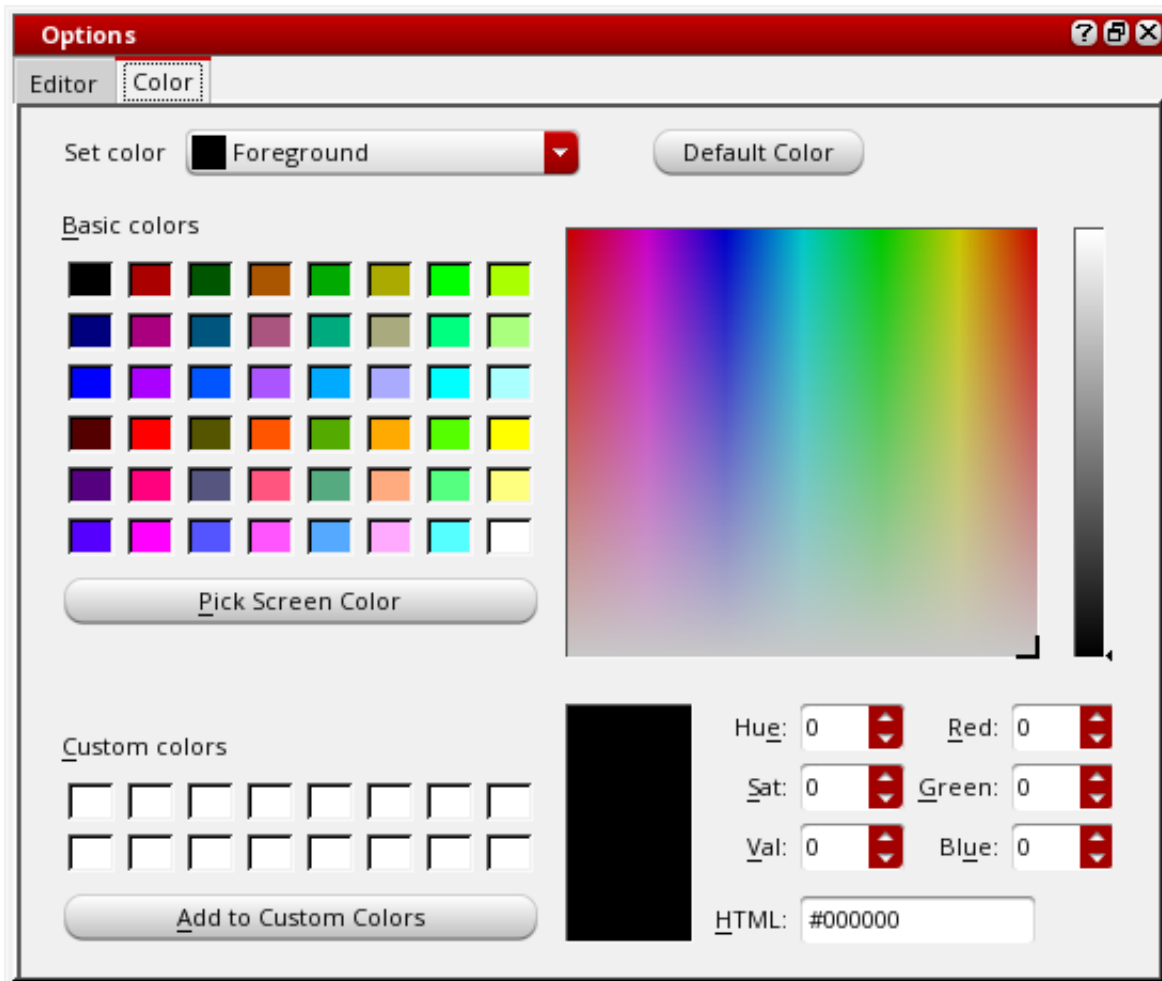
## Modifying Colors

A technology file opened in the editor uses colors to visually distinguish between different types of information.

To modify color settings:

1. Choose *Options – Color Settings*.

The Options form opens with the Color tab displayed.



2. Select an option from the *Set color* list, for example *Mismatching Bracket*, and set a color for it, as required.

**Note:** To reset colors to their original settings, click *Default Color*. Default colors are set in the `.cdsenv` file located at `tools/dfII/etc/tools/SkillIDE`.

Text does not reflect its color assignment when you type a keyword that is not recognized by the editor, for example the misspelled word `layerDefinitons` in the illustration on the right

# Virtuoso Technology Data User Guide

## Editing, Reusing, and Merging Technology File Data

below is not blue. This provides a visual indication that the keyword has been typed incorrectly.

```
;*****  
; LAYER DEFINITION  
;*****  
layerDefinitions(  
  (
```

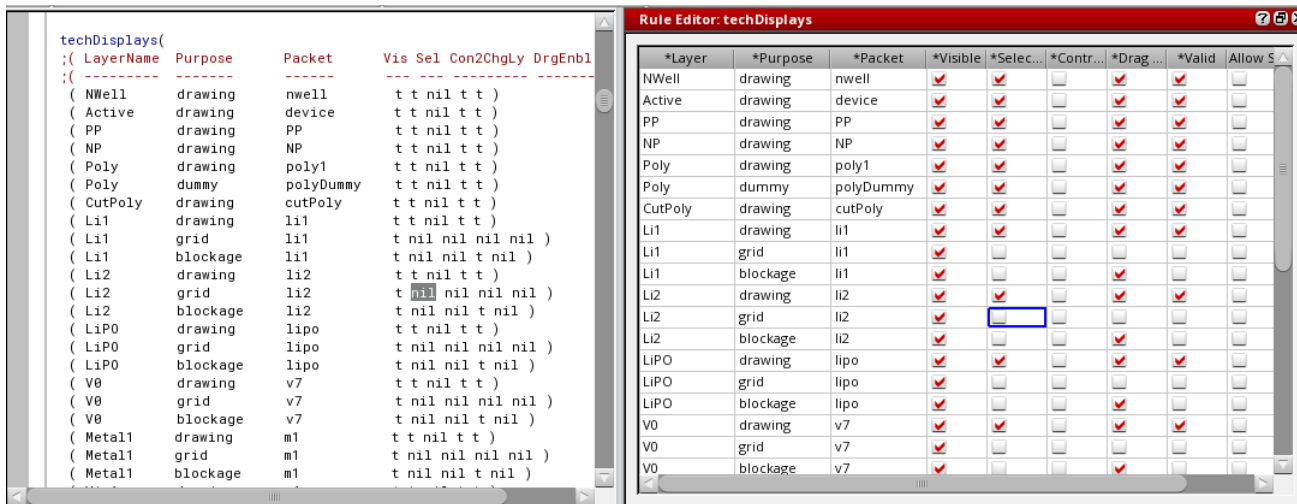
```
;*****  
; LAYER DEFINITION  
;*****  
layerDefinit|pns(  
  (
```

## Working with the Rule Editor

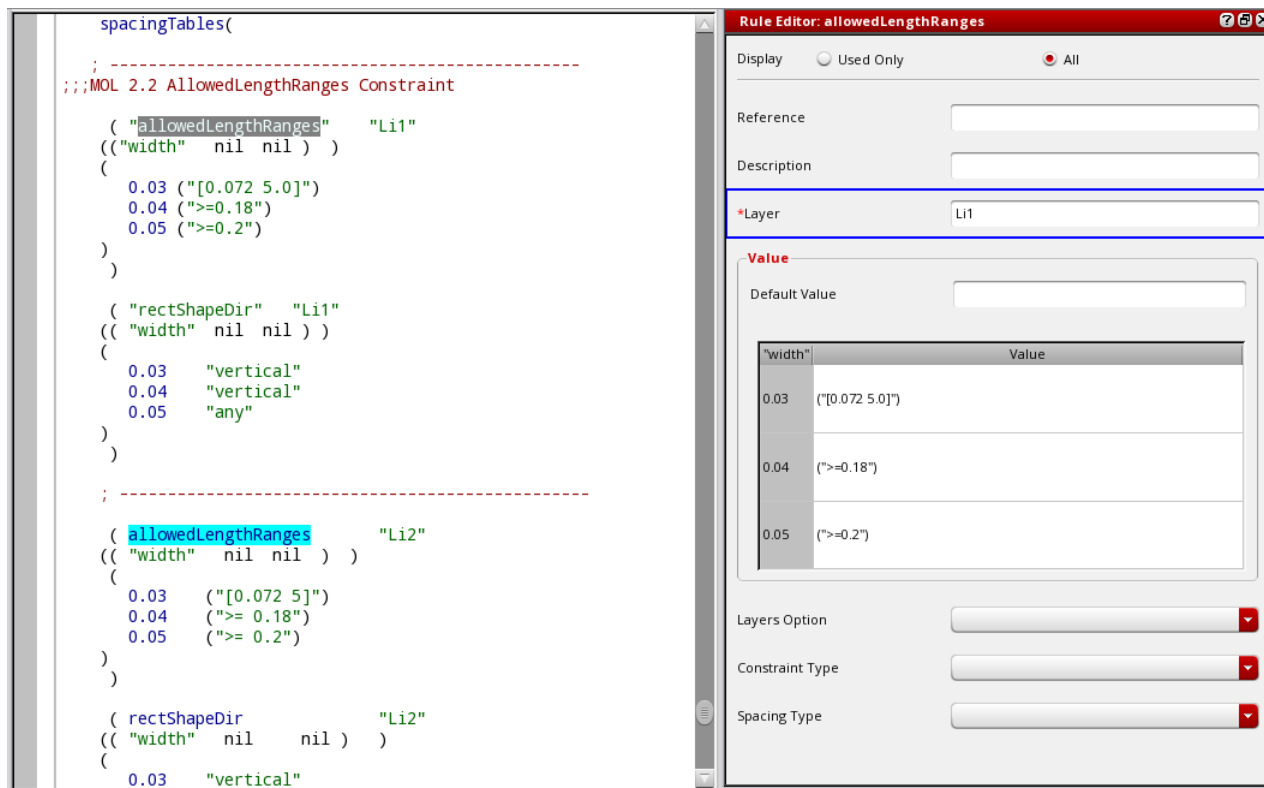
The Rule Editor displays a selected constraint or statement in a separate window, in which you can work more easily because you do not need to manually format any of the text.

All constraints and these statements are supported by the Rule Editor: `techPurposes`, `techLayers`, `techLayerPurposePriorities`, `techDisplays`, and `techDerivedLayers`.

In the following screenshot, the `techDisplays` statement is selected in the text editor and its parameters and values are shown in the Rule Editor.



In the following screenshot, the `allowedLengthRanges` constraint is selected.



**Note:** If the Rule Editor is not visible, choose *Window – Assistants – Rule Editor*. The Rule Editor appears docked at the right edge of the application window.

Some features of the Rule Editor are briefly mentioned as follows. Other features are described in detail in subsequent topics.

- All parameters possible for a constraint or statement are shown by default, and the *All* radio button appears selected.

You can select *Used Only* to view parameters that are currently used for a constraint. This is a read-only view.

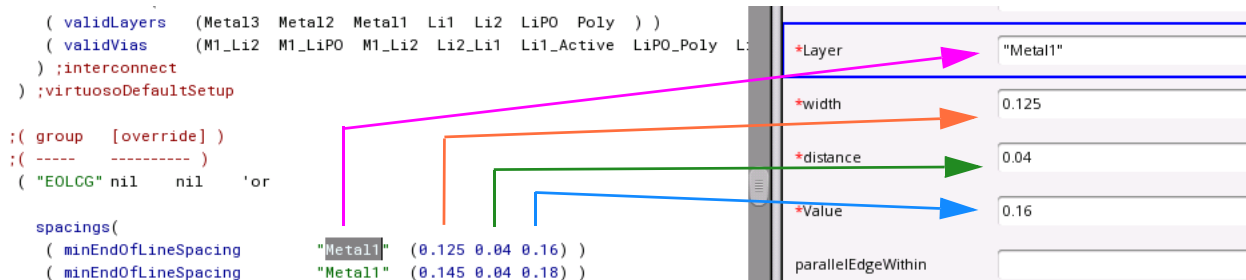
- Mandatory parameters, such as `Layer` in this case, are marked by asterisks.



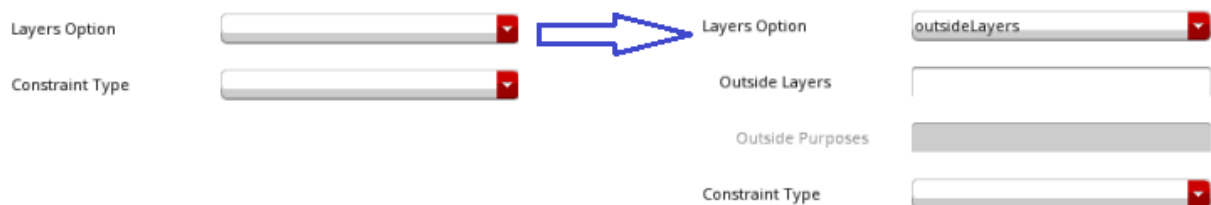
## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

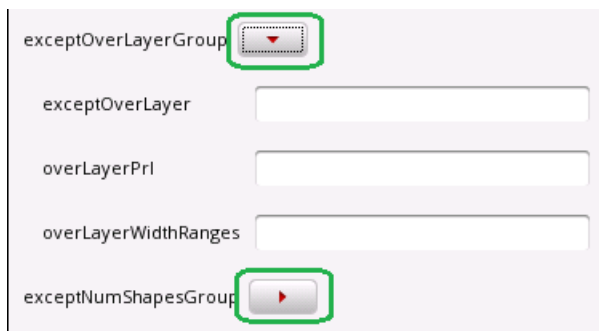
- Positional parameters, where a parameter is indicated only by its position on a row in the file, are sequenced accordingly in the Rule Editor.



- Parameters that can have only a limited number of values have drop-down lists from which you can select a value.
- Parameter nesting is shown using indentation, for example, *Outside Purposes* is nested in *Outside Layers*. Nested parameters are not shown unless the parent parameter is selected or populated with a value. When a value is selected, the nested parameters pertinent to the selected value are shown.



- Related parameters can be collapsed or expanded by using the arrow buttons indicated.



- A selected value in the text editor appears cross-highlighted with a box outline in the Rule Editor.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

---

When you select a value in the Rule Editor, it appears highlighted in the text editor.

```
( allowedLengthRanges      "Li2"  
( ( "width"    nil  nil  ) )  
(  
  0.03    (" [0.072 5]" )  
  0.04    (">= 0.18" )  
  0.05    (">= 0.2" )
```

"width"	Value
0.03	(" [0.072 5]" )
0.04	(">= 0.18" )
0.05	(">= 0.2" )

When you select a table index value in the Rule Editor, all related values appear highlighted in the text editor.

When you change a value, it appears highlighted in the text editor, which draws your attention so you can check your changes before moving the pointer away.

```
( allowedLengthRanges      "Li2"  
( ( "width"    nil  nil  ) )  
(  
  0.03    (" [0.072 5]" )  
  0.04    (">= 0.18" )  
  0.05    (">= 0.2" )
```

"width"	Value
0.03	(" [0.072 5]" )
0.04	(">= 0.18" )
0.05	(">= 0.2" )

The highlighting disappears when you move the pointer to another part of the file.

## Modifying Tables in the Rule Editor

Tabular information in a technology file is effectively represented using tables in the Rule Editor. You can further modify such tables by typing directly in the table for convenience.

The screenshot shows the Rule Editor interface. On the left is a code editor with a table definition. On the right is a visual table editor for the 'CutClass Value' table.

**Code Editor (Left):**

```
spacingTables(  
  ( minOutClassSpacing "Via1"  
    (( "rowHeader" ("VA" ("VB" 'shortEdge) ("VB" 'longEdge)  
      "colHeader" nil nil ) 'paraOverlap 0 'sameMetal  
      'cutClassProfile  
        (( "rowHeader" (( "VB" 'shortEdge) ("VB" 'longEdge)  
          "colHeader" ("VA" ("VB" 'shortEdge) ("VB" '1  
    )  
  )  
  ; 1st row  
  (( "VB" 'shortEdge) ("VA" 'shortEdge) ) 0.001  
  (( "VB" 'shortEdge) ("VA" 'longEdge) ) 0.001  
  (( "VB" 'shortEdge) ("VB" 'shortEdge) ) 0.003  
  (( "VB" 'shortEdge) ("VB" 'longEdge) ) 0.009  
  (( "VB" 'shortEdge) ("VC" 'shortEdge) ) 0.005  
  (( "VB" 'shortEdge) ("VC" 'longEdge) ) 0.005  
  ; 2nd row  
  (( "VB" 'longEdge) ("VA" 'shortEdge) ) 0.007  
  (( "VB" 'longEdge) ("VA" 'longEdge) ) 0.007  
  (( "VB" 'longEdge) ("VB" 'shortEdge) ) 0.009  
  (( "VB" 'longEdge) ("VB" 'longEdge) ) 0.01  
  (( "VB" 'longEdge) ("VC" 'shortEdge) ) 0.011  
  (( "VB" 'longEdge) ("VC" 'longEdge) ) 0.011  
  )  
  ; 3rd row  
  (( "VA" 'shortEdge) ("VA" 'shortEdge) ) (0.008 'centerToCe  
  (( "VA" 'shortEdge) ("VA" 'longEdge) ) (0.008 'centerToCe  
  (( "VA" 'shortEdge) ("VB" 'shortEdge) ) 0.009  
  (( "VA" 'shortEdge) ("VB" 'longEdge) ) 0.01  
  (( "VA" 'shortEdge) ("VC" 'shortEdge) ) 0.014  
  (( "VA" 'shortEdge) ("VC" 'longEdge) ) 0.014  
  ; 2nd row  
  (( "VA" 'longEdge) ("VA" 'shortEdge) ) (0.008 'centerToCe  
  (( "VA" 'longEdge) ("VA" 'longEdge) ) (0.008 'centerToCe  
  (( "VA" 'longEdge) ("VB" 'shortEdge) ) 0.009  
  (( "VA" 'longEdge) ("VB" 'longEdge) ) 0.01  
  (( "VA" 'longEdge) ("VC" 'shortEdge) ) 0.014  
  (( "VA" 'longEdge) ("VC" 'longEdge) ) 0.014  
  ; 3rd row  
  (( "VB" 'shortEdge) ("VA" 'shortEdge) ) 0.009  
  )  
)
```

**Visual Table Editor (Right):**

**CutClass Value**

Default Value: 0.099

		"colHeader"			
		"VA"	"VB" 'shortEdge	"VB" 'longEdge	"VC"
"rowHeader"	"VA"	0.008 'centerToCenter	0.009	0.01	0.014
	"VB" 'shortEdge	0.009	0.015	0.016	0.017
	"VB" 'longEdge	0.001	0.003	0.009	0.005
	"VC"	0.01	0.016	0.022	0.024
		0.007	0.009	0.01	0.011
		0.014	0.017	0.024	0.03

Connectivity Type: sameMetal

Para Overlap: 0

Overlap Direction: [Dropdown]

Exact Aligned: [Dropdown]

Line: 337 Col: 23

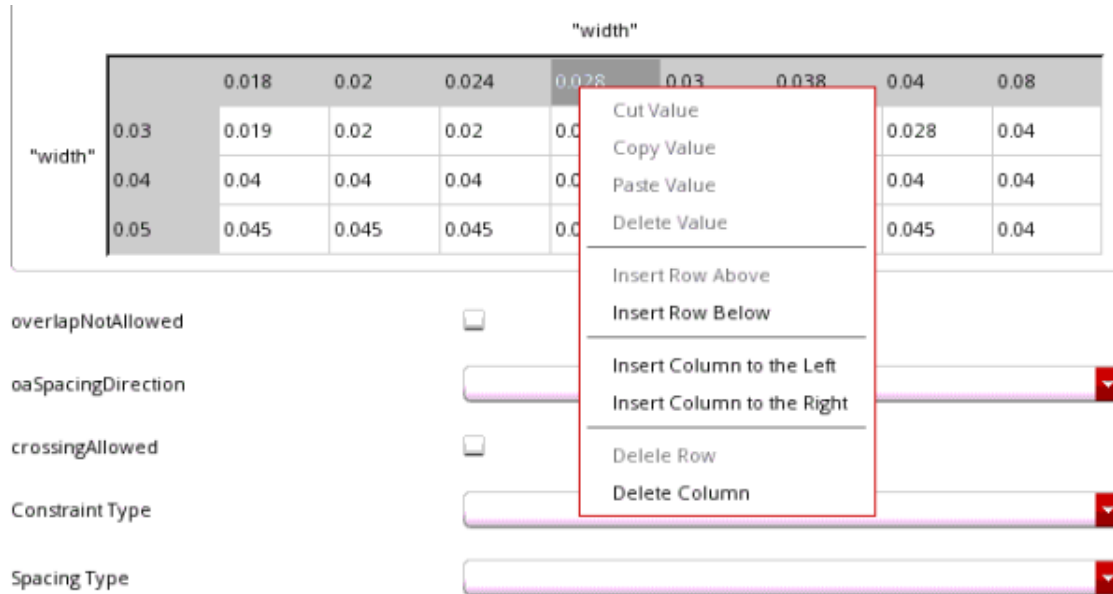
You can type directly into a table to make any changes. Additionally, the context menu, which appears when you right-click a table, offers the following options:

- Cut, copy, paste, or delete values
- Insert columns to the right or left of a selected column
- Insert rows above or below a selected row

# Virtuoso Technology Data User Guide

## Editing, Reusing, and Merging Technology File Data

### ■ Delete selected columns or rows



When you insert a column between columns or a row between rows, an interpolated value that is exactly between the values in adjacent header cells populates the index cell by default.

When you append a row or column at the beginning or the end of a table, a value incremented by 0.01 populates the index cell by default.

## Checking a File for Errors

Having made changes in a technology file, you can check it for syntax and consistency errors by using one of the following operations:

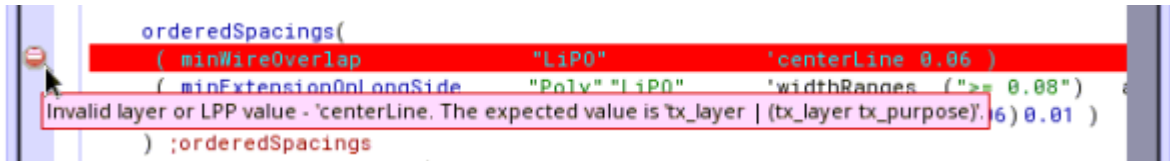
- Choose *File – Check* to check the file for errors.
- Choose *File – Check and Save* to save changes if no errors are reported during checking. If errors are found, a message appears indicating the presence of errors and seeking confirmation before saving the file.

The Techfile IDE flags errors in three ways:

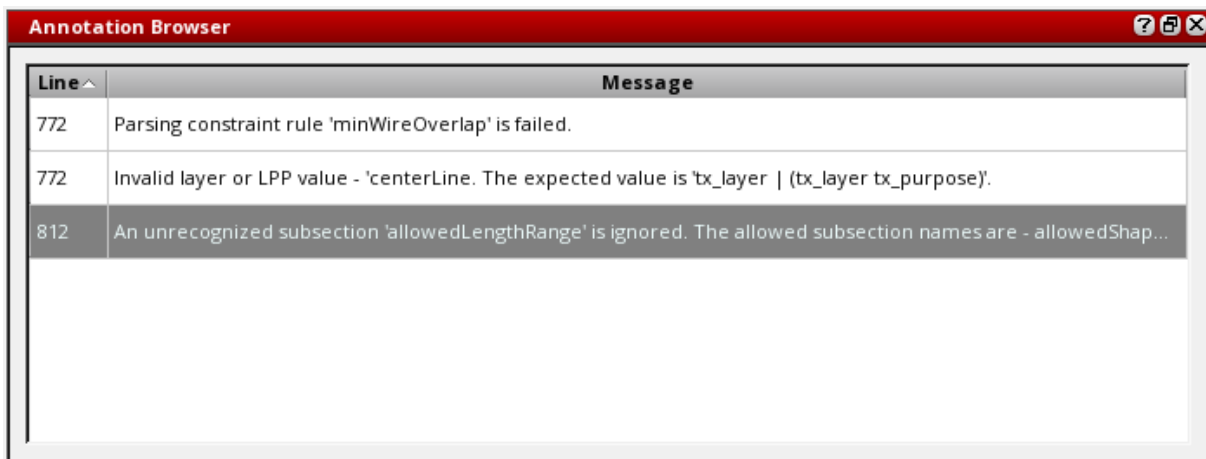
## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

- The text editor pane shows a red icon in the margin and highlights the line with the error. When you poise the pointer over the icon, a tooltip appears describing the error.



- The message appears in the Annotation Browser, which also lists other errors in the file, including line numbers in a sortable column.



To view a line that has an error, click the row in the Annotation Browser.

**Note:** If the Annotation Browser is not visible, choose *Window – Assistants – Annotation Browser*. The Annotation Browser appears docked at the lower edge of the application window.

- The CIW lists warning messages.

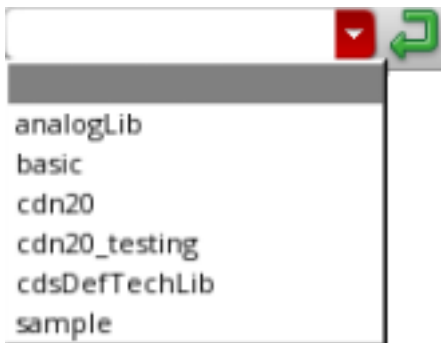


**Note:** Consistency checking is done at the syntax level. The technology file is not checked for completion.

## Loading a Technology File into a Technology Library

To make the technology file available in the database, load it into an existing Virtuoso library as follows:

1. Select a library from the list of accessible libraries in the *Technology File* combo box on the toolbar.



2. Click the adjacent *Load Technology File* button.

The technology file in the editor is loaded into the selected library. If the library already contains a technology database, it is fully replaced.

If you want to merge changes to a technology database or load the technology file into a new library, use the *Load* button in the Technology File Manager. For more information, see [Merging New ASCII Technology Data into an Existing Technology Library](#) on page 102.

## Environment Variables for the Techfile IDE

The `.cdsenv` file located at `tools/dfII/etc/tools/techfileIDE` includes the following environment variables, which support the Techfile IDE.

Environment Variable	Usage
----------------------	-------

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

---

<code>techfileIDE.parser</code> <code>checkMonotonicViolations</code>	<p>Determines the way in which spacing tables are checked for monotonic violations.</p> <p>This variable can have one of the following values:</p> <ul style="list-style-type: none"><li>■ <b>All:</b> Spacing tables are checked for monotonically increasing values. This is the default value.</li><li>■ <b>None:</b> Spacing tables are not checked for monotonic violations.</li><li>■ <b>twoWidths:</b> Only <code>twoWidths</code> spacing tables are checked for monotonic violations.</li><li>■ <b>widthLength:</b> Only <code>widthLength</code> spacing tables are checked for monotonic violations.</li></ul>
<code>techfileIDE.parser</code> <code>ignoreUndefTechParams</code>	<p>When set to <code>t</code>, ignores undefined <code>techParams</code>, which are useful when using incremental technology databases. The default value is <code>nil</code>.</p>

## Reusing Technology Data to Build a New Technology Library

This section discusses the following topics:

- Creating an ASCII technology file from a technology library
- Copying a technology library to use as a basis for creating a new technology library

### Creating an ASCII File from a Technology Library to Edit and Load Changes

To obtain a writable ASCII technology file, you can dump all or a portion of a technology database to an ASCII technology file. You can dump only the local technology database, not the effective technology database.

**Note:** A dumped ASCII technology file differs in the following ways from the original ASCII technology file that was compiled to create the technology library:

- It does not contain comments identified by semicolons ( ; ) that were in the original ASCII file, although it does contain comments made with the `comment` statement.
- It contains all of the data from originally included files and no `include` statements.
- It contains any changes made in virtual memory and saved to disk during design sessions.

There are the following reasons where the dump ASCII technology file is different from the original ASCII technology file:

1. The original technology file has system reserved layer:

In the IC610 release, every single `techLib` became an ITDB where `techLib` inherits the system reserved layers and other system specific information from `cdsDefTechLib` and technology file load ignores this information from ASCII technology file.

Therefore, when the technology file is dumped again following the standard ITDB convention, only the top level `techLib` which does not have the system reserved info is dumped back.

2. The original technology file has a `spacingTable` in which all the rule value for all the indexes of the table are not explicitly specified. Let's consider the example given below:



### Original technology file:

```
( minNumCut          "Via8"
  (( "width"   nil    nil ) 1 )
  (
    0.5        2
    1.0        4
  )
) ;spacingTables
```

### After Load/Dump Roundtrip:

```
( minNumCut          "Via8"
  (( "width"   nil    nil ) 1 )]
  (
    0.36       1
    0.5        2
    1.0        4
  )
) ;spacingTables
```

In this case the `minWidth` of `Via8` layer is `0.36` but the `minNumCut` rule does not explicitly specify how many cuts should be used when the width is `0.36`. Therefore, the technology file loader estimates this rule and puts it into the `techLib`, which then gets dumped in the ASCII technology file.

To create an ASCII technology file from a technology library, do the following:

1. From the Technology File Manager, choose *Dump*.

The Dump Technology File form appears.

**Dump Technology File**

Technology Library: **cdsDefTechLib**

☐ Select All ☐ Dump Empty Section Headers

- ☐ controls
- ☐ layerDefinitions
- ☐ layerRules
- ☐ siteDefs
- ☐ viaDefs
- ☐ packaging
- ☐ viaSpecs
- ☐ constraintGroups
- ☐ leRules
- ▼ ☐ devices
  - ☐ cdsVia
  - ☐ cdsGuardRing
  - ☐ User Defined Device
  - ☐ MPP
  - ☐ extractDevice
  - ☐ waveguide
  - ☐ ruleContact

ASCII Technology File:  **Browse...**

Open in Techfile IDE ☐

**OK** **Cancel** **Defaults** **Apply** **Help**

For a description of this form, see [Appendix A](#).

2. From the *Technology Library* cyclic field, select the library from which to dump the technology file.
3. Select the sections of the technology file that you want to dump.  
Select *Select All* to dump all sections.
4. In the *ASCII Technology File* field, type the name of the ASCII file that you want to create.



**Caution**

***It is safest to specify a new ASCII technology filename rather than overwrite an existing technology file. An ASCII file produced with the Dump command does not contain any of the comments or SKILL programs that your original file might contain. You might need the original ASCII technology file later.***

5. By default, the technology file opens in the configured text editor. Select the *Open in Techfile IDE* check box to open the file in the user-friendly Techfile IDE.
6. Click **OK**.

The ASCII file opens in a shell window. You can edit this file.

For more information about the syntax of the technology file, see the [Virtuoso Technology Data ASCII Files Reference](#).

After editing the ASCII file, you can compile and load it into the technology database in virtual memory, choosing either to merge it with the current technology database or to replace the current technology database with the new data. (For information on loading the technology file and merging its technology data with the technology data already in the technology library in virtual memory, see [“Merging New ASCII Technology Data into an Existing Technology Library”](#) on page 102. For information on loading the technology file and replacing the technology data in the technology library in virtual memory with the new technology file data, see [“Replacing Existing Technology Data in a Technology Library”](#) on page 104.) Then, if necessary, copy the changes in the ASCII file to your original annotated ASCII technology file.



**Important**

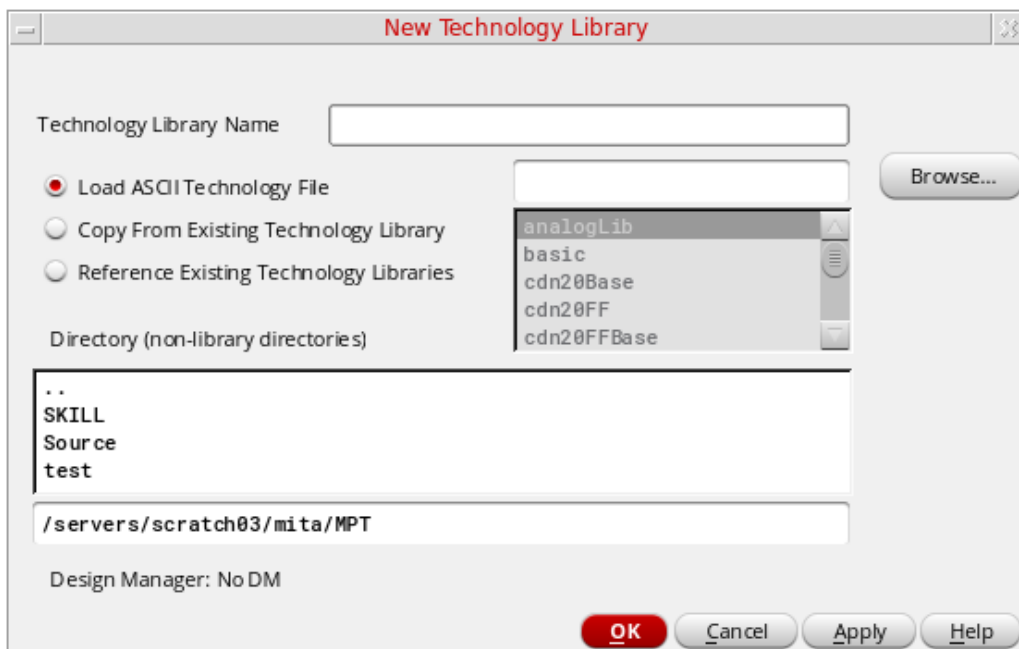
The dumper will dump all constructs that the current version of Virtuoso supports. Any newer technology information or constraints are ignored.

## Copying a Technology Library to Use As a Basis for Creating a New Technology Library

You can copy an entire technology library to use as a basis for creating a new technology library. To do so, perform the following steps:

1. From the Technology File Manager, choose *New*.

The New Technology Library form appears.



For a description of this form, see [Appendix A](#).

2. Turn on the *Copy From Existing Technology Library* radio button.
3. From the *Copy From Existing Technology Library* list box, choose the technology library to copy.
4. In the *Technology Library Name* field, type the path and name of the technology library to create. (You can navigate your directory structure in the *Directory* list box.)
5. Click *Apply* or *OK*.

The software creates a copy of the selected technology library in the specified directory. You can dump, edit, and recompile an ASCII technology file to alter the technology library or you can edit technology data in virtual memory.

## Loading Technology Data into Virtual Memory

This section explains how to load data into virtual memory in two ways. With the *Technology File – Load* command, you can merge technology data defined in an ASCII technology file with the technology data already in the technology database in virtual memory or you can replace the technology database in virtual memory with different technology data defined in a technology file.

There are occasions when you must load your technology database into the most recent version of the OpenAccess Data Model. In these situations, the technology database is uprevved to the later version when read into virtual memory.

During your editing session you can save the technology database to disk at any time. All of the edits are done in the later Data Model version and the output information reflects the new version number. If you do not save during the session, you are asked if you want to save at exit.

In the event that the same technology database is opened by multiple users at the same time, the first user to open it, with write privileges, will obtain a writable copy of the technology database in VM, and is given the option to save the technology for that session.

All other users are informed that their (up-revved in VM) copy of the technology database is not writable at that time even though they have write permission to the technology database.

For technology databases that do not have write permission, a warning is displayed.

### DataModel Version Considerations

For a technology database with an OpenAccess Data Model (DM) version less than DM3, the technology database is revised up to DM3 in virtual memory (VM) because the relationship between the referenced technology database `cdsDefTechLib` and the technology database is set when the technology database is open in VM.

This reference relationship between the technology database and the `cdsDefTechLib` is not retained after a save to disk. This means that the saved technology database does not contain DM3 data even though the reference relationship in VM is still active.

While working in VM in the DM3 version, a message appears in the CIW similar to the following:

```
\o INFO (TECH-150003): The technology database "tech1" has been automatically
\o updated from revision 223500(DM 0) to revision 226610(DM 3)
\o in virtual memory. It will be opened in 'a' mode for you to save it to disk.
```

Upon saving and querying the database using `cdsPrintOAFeatures`, it is confirmed that the DM version is not DM3:

```
cdsPrintOAFeatures -lib tech1  
Data model revision of tech tech1: DM0
```

## Merging New ASCII Technology Data into an Existing Technology Library

You can define new or edited technology data in an ASCII technology file and then merge that data with an existing technology library by compiling and loading the new technology file according to the following steps (refer to [“Replacing Existing Technology Data in a Technology Library”](#) on page 104 for information on replacing the technology library). Merging new technology data does not alter existing technology file data that is not specified in the ASCII technology file being merged into the technology library.

**Note:** When loading a technology file in the Merge mode, all constraints in the technology database are replaced with the constraints in the incoming technology file.

1. From the Technology File Manager, choose *Load*.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Technology File Data

---

The Load Technology File form appears.

Choose the technology library into which to merge data.

Type the name of the ASCII technology file to load.

Browse directories and files.

Choose the ASCII technology file sections to load.

Load Technology File

ASCII Technology File  Browse...

Classes ☐ Select All

<input type="checkbox"/> controls	<input type="checkbox"/> layerDefinitions
<input type="checkbox"/> layerRules	<input type="checkbox"/> siteDefs
<input type="checkbox"/> viaDefs	<input type="checkbox"/> packaging
<input type="checkbox"/> devices	<input type="checkbox"/> viaSpecs
<input type="checkbox"/> constraintGroups	<input type="checkbox"/> leRules

Technology Library

☒ Merge ☐ Replace

OK Cancel Defaults Apply Help

Choose *Merge*.

For a description of this form, see [Appendix A](#).

2. In the *ASCII Technology File* field, type the name of the ASCII technology file you want to compile and load.
3. Click the sections you want to compile and load from the ASCII technology file.  
To load all sections, click *Select All*.
4. From the *Technology Library* cyclic field, choose the technology library into which you want to compile and load the ASCII technology file data.

5. Click *Merge*.

**Note:** When you choose *Merge*, existing data specifications in the technology library that are order-dependent are replaced. New functions that are order-dependent and order-independent are appended to the existing data in the technology library in virtual memory.

6. Click *OK*.

The software compiles the technology file and loads it into virtual memory. If your ASCII file does not contain all of the sections you chose, a dialog box appears listing the missing sections. Click *OK* to continue or *Cancel* to quit.

For information about when each section is compiled and when the process is finished, look for messages in the DFII CIW and in the `techManager.log` file. For example:

```
Compiling class 'layerDefinitions' ....
Compiling class 'layerRules' ....
Compiling class 'constraintGroups' ....
Technology file '~/xyz.tf' loaded successfully.
```

## Replacing Existing Technology Data in a Technology Library

You can define new or edited technology data in an ASCII technology file and then replace an existing technology library by compiling and loading the new technology file according to the following steps (refer to “[Merging New ASCII Technology Data into an Existing Technology Library](#)” on page 102 for information on merging the technology data with the existing technology library). Loading in replace mode removes existing technology data that is not specified in the ASCII technology file being loaded to replace the technology library. Loading in replace mode replaces the entire database with the data specified in the ASCII technology file; any data not specified in the technology file but in the existing library is deleted from the library.

**Note:** `ConstraintGroups` are always loaded in replace mode in OpenAccess.

1. From the Technology File Manager, choose *Load*.

The Load Technology File form appears.

For a description of this form, see [Appendix A](#).

2. In the *ASCII Technology file* field, type the name of the ASCII technology file you want to compile and load.

3. Click the sections you want to compile and load from the ASCII technology file.

To load all sections, click *Select All*.



4. From the *Technology Library* cyclic field, choose the technology library into which you want to compile and load the ASCII technology file data.

5. Click *Replace*.

**Note:** When you choose *Replace*, the software replaces the entire technology library to reflect the technology data defined in the replacement technology file.

6. Click *OK*.

The software compiles the technology file and loads it into virtual memory. If your ASCII file does not contain all of the sections you chose, a dialog box appears listing the missing sections. Click *OK* to continue or *Cancel* to quit.

For information about when each section is compiled and when the process is finished, look for messages in the DFII CIW and in the `techManager.log` file. For example:

```
Compiling class 'layerDefinitions'....
Compiling class 'layerRules'....
Compiling class 'constraintGroups'....
Compiling class 'viaDefs'....
Technology file '~/design1.tf' loaded successfully.
```

## Discarding an Edited Technology Database from Virtual Memory

If you have loaded an edited ASCII file to create the technology library you have been using and have decided not to keep edits you have made during a design session, you can reload the original ASCII technology file from disk to recreate the original library in virtual memory.

1. From the Technology File Manager, choose *Discard*.

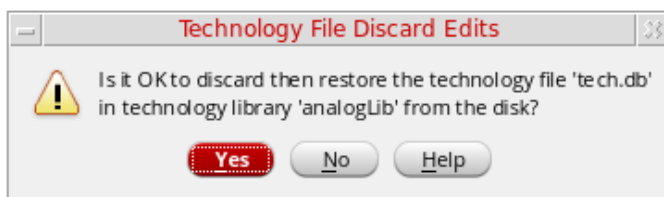
The Discard Edits To Technology File form appears.



For a description of this form, see [Appendix A](#).

2. In the *Technology Library* cyclic field, choose the library from which to discard your edits.
3. Click *OK*.

The Discard Edits dialog box appears, asking you to confirm the discard and reload the data saved on disk.



4. Click *Yes*.

The technology library on disk is loaded into virtual memory, deleting any changes you made since you last saved.

A message in the DFII CIW and in the `techManager.log` file indicates that the command was successful.

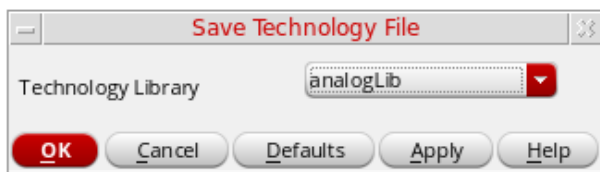
Technology file 'cellTechLib' was restored successfully.

## Saving a Technology Library Edited in Virtual Memory to Disk

To permanently save changes to a technology library edited in virtual memory, you must save the library to disk.

1. From the Technology File Manager, choose *Save*.

The Save Technology File form appears.



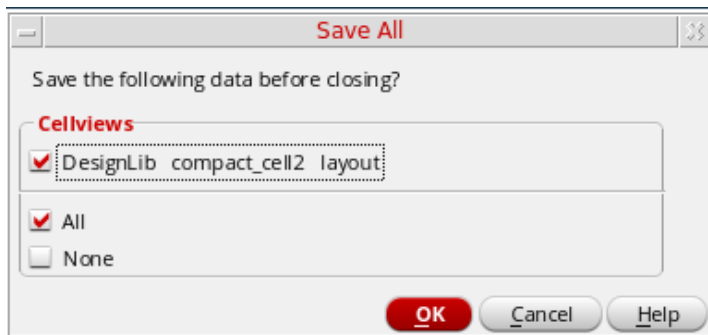
For a description of this form, see [Appendix A](#).

2. From the *Technology Library* cyclic field, choose the library to save.
3. Click *OK*.

A dialog box appears, asking you to confirm the save to disk. Click *Yes*.  
The technology library in virtual memory is saved to disk.

Message boxes appear if you change technology data and then try to do anything that will cause you to lose those changes if you do not save them. For example, when you quit the software or apply layer display changes. The message boxes give you the opportunity to save the changes or to proceed without saving.

For example, If you try to exit the software without saving your changes, the message box shown below is displayed:



- To save the changes, click *OK*.
- To discard the changes, select *None* and click *OK*.

# **Virtuoso Technology Data User Guide**

## Editing, Reusing, and Merging Technology File Data

---

---

## Editing, Reusing, and Merging Display Resources

---

This chapter discusses the following topics:

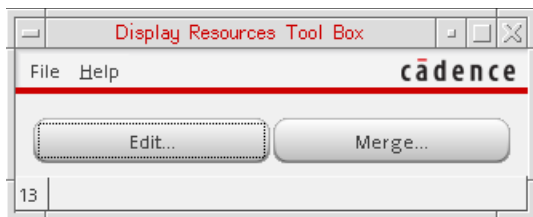
- Using Display Resource Manager
  - ❑ Opening Display Resources Tool Box
  - ❑ Using Display Resources Tool Box
  - ❑ Editing Display Resources
  - ❑ Merging Display Resource Files
- Using Display Resource Editor
  - ❑ Accessing Display Resource Data through Display Resource Editor
  - ❑ Deleting Display Resources
  - ❑ Loading a Display Resource File in Virtual Memory
  - ❑ Saving Display Resource Data to a File
  - ❑ Reloading Source Display Resource Files
  - ❑ Setting Selection Display Colors and Dynamic Highlights
- Editing Display Resource Data with SKILL Functions
- Editing a Display Resource File
- Testing a Display Resource File

## Using Display Resource Manager

Display Resource Manager lets you manipulate display resources available in virtual memory during a design session.

### Opening Display Resources Tool Box

From the CIW menu bar, choose *Tools – Display Resource Manager* to open the Display Resources Tool Box form.



### SKILL Function

`techManagerOpenDisplayToolBox()`

### Using Display Resources Tool Box

The Display Resources Tool Box form provides access to the following two commands:

- **Edit:** Opens the Display Resource Editor form. You can use this form to edit display resources loaded in virtual memory, save your changes to a display resource file, or load an existing display resource file to virtual memory. For more information, see [Editing Display Resources](#).
- **Merge:** Opens the Merge Display Resource Files form. You can use this form to merge multiple display resource files into a single display resource file. For more information, see [Merging Display Resource Files](#).

### Editing Display Resources

During a design session, you can edit display resources loaded in virtual memory, load an existing display resource file into virtual memory, or create a new display resource file from the data currently loaded in virtual memory.

To edit the display resources currently loaded in virtual memory, you can use the following methods:

- Use the Display Resource Editor form to work with display resources. For example, you can add and edit display packet definitions; add colors, stipple patterns, and line styles; edit colors, stipple patterns, and lines styles; and add display devices you want to use during a design session. For more information, see [Using Display Resource Editor](#).
- Use the *File – Load* command to merge a customized display resource file with the display resource data already loaded in virtual memory. For more information, see [Loading a Display Resource File in Virtual Memory](#).
- Use Cadence® SKILL® language functions to load a display resource file in virtual memory and edit the display resources. See [Editing Display Resource Data with SKILL Functions](#) for a summary of the SKILL functions available for manipulating display resources.

You can save to a new display resource file on disk either all the display resource data currently loaded in virtual memory or only the display resource data that you modified during a design session. For more information, see [Saving Display Resource Data to a File](#).

To reuse an existing display resource file to create a new one, create a copy of the existing file and edit it in a text editor. The sequence in which this new file is loaded and used depends on where you store it. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).

For information about creating a new display resource file from scratch, see [Display Resource File Development](#).

## Merging Display Resource Files

While working on a design, you can merge multiple display resource files into one file, and, if required, load this file into virtual memory. During the merge operation, if a resource is defined in more than one display resource file, its definition in the last merged file overrides any other definitions. Therefore, to obtain the results that you want, you must be aware of the resources defined in the various display resource files that you want to merge and the order in which these files should be merged.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

The following example illustrates how the merge operation overwrites display resource definitions that are defined differently in the display resource files being merged:

#### Files being merged, in order listed below

First file loaded for the merge.

```
firstlib/display.drf
( display  white  255  255  255 )
( display  silver 217  230  255 )
( display  cream  255  255  204 )
```

Second file merged. Its definition of *silver* overwrites the definition of *silver* in the first file.

```
secondlib/display.drf
( display  silver 220  233  252 )
( display  pink   255  191  242 )
( display  magenta 255   0   254 )
```

Third file merged. Its definition of *magenta* overwrites the definition of *magenta* in the second file.

```
thirdlib/display.drf
( display  magenta 255   0   255 )
```

#### Resultant merged file

```
mergedlib/display.drf
( display  white  255  255  255 )
( display  cream  255  255  204 )
( display  silver 220  233  252 )
( display  pink   255  191  242 )
( display  magenta 255   0   255 )
```

## Merging Multiple Display Resource Files

To merge multiple display resource files:

1. In the Display Resources Tool Box form, click *Merge*.
2. Click *OK* to close the message box that is displayed.



The Merge Display Resource Files form is displayed.

3. Identify the files that you want to merge and the order in which you want to merge them.

**Note:** The order of selection is important. If a display resource is defined in more than one file, its definition in the last merged file overwrites any other definition.

4. To select the files to merge, do one of the following for each file, starting with the file that you want to be merged last:
  - ☐ In the *From Library* list box, select a library name. The name of the display resource file stored in that library appears in the *Merge DRF files in sequence* list box, along with the location where it is stored.
  - ☐ In the *From File* field, specify a filename (you can click *Browse* to select the required file) and click *Add*. The filename appears in the *Merge DRF files in sequence* box.

**Note:** When you add a file, it is added to the bottom of the list in the *Merge DRF files in sequence* list box. The listed files are merged from the bottom up. Consequently, the file you add first is merged last and the file you add last is merged first.

5. In the *Destination DRF* field, specify the path and the name of the new file.

**Note:** The file must be named `display.drf`, and if you want to automatically load this file when you run the software, it must be saved to a location where the DFII initialization process can read it. You must also take into account any other `display.drf` files loaded at the time of initialization. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).

6. Ensure that the *Load Merged DRF* check box is selected (the default) if you want the file to be automatically loaded into virtual memory after the merge operation is complete.

If you do not want to immediately view the results of the merge operation, you can load the file later from the Display Resource Editor.

7. Click *OK* or *Apply*.

## Using Display Resource Editor

Display Resource Editor (DRE) lets you edit the display resource data loaded in virtual memory. Depending on your system setup, this data may be a combination of multiple display resource files. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).

In the Display Resources Tool Box form, click *Edit* to open the Display Resource Editor form.

# Virtuoso Technology Data User Guide

## Editing, Reusing, and Merging Display Resources

The Display Resource Editor form is displayed.

Load, Save, and reinitialize display resources; exit Display Resource Editor.

Choose *DRE Tool* from the *Help* menu.

Sort the LPP entries by typing layer, purpose, fill color, or outline color.

Select a *Fill Style* to view the fill and outline colors, stipples, and line styles associated with it.

Choose or create a new display device.

Choose the required *View Mode*: *All LPPs*, *All Packets*, *All Valid LPPs*, or *Active Palette Filtered LPP*.

Contents of this column depend on the value selected in the *Device* list.

View a list of all LPPs associated with a packet.

Create a new packet by specifying the packet name in the text box.

View the current and modified display packet attributes with the swatch preview.

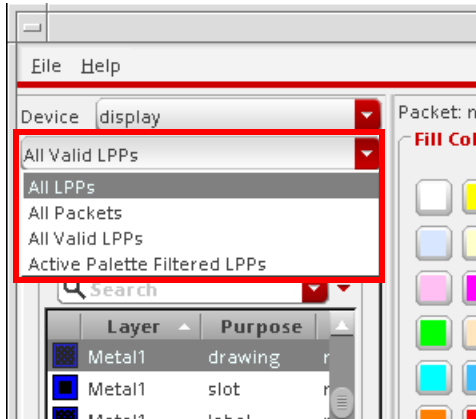
The screenshot shows the 'Display Resource Editor' window. At the top, there are menu buttons: 'File', 'Help', and 'cadence'. Below the menu is a 'Device' dropdown set to 'display', and a 'Packet' dropdown set to 'm1'. A 'Select LPP' dialog is open, showing a table with columns 'Layer' and 'Purpose'. The table lists various LPPs for 'Metal1' and 'Metal2' layers, with purposes like 'drawing', 'slot', 'label', 'fill', 'boundary', 'pin', and 'net'. To the right of the dialog are five columns of color and style swatches: 'Fill Color', 'Outline Color', 'Stipple', 'Line Style', and 'Fill Style'. At the bottom, there are buttons for 'Custom...', 'Create New Packet', and 'Modify Current Packet'. A status bar at the bottom says 'Packet: m1 (Click the Associations tab to view all LPPs in gpd090 library that use this packet)'.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

The options shown in the figure below determine the view mode:



- **All LPPs:** Displays all LPPs defined in the selected technology file, including system-reserved LPPs. If a Palette is open, then the LPP selected in the Palette is automatically selected in the DRE form.
- **All Packets:** Displays all display packets defined for the selected device.
- **All Valid LPPs:** Displays all valid LPPs for the selected technology file. If a Palette is open, then the LPP selected in the Palette is automatically selected in the DRE form.
- **Active Palette Filtered LPPs:** Displays all LPPs displayed in the Palette, in the same sequence. However, the LPP selected in the Palette is not selected automatically in the DRE form.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

When the view mode is set to *All Packets* or *Active Palette Filtered LPPs*, the name of the active technology library can be determined from the message that appears on the status bar, as shown below.



## Accessing Display Resource Data through Display Resource Editor

With Display Resource Editor, You can edit most, but not all, of the display resource data loaded in virtual memory. The following is a list of search and editing functions that you can perform with Display Resource Editor:

- Finding a Display Packet or a Layer-Purpose Pair by Name
- Editing a Display Packet Definition
- Adding a Display Packet Definition
- Editing a Display Packet Definition
- Adding Color
- Editing Color
- Adding a Stipple Pattern
- Editing a Stipple Pattern

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

- Adding a Line Style
- Editing a Line Style
- Adding a Display Device

The following is a list of functions that you cannot perform with Display Resource Editor:

- Deleting Colors, Stipples, Line Styles, and Devices
- Deleting a Display Packet

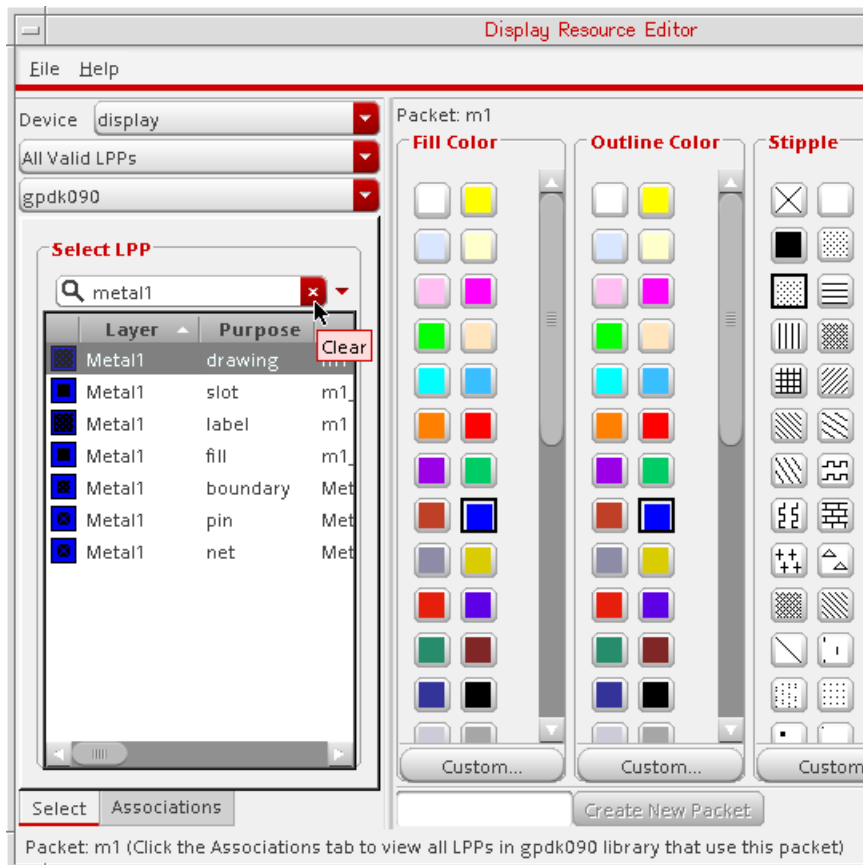
To perform these functions, you must edit the `display.drf` file in a text editor or edit the display resource data in virtual memory by using SKILL functions.

After you have edited the display resource data, you can perform the following functions by using Display Resource Editor:

- Loading a Display Resource File in Virtual Memory
- Saving Display Resource Data to a File
- Reloading Source Display Resource Files

## Finding a Display Packet or a Layer-Purpose Pair by Name

To find a display packet with a specific name, type the name in the *Search* text box. As you type each character, the list below is automatically updated to display the packets or layer-purpose pairs that match the search string. Click the *Clear* button to cancel the filter.



## Adding a Display Packet Definition

To add a new display packet definition:

1. In the Display Resource Editor form, select the required view mode as *All Packets*.
2. Select an existing display packet, and, in the right pane, select the *Fill Color*, *Outline Color*, *Stipple*, and *Line Style* attributes that you want to set for the new display packet.
3. In the text box to the left of the *Create New Packet* button, type a name for the new packet and click *Create New Packet*.

**Note:** The *Create New Packet* button is enabled when you type a name in the text box.

The new display packet is created and listed in the *Packet Name* list in the left pane. It is also added to the display resource data loaded in virtual memory.

To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Editing a Display Packet Definition

Display packets control how layers are displayed and plotted in designs. A display packet can be common to many layers. When you change the definition of a display packet, all layers that use that display packet are automatically updated.

To change a display packet definition loaded in virtual memory, do the following:

1. In the Display Resource Editor form, select the required view mode as *All Packets*.
2. Scroll down the list of packets or use the *Search* field to locate the required display packet.

The *Fill Color*, *Outline Color*, *Stipple*, and *Line Style* attributes currently set for the selected packet are highlighted in the right pane.

**Note:** You can also set the view mode to *All LPPs*, *All Valid LPPs*, or *Active Palette Filtered LPPs* to select an LPP, if required. This displays in the right pane the attributes of the display packet used by the selected LPP. Several LPPs can share the same display packet. Therefore, when you change the display packet definition for one LPP, the appearance of all other LPPs that use the display packet is updated. To view a list of LPPs that use the same display packet, click the *Associations* tab.

3. Select the required *Fill Color*, *Outline Color*, *Stipple*, and *Line Style* attributes for the display packet.
4. Click *Modify Current Packet*.

The display packet definition loaded in virtual memory is updated. The icon for the display packet in the Display Resource Editor is also updated with the newly set attributes. If a cellview is open, it is automatically updated.

To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

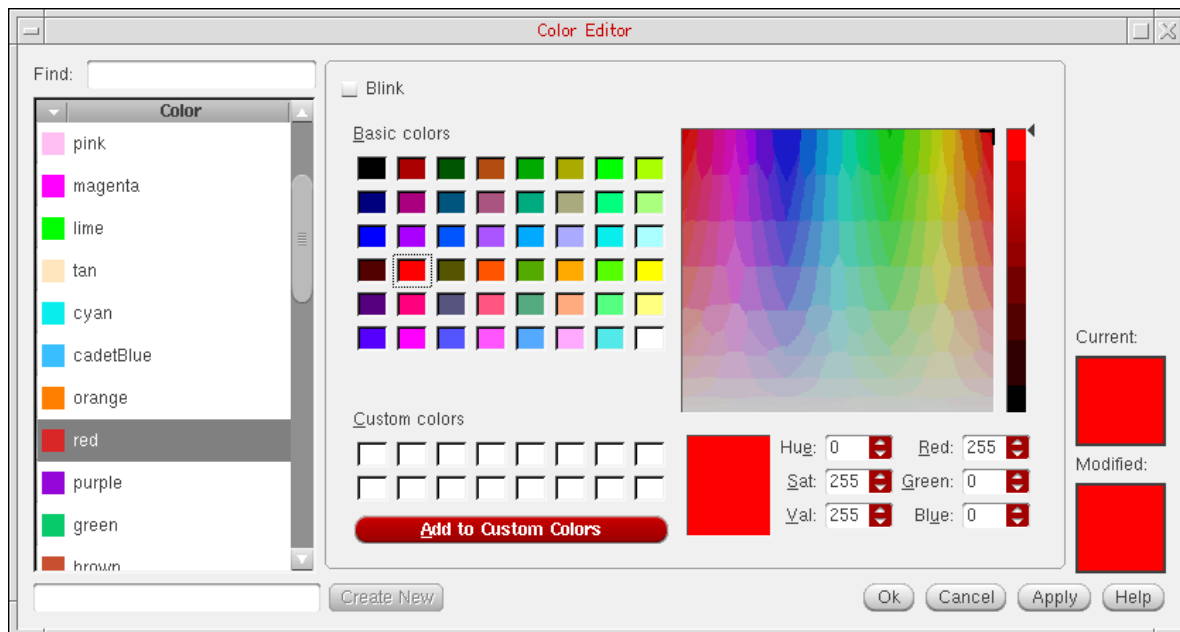


## Adding Color

To add a new color to the color set, do the following:

1. In the Display Resource Editor form, click *Custom* in the *Fill Color* or *Outline Color* column.

The Color Editor form is displayed, with the current color loaded in the swatch area.



If a color does not appear in the swatch area, see [Virtuoso Design Environment User Guide](#) for information about mapping colors.

2. In the text box to the left of the *Create New* button, type a name for the new color and click *Create New*. The new color is listed in the *Color* list above.

**Note:** The *Create New* button is enabled when you type a name in the text box.

3. Drag the slider up or down, or type in the *Red*, *Green*, and *Blue* fields the required values, to create a new color.

The color corresponding to the specified values is displayed in the *Modified* swatch area.

4. Select the *Blink* check box to make the color a blinking color.
5. Click *Apply* or *Ok*.

The new color is listed in the Display Resource Editor form. It is also added to the display resource data loaded in virtual memory.

To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Editing Color

Fill and outline colors come from the same set of colors. To edit an existing color, do the following:

1. In the Display Resource Editor form, select the required color and click *Custom* in the *Fill Color* or *Outline Color* column.

The Color Editor form is displayed, with the selected color loaded in the swatch area. If a color does not appear in the swatch area, see [Virtuoso Design Environment User Guide](#) for information about mapping colors.

2. Drag the sliders up or down, or type in the *Red*, *Green*, and *Blue* fields the required values, to edit the color. The color corresponding to the specified values is displayed in the *Modified* swatch area.
3. Select the *Blink* check box to make the color a blinking color.
4. Click *Apply* or *Ok*.

The new color is listed in the Display Resource Editor form. It is also added to the display resource data loaded in virtual memory.

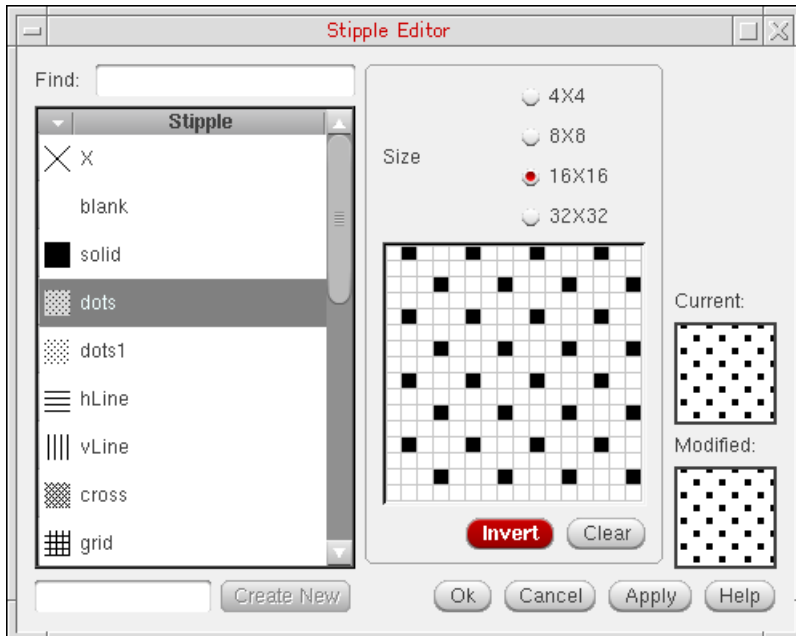
If a cellview is open, the layer information in the Palette is automatically updated. To update the cellview, choose *View – Redraw*. To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Adding a Stipple Pattern

To add a new stipple, do the following:

1. In the Display Resource Editor form, click *Custom* in the *Stipple* column.

The Stipple Editor form is displayed.



2. In the text box to the left of the *Create New* button, type a name for the new stipple and click *Create New*. The new stipple is listed in the *Stipple* list above.

**Note:** The *Create New* button is enabled when you type a name in the text box.

3. From the grid resolutions listed under *Size*, select the required value.
4. Click the pixels in the grid until the *Modified* swatch area shows the required stipple pattern.
  - ☐ To clear all the pixels in the grid, click *Clear*.
  - ☐ To invert the pixel selection, click *Invert*.

5. Click *Apply* or *Ok*.

The new stipple is listed in the Display Resource Editor form. It is also added to the display resource data loaded in virtual memory.

To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Editing a Stipple Pattern

To edit a stipple, do the following:

1. In the Display Resource Editor form, select the required stipple and click *Custom* in the *Stipple* column.

The Stipple Editor form is displayed, with the selected stipple loaded in the swatch area.

**Note:** You cannot modify a reserved stipple.

2. Select a grid resolution from those listed under *Size*, if required, and click the pixels in the grid until the *Modified* swatch area shows the required stipple pattern.

- ☐ To clear all pixels in the grid, click *Clear*.
- ☐ To invert the pixel selection, click *Invert*.
- ☐ To start again with the original pattern, click the stipple name in the *Stipple* list. You can also select another stipple to edit, if required.

3. Click *Apply* or *Ok*.

The new stipple pattern is listed in the Display Resource Editor form. It is also added to the display resource data loaded in virtual memory.

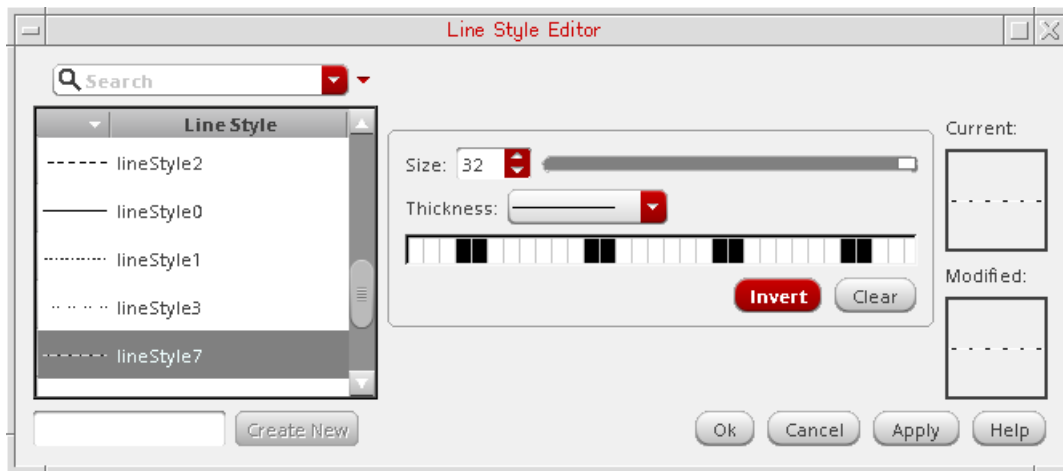
If a cellview is open, the layer information in the Palette is automatically updated. To update the cellview, choose *View – Redraw*. To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Adding a Line Style

To add a new line style, do the following:

1. In the Display Resource Editor form, click *Custom* in the Line Style column.

The Line Style Editor form is displayed.



2. In the text box to the left of the *Create New* button, type a name for the new line style and click *Create New*. The new line style is listed in the *Line Style* list above.

**Note:** The *Create New* button is enabled when you type a name in the text box.

3. In the *Size* field, specify the required grid resolution.
4. In the *Thickness* field, specify the required line thickness.
5. Click the pixels in the grid until the *Modified* swatch area shows the required line style pattern.
  - ☐ To clear all the pixels in the grid, click *Clear*.
  - ☐ To invert the pixel selection, click *Invert*.
6. Click *Apply* or *Ok*.

The new line style is listed in the Display Resource Editor form. It is also added to the display resource data loaded in virtual memory.

To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Editing a Line Style

To edit a line style, do the following:

1. In the Display Resource Editor form, select the required line style and click *Custom* in the *Line Style* column.

The Line Style Editor form is displayed, with the selected line style loaded in the swatch area.

**Note:** You cannot modify a reserved line style.

2. In the *Size* field, specify a grid resolution, and in the *Thickness* field, specify the line thickness, if required, and click the pixels in the grid until the *Modified* swatch area shows the required line style pattern.

- ☐ To clear all pixels in the grid, click *Clear*.
- ☐ To invert the pixel selection, click *Invert*.
- ☐ To start again with the original pattern, click the line style name in the *Line Style* list. You can also select another line style to edit, if required.

3. Click *Apply* or *Ok*.

The new line style is listed in the Display Resource Editor form. It is also added to the display resource data loaded in virtual memory.

If a cellview is open, the layer information in the Palette is automatically updated. To update the cellview, choose *View – Redraw*. To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Adding a Display Device

A display device is a piece of hardware, such as a monitor or a plotter, which you use to display, plot, or print designs. You can define a display packet to appear differently on different display devices. The table below lists commonly used display devices.

For information about setting up plotters, refer to the [Plotter Configuration User Guide](#).

---

Device Name	Type
display	Color monitors
hp6	Hewlett-Packard 6-carousel pen plotters
hp8	Hewlett-Packard 8-carousel pen plotters
psb	PostScript black-and-white plotters
versatecb	Versatec and CalComp black-and-white plotters
versatecc	Versatec and CalComp color plotters
XBlackWhite	Black-and-white X Window System monitors

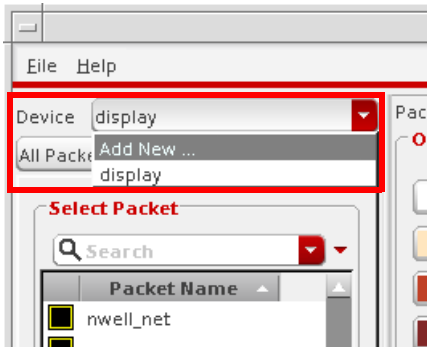
## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

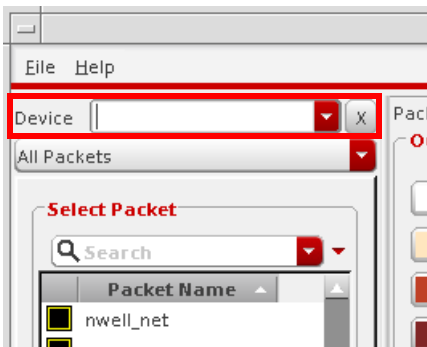
Device Name	Type
X4PlaneColor	4-plane color X Window System monitors

To add a display device, do the following:

1. From the *Device* list, choose *Add New*.



2. In the text box that is displayed, type the new device name.



3. Press Enter.

The new display device is added to the *Device* list and set as the active device.

To save your changes to disk, use the Save As form. For more information, see [Saving Display Resource Data to a File](#).

## Deleting Display Resources

You must always exercise caution while deleting display resources because the display resource that you delete could be in use in multiple technology libraries and designs.

## **Deleting Colors, Stipples, Line Styles, and Devices**

Before you delete a display resource such as a color, stipple, or line style, consider the following:

- Display packets are defined by a set of display resources. If you delete a display resource, the display packets using that display resource become invalid, as do all the layers that use those display packets. To preserve affected display packets, you must update the display packet definitions to include an existing display resource.
- Because the system merges several source display resource files to create the display resource data you use, the color, stipple, or line style resource that you want to delete may exist in more than one source display resource file. It might even be defined differently in different source files. To completely delete the display resource, you must delete it from all source display resource files.

You can create display resource files in your home directory for personal use. If the data in these files overwrites the default data, you may need to delete these personal display resource files and reinitialize the display resource data.

### ***Finding Display Packets Affected by Deleting a Display Resource***

To determine which display packets are affected by deleting a display resource, do the following:

1. Save all the display resource data currently loaded in virtual memory. For more information, see [Saving Display Resource Data to a File](#).
2. In a text editor, open the file that you just created and search for the name of the display resource that you want to delete.

**Note:** Line styles and stipples can have the same name.

The following figures show samples of display resource definitions:

#### **□ A color definition**

```
; ( DisplayName      ColorsName      Red      Green      Blue )  
( display          yellow          255      255      0      )
```



## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

#### ❑ A stipple definition

```
; ( Displayname StippleName Bitmap
  ( display      dots          ( ( 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 )
                                ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 )
                                ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 )
                                ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 )
                                ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 )
                                ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 )
                                ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 )
                                ( 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 )
                                ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 )
                                ( 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 )
                                ( 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 )
                                ) )
```

#### ❑ A line style definition

```
; ( DisplayName LineStyle Size Pattern
  ( display      wide      1      ( 1 1 0 1 0 1 1 1 0 1 ) )
```

3. Search for the display resource name again to locate a display packet definition that uses the display resource.

#### A display packet definition

```
; ( DisplayName PacketName Stipple LineStyle Fill Outline )
  ( display      grayslashwide slash wide gray gray )
```

4. Note down the name of the display packet.
5. Repeat steps 6, 7, and 8 until you have located all the display resource definitions that use the display resources that you want to delete.
6. Exit the text editor.

### ***Deleting a Display Resource from a Display Resource File***

Do the following for each source display resource file that you want to edit.

**Note:** If you make a change to a local display resource file or to a display resource file in your design management system, you need to inform the designers who use that design hierarchy.

1. In a text editor, open the display resource file.
2. Search for the name of the display resource to delete.
3. Delete the definition of the display resource.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

For example, to delete the `wide` line style, delete the definition starting from the opening parenthesis through the closing parenthesis.

```
; ( DisplayName      LineStyle      Size      Pattern      )  
  ( display          wide           1          (1 1 0 1 0 1 1 1 0 1 ) )
```

4. For each display packet that uses the display resource just deleted, replace the deleted display resource with an existing display resource.

For example, you may want to replace the `wide` line style with the `solid` line style, as shown below:

```
; ( DisplayName PacketName      Stipple LineStyle Fill Outline)  
; ( display      grayslashwide slash   wide      gray gray)  
  ( display      grayslashwide slash   solid      gray gray)
```

For information about locating all such display packet names, see [Finding Display Packets Affected by Deleting a Display Resource](#).

5. Save the file and exit the text editor.
6. Repeat this procedure for all display resource files from which you want to delete a display resource.
7. Broadcast your changes to all users of your design hierarchy.

The following is a sample broadcast message:

Notice of technology change:

The line style "wide" has been deleted from the design hierarchy. Display packets using this line style have been updated to use the "solid" line style. (Display packets determine how your layers appear on screen and in your plots.)

If you use a display resource file in your home directory or work area, please update your file.

If you need assistance call John Doe at 3340.

**Note:** You can also use SKILL functions to delete display resources in virtual memory. For more information, see [Editing Display Resource Data with SKILL Functions](#).

## Deleting a Display Packet

Technology files reference display packet names. Therefore, before you delete a display packet, ensure that your technology files do not reference that display packet. As a precaution, consider leaving unused display packets in your display resource file for later use.

- If you are sure about deleting a display packet, open the source display resource file in a text editor and delete the required display packet definition.

To refresh the display resource data during a design session, rerun the software, or, in the Display Resource Editor, choose *File – Reinitialize*. For more information about reinitializing display resource data, see [Reloading Source Display Resource Files](#).

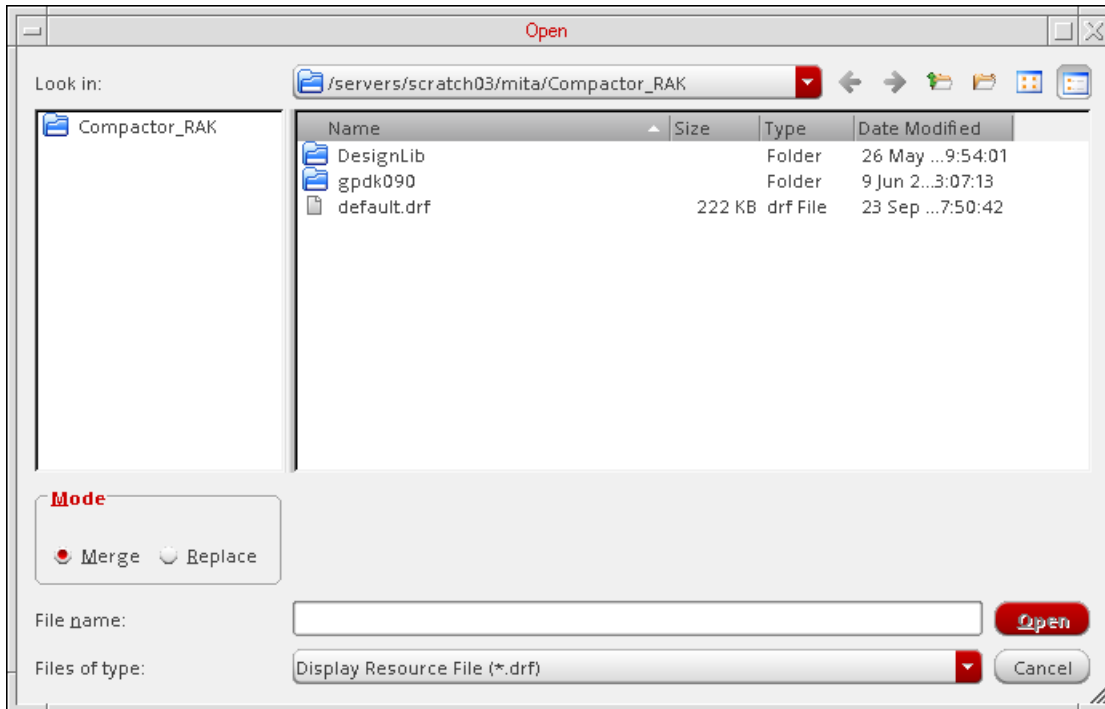
**Note:** You can also use SKILL functions to delete display resources in virtual memory. For more information, see [Editing Display Resource Data with SKILL Functions](#).

## Loading a Display Resource File in Virtual Memory

When you load a display resource file, the data in the file is merged with the data that is already loaded in virtual memory. To merge additional display resource data with the display resource data in virtual memory, perform the following steps:

1. In the Display Resource Editor form, choose *File – Load*.

The Open form is displayed.



2. Select a file to load.
3. In the *Mode* group box, choose *Merge*.

To replace the display resource data in virtual memory, choose *Replace*.

4. Click *Open*.

The selected file is loaded into virtual memory and Display Resource Editor is updated with the display resource data from the selected file. If a cellview is open, choose *View – Redraw* to update the cellview.

However, if you have any unsaved changes, you are prompted to save your changes first. For more information about how to save the changes made to the display resource data in virtual memory, see [Saving Display Resource Data to a File](#).

## Saving Display Resource Data to a File

You can save the display resource data loaded in virtual memory to a text file that you can use in future sessions. Additionally, you can save either all the display resource data or only the changes that you made during a design session. To use the new display resource file in a design session in future, you can place the file in your home directory or in the directory

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

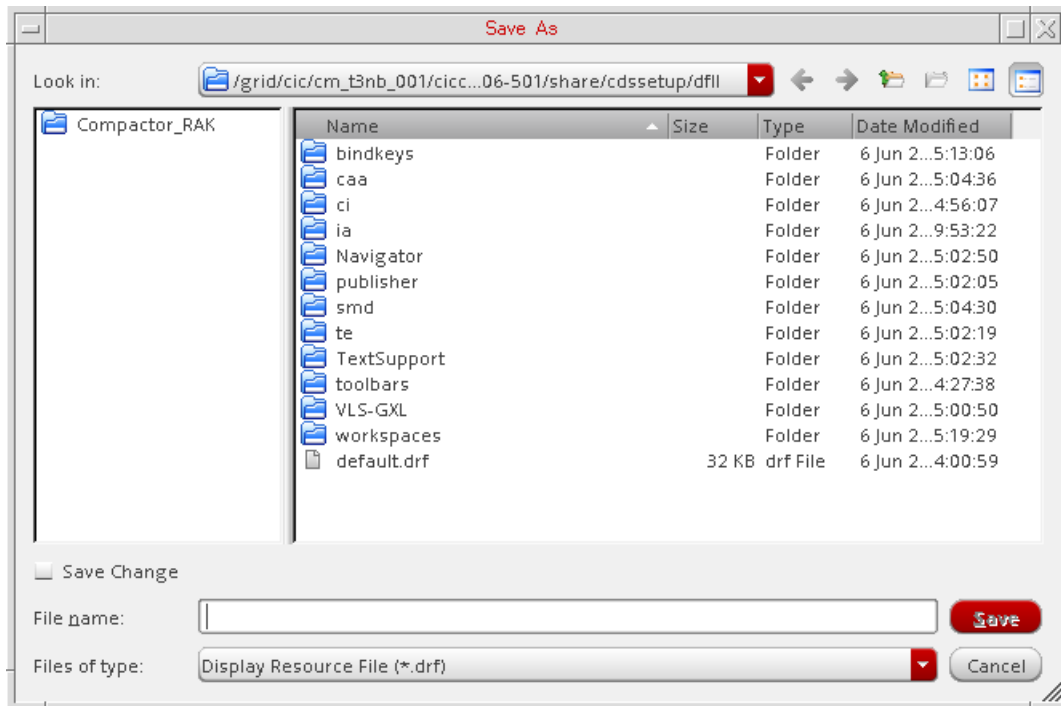
from which you run the software. By placing the customized display resource data in your home directory, you have control over how your designs appear on screen and in plots.

**Note:** If you save all the display resource data available in virtual memory, the saved file contains all the display resource data loaded at the start of the software session *plus* the changes made during the session. In this case, you may want to edit the saved file to include only the data you need before you place the file in your home directory. For more information, see [Editing a Display Resource File](#).

To save to a file the display resource data currently loaded in virtual memory, do the following:

1. In the Display Resource Editor form, choose *File – Save*.

The Save As form is displayed.



2. Select the directory to which you want to save the file, and in the *File name* text box, specify a name for the file.
3. Ensure that the *Save Change* check box is deselected if you want to save all the display resource data currently loaded in virtual memory. To save only the display resource data modified during the current design session, select the *Save Change* check box.
4. Click *Save*.

A display resource file with the specified name is created.

## Virtuoso Technology Data User Guide

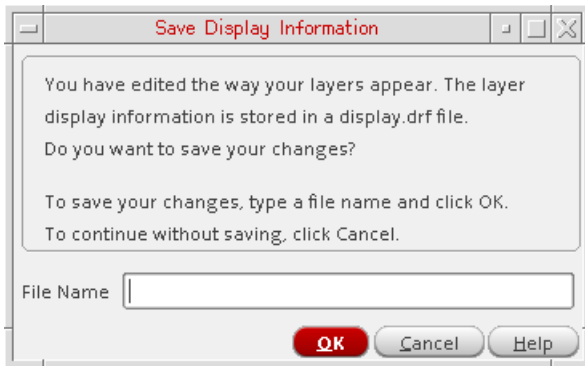
### Editing, Reusing, and Merging Display Resources

---

You can now do one of the following:

- Edit the saved file. For more information, see [Editing a Display Resource File](#).
- Add the file to a location from which the file is automatically loaded every time you run the software. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).
- Load the file manually. For more information, see [Merging Display Resource Files](#).
- Discard the edits made during the current session by reloading the original display resource data. For more information, see [Reloading Source Display Resource Files](#).

If you did not save your changes while exiting Display Resource Editor (or if you made changes using SKILL functions), the Save Display Information form, shown below, is displayed when you exit Virtuoso. Type a filename in the *File Name* field and click *OK* to save your changes, or click *Cancel* to discard your changes.



If you do not want this form to be displayed when you exit Virtuoso, set the `drmSuppressSaveDialogBox` to `t` in the CIW, as shown below. As a result, any changes that you have made are automatically discarded.

```
envSetVal("layout" "drmSuppressSaveDialogBox" 'boolean t)
```

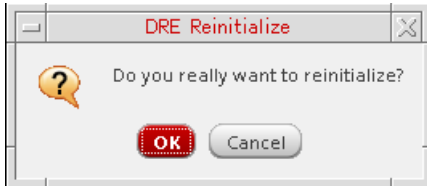
The default value of this environment variable is `nil`. You can also include it in the `.cdsenv` file in your home directory or in the `.cdsinit` file.

## Reloading Source Display Resource Files

If you modified the display resource data in your current design session and now want to reload the data you started with, do the following:

1. In the Display Resource Editor, choose *File – Reinitialize*.

2. In the dialog box that is displayed, click *OK*.



The source display resource files on disk are loaded into virtual memory in sequence. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).

## Setting Selection Display Colors and Dynamic Highlights

You can specify different colors to distinguish between selected and unselected objects on the same layer. You can also choose to specify the `dynamicHighlight` display packet to dynamically highlight objects in the design.

### Setting Selection Display Colors

If the `useSelectDeviceForSelectionDisplay` environment variable is set to `t`, the `select` and `hierSelect` devices, if defined, determine the display attributes of the selected objects. The `hierSelect` device lets you use different display packets for hierarchical objects such as instances and vias. The default value of this environment variable is `nil`.

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

If the environment variable is set to `t`, but the `select` and `hierSelect` devices are not found, the default `display` device is used. You can add the `select` and `hierSelect` devices either by using the Display Resource Editor form or by including them in the `drDefineDisplay` section of the `display.drf` file.

### Setting Dynamic Highlights

If you set the `useDynamicHighlightPacket` environment variable to `t`, a packet named `dynamicHighlight`, which determines how dynamic highlights around objects appear, is automatically created when you first dynamically highlight an object. However, this packet is not saved automatically in `display.drf`.

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

If you save this packet, it is available in Display Resource Editor even when the environment variable is set to `nil`.

The `useDynamicHighlightPacket` environment variable is also defined in the "layout" partition. As a result, the appearance of the dynamic highlights in layout windows can be different from that in other graphic applications.

```
envSetVal("layout" "useDynamicHighlightPacket" 'boolean t)
```

The default value of this environment variable is `nil`.



## Editing Display Resource Data with SKILL Functions

The following table lists the SKILL functions that return information and operate on display resource data:

SKILL Function	Description
<u>drDeleteDisplay</u>	Deletes the specified display device from virtual memory.
<u>drDeleteColor</u>	Deletes the definition of the specified color for the specified display device from virtual memory.
<u>drDeleteLineStyle</u>	Deletes the specified line style from virtual memory.
<u>drDeletePacket</u>	Deletes the definition of the specified display packet for the specified display device from virtual memory.
<u>drDeleteStipple</u>	Deletes the definition of the specified stipple for the specified display device from virtual memory.
<u>drDumpDrf</u>	Dumps to a file either all the display resource data in virtual memory or only the changes made in virtual memory.
<u>drFindPacket</u>	Reads virtual memory and returns a list of attributes of the specified display packet for the specified display device.
<u>drGetColor</u>	Reads virtual memory and returns the display device, the color name, and the red, green, blue, and blink values for the specified color.
<u>drGetDisplay</u>	Reads virtual memory and returns the display device identifier of the specified display device name.
<u>drGetDisplayIdList</u>	Reads virtual memory and returns a complete list of display device identifiers.
<u>drGetDisplayName</u>	Reads virtual memory and returns the display device name of the specified display device identifier.
<u>drGetDisplayNameList</u>	Reads virtual memory and returns a complete list of display device names.
<u>drGetLineStyle</u>	Reads virtual memory and returns the display device name and the line style name, thickness, and pattern for the specified line style.

## Virtuoso Technology Data User Guide

### Editing, Reusing, and Merging Display Resources

---

SKILL Function	Description
<u>drGetLineStyleIndexByName</u>	Reads virtual memory and returns the line style index number for the specified line style for the specified display device.
<u>drGetPacket</u>	Reads virtual memory and returns the definition of the specified display packet for the specified display device.
<u>drGetPacketList</u>	Reads virtual memory and returns a list of display packets defined for the specified display device.
<u>drGetPacketAlias</u>	Reads virtual memory and returns a list of display packets aliased to the specified display packet.
<u>drGetPacketFillStyle</u>	Reads virtual memory and returns the fill style number of the specified display packet for the specified display device.
<u>drGetStipple</u>	Reads virtual memory and returns the display device name and the stipple name, width, height, and pattern for the specified stipple.
<u>drGetStippleIndexByName</u>	Reads virtual memory and returns the stipple index number for the specified stipple name.
<u>drLoadDrf</u>	Loads the display resource file (usually named <code>display.drf</code> ) from any location.
<u>drSetPacket</u>	Updates in virtual memory the value of the specified display packet for the specified display device.

For more information about these SKILL functions, see *Virtuoso Technology Data SKILL Reference*.

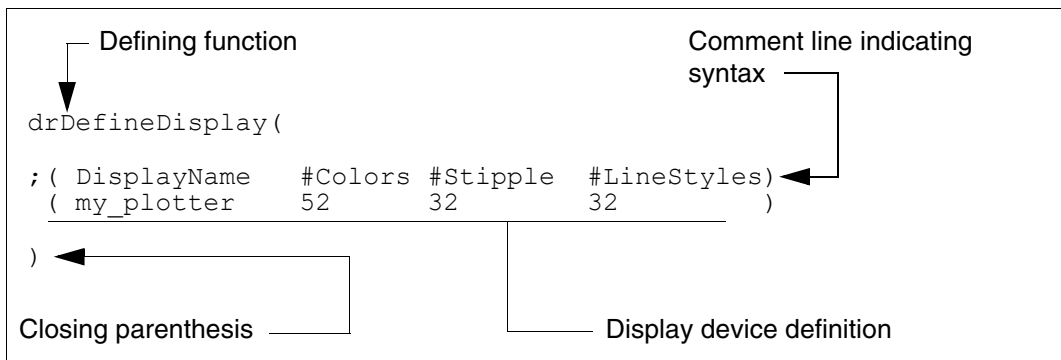
## Editing a Display Resource File

You can edit in a text editor a display resource file, which you had saved earlier, to retain only the customized display resources. For this, do the following:

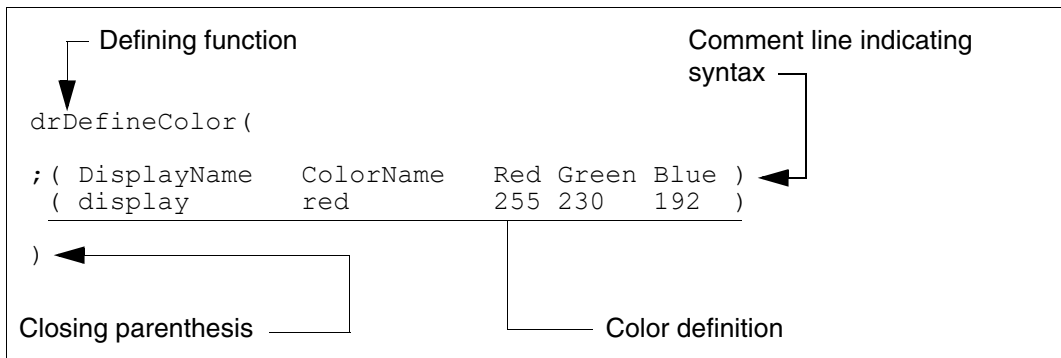
1. Create a backup copy of the file.
2. In a text editor, open the original copy of the file.
3. Insert comments in the file to keep a record of the reason for customization and the details of the customized data, as shown below:  

```
; Created 1/24/15 for Emerald project.  
; Changed red color to look tan.
```
4. Delete all lines except those defining the display resources that you had customized. You can choose to preserve the original display device, color, stipple, line style, and display packet definitions as comments.

### ☐ Customizing a display device



### ☐ Customizing a color

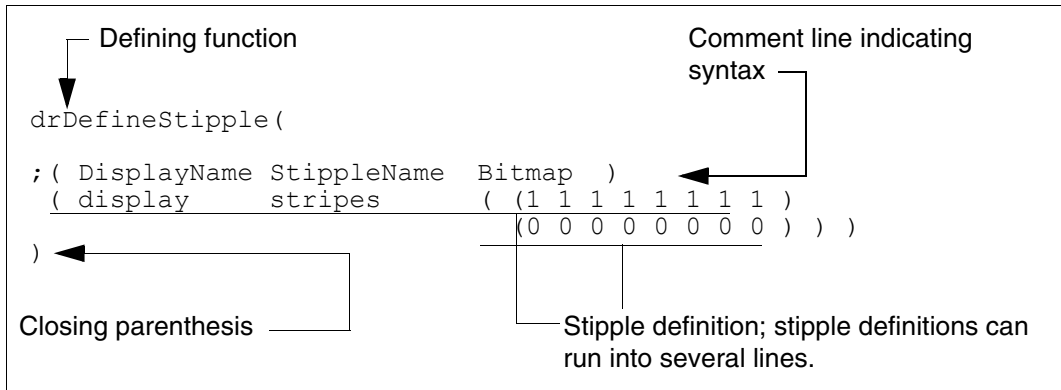


## Virtuoso Technology Data User Guide

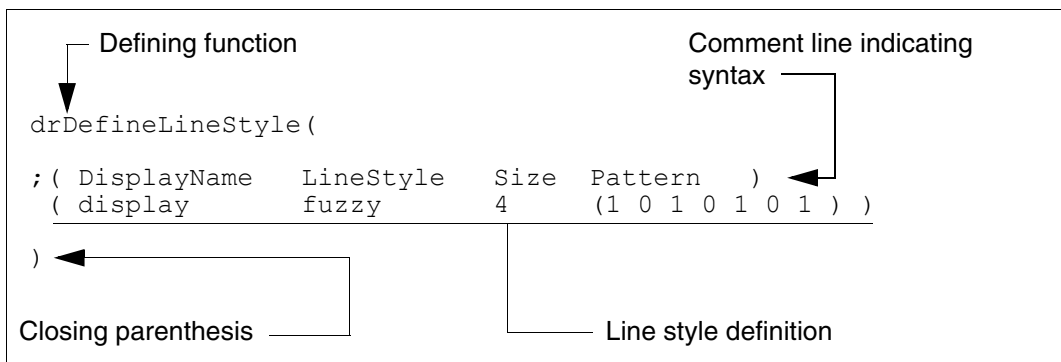
### Editing, Reusing, and Merging Display Resources

---

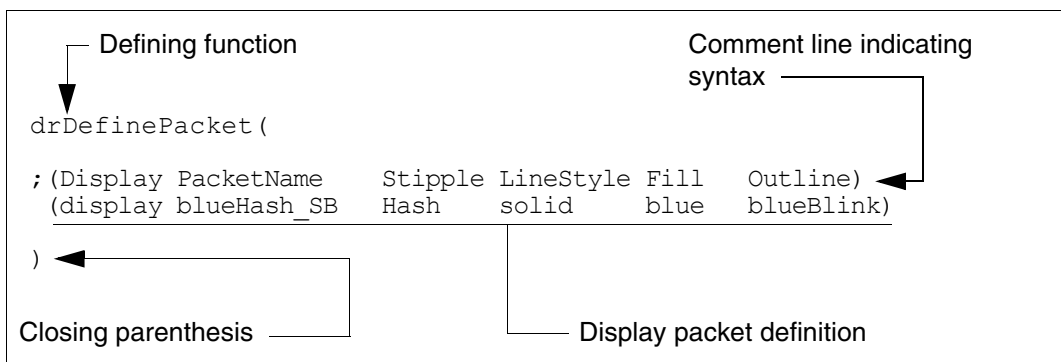
#### ❑ Customizing a stipple



#### ❑ Customizing a line style



#### ❑ Customizing a display packet



5. Save the changes and exit the text editor.

6. Test the customized resource data file. For more information, see [Testing a Display Resource File](#).

7. After verifying that the customized display resource definitions work as expected, do one of the following:

- ❑ Paste the customized display resource definitions into a source display resource file. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).
- ❑ Place the file in your home directory or work area directory.

As a result, your changes are automatically loaded every time you run the software.

## Testing a Display Resource File

To test a display resource data file, do the following:

1. In the Display Resource Editor form, choose *File – Reinitialize*.
2. In the dialog box that is displayed, click *OK*.

The source display resource files on disk are reloaded into virtual memory in sequence. For more information, see [How Cadence Design Software Handles Multiple Display Resource Files](#).

3. Choose *File – Load*.

The Open form is displayed.

4. In the *File name* field, type the name of the file that you edited.
5. Click *Open*.

If there are errors in your file, messages are displayed in CIW and in the `techManager.log` file.

6. Open a cellview and look for objects that use the customized display resources.

# **Virtuoso Technology Data User Guide**

## Editing, Reusing, and Merging Display Resources

---

---

## Defining and Installing Devices

---

This chapter discusses the following:

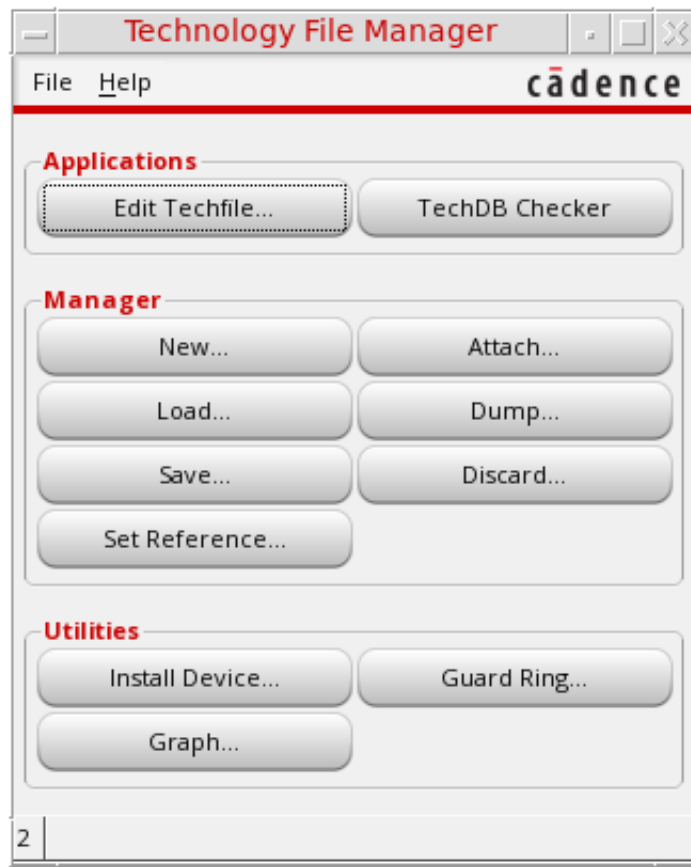
- [Bringing up the Install Device Form](#) on page 144
- [Guard Rings](#) on page 146
  - [About Guard Rings](#) on page 146
  - [About the Install Device Form for Guard Rings](#) on page 150
  - [Defining and Installing a New Guard Ring](#) on page 158
  - [Defining a New Guard Ring from an Existing Guard Ring](#) on page 165
  - [Modifying an Existing Guard Ring](#) on page 167
  - [Deleting a Guard Ring](#) on page 168

## Bringing up the Install Device Form

To bring up the Install Device form to begin defining a device,

1. From the CIW, choose *Tools – Technology File Manager*.

The Technology File Manager form is displayed.



2. From the Technology File Manager form, choose *Install Device*.



## Virtuoso Technology Data User Guide

### Defining and Installing Devices

---

The Install Device form appears.

The 'Install Device' dialog box is shown with the following fields and controls:

- Technology Library:** A dropdown menu set to 'lib'.
- Device Type:** A dropdown menu set to 'guardRing'.
- Name:** A text field containing 'M1\_M2'.
- Buttons:** 'View Device Configuration' and 'Delete'.
- Options:** 'Layers' (checked), 'Rule' (unchecked), 'Stretch Handles' (unchecked), and 'Parameter Defaults' (unchecked).
- Layers List:**
  - Diffusion:** A color swatch, a text field with 'Metal1', and a dropdown menu set to 'drw'.
  - Contact:** A color swatch, a text field with 'Via1', and a dropdown menu set to 'drw'.
  - Metal:** A color swatch, a text field with 'Metal2', and a dropdown menu set to 'drw'.
- Buttons:** 'Add Implant Layer'.
- Footer Buttons:** 'OK', 'Cancel', 'Apply', and 'Help'.

## Guard Rings

This section covers the following topics:

[About Guard Rings](#) on page 146

[About the Install Device Form for Guard Rings](#) on page 150

[Defining and Installing a New Guard Ring](#) on page 158

[Defining a New Guard Ring from an Existing Guard Ring](#) on page 165

[Modifying an Existing Guard Ring](#) on page 167

[Deleting a Guard Ring](#) on page 168

### About Guard Rings

Guard rings are a special type of ROD multipart path that are used to encircle one or more objects, e.g. devices or device chains. They are intended to be either p-diffusion or n-diffusion.

**Note:** You can find detailed information about multipart paths in Chapter 10 of the Virtuoso Layout Editor User Guide entitled “[Creating and Editing Multipart Paths](#)”. However, this level of detail is not necessary in order to successfully install or create guard rings.

Guard rings are displayed in the Install Device form consist of the following parts:

- Diffusion layer

This layer is the master path. It is an ordinary path with flush or offset ends that must enclose all other parts of the guard ring, except the implant layer when one is required. It is recommended that you choose a p-diffusion or n-diffusion, or another type of diffusion, for the *Diffusion* layer field. A master path extending outside of an offset subpath, or which is wide compared to the objects which it will enclose, will require a larger Enclosed by value on the Create Guard Ring form.

- Metal layer

This layer is an offset subpath. The layer specified for the *Metal* layer cyclic field encloses the vias (set of subrectangles), which are centered within the *Metal* layer. You can specify a positive or negative offset. The offset is measured from the centerline of the *Diffusion* layer to the centerline of the *Metal* layer. The *Metal* layer is entirely enclosed by the *Diffusion* layer.

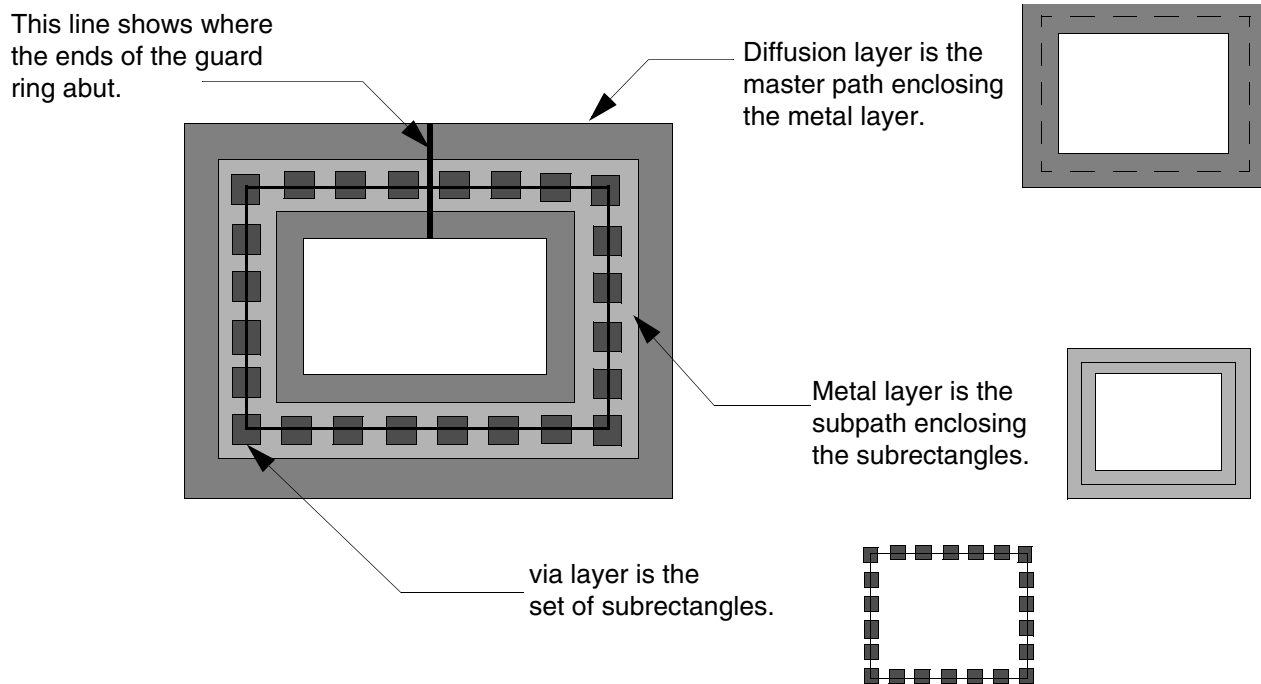
■ **Contact layer**

This layer is the set of subrectangles, which are vias. The layer specified for the *Contact* layer cyclic field is entirely enclosed by the *Metal* and the *Diffusion* layer; the vias are centered within the *Metal* layer. It is recommended that you choose a via layer for the *Contact* layer field, with its function defined as `cut` or `li`.

■ **Implant layer**

This layer is the second, optional, offset subpath. You can define only one implant layer for a guard ring. When specified, the layer chosen for the *Implant1* layer cyclic field must entirely enclose the *Diffusion* layer. It is recommended that you choose an implant layer for the *Implant1* layer field, with its function defined as `nplus` or `pplus`.

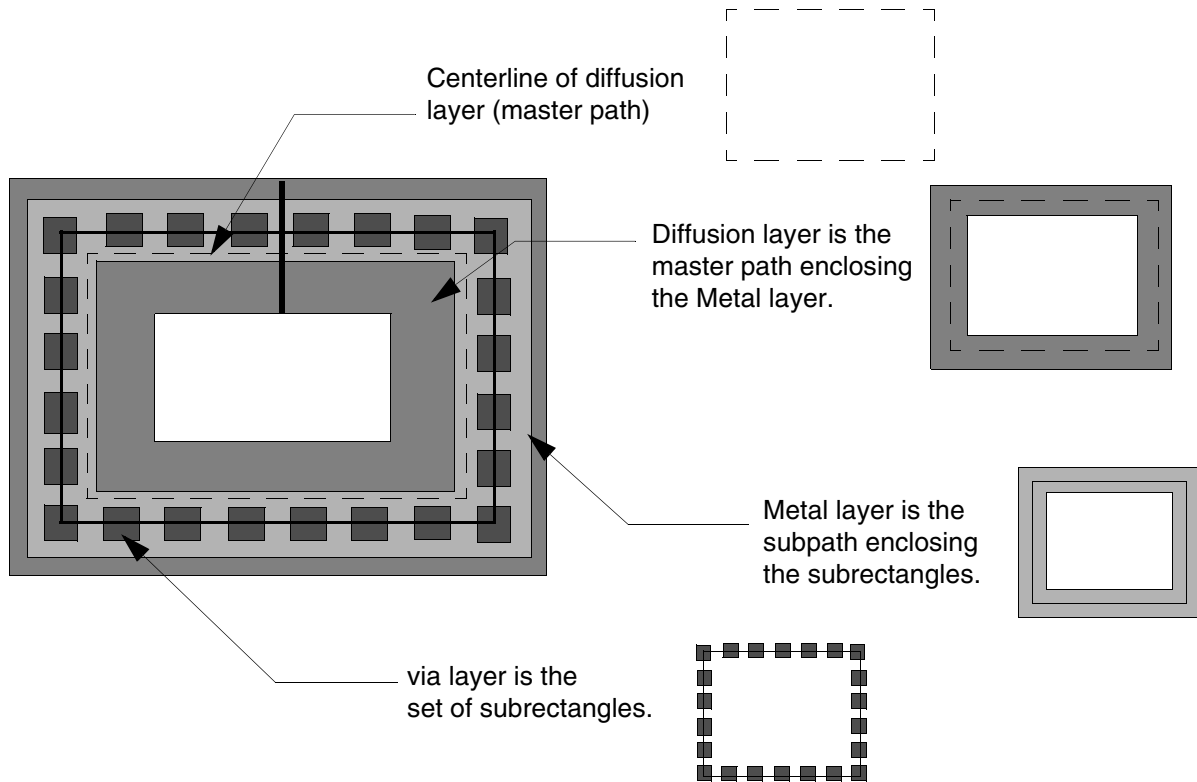
For example, the guard ring shown below consists of a master path on a diffusion layer, a subpath on a metal layer, and a set of subrectangles on a via layer. The centerlines of all three are coincident; the diffusion layer encloses the metal layer, which encloses the vias.



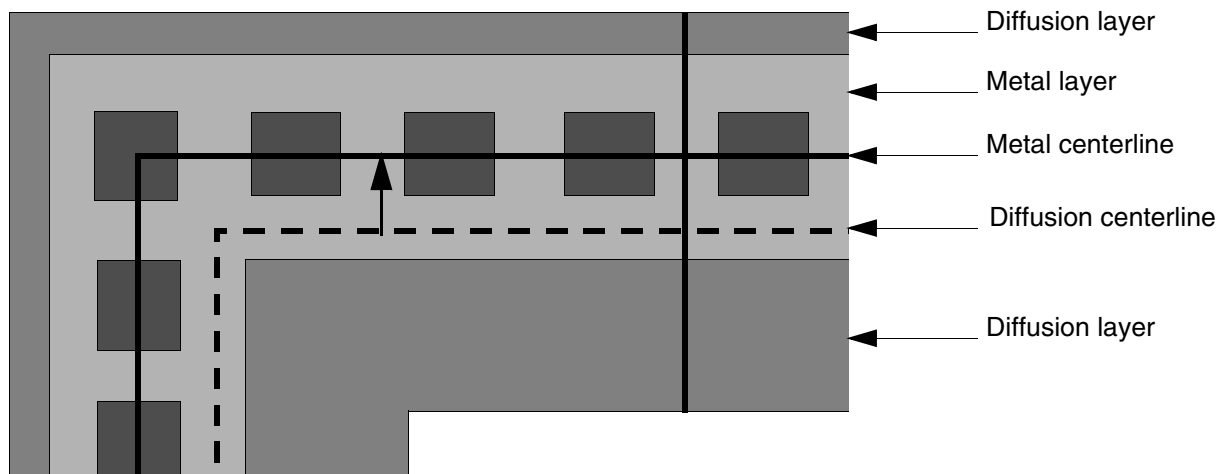
## Virtuoso Technology Data User Guide

### Defining and Installing Devices

An example of a guard ring with the metal layer (and therefore, also the vias) offset from the diffusion layer by a positive number is shown below.



The offset is measured from the centerline of the diffusion layer (which is behind the metal layer) to the centerline of the metal layer.

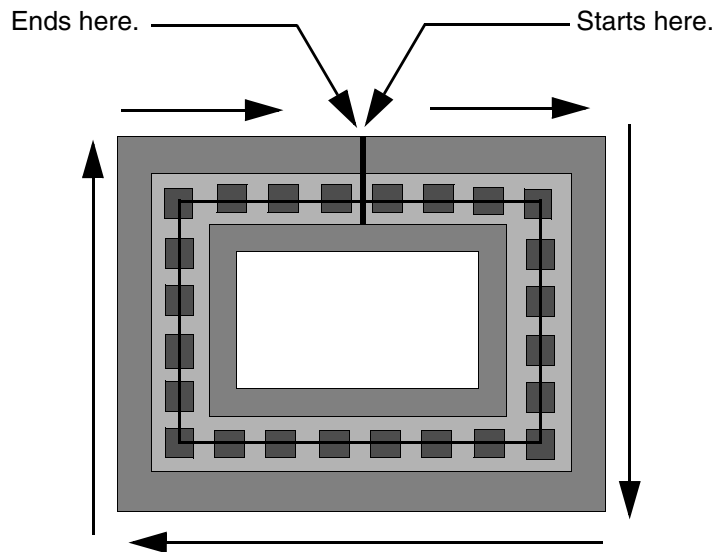


## Virtuoso Technology Data User Guide

### Defining and Installing Devices

---

The system creates guard rings in a clockwise direction, starting at the center of the longest side for polygons, top or left longest side for rectangles, and center of the left side for squares.



Guard rings are a limited, special type of multipart path (MPP.) The technology file associated with the library you are using might contain templates for guard rings and other types of MPPs. When you select *guardRing* for *Device Type* on the Install Device form, the names of MPP templates defined in your technology file that match guard ring specifications (a master path with truncate or extend ends, one set of subrectangles, and one or two offset subpaths, on any layers) appear in the *Name* field.

**Note:** Guard ring templates that include path extensions might be included in the list of guard rings in the *Name* field. The system lets you use these guard rings, but recalculates the extensions.

## About the Install Device Form for Guard Rings

The screenshot shows the 'Install Device' dialog box. At the top, the title bar says 'Install Device'. Below it, there are several fields and buttons. The 'Technology Library' field is set to 'lib'. To its right is a 'View Device Configuration' button. Below 'Technology Library' is the 'Device Type' field, set to 'guardRing'. To its right is the 'Name' field, set to 'M1\_M2', and a 'Delete' button. Below these fields are four checkboxes: 'Layers' (checked), 'Rule' (unchecked), 'Stretch Handles' (unchecked), and 'Parameter Defaults' (unchecked). Below the checkboxes are three rows of material settings. The first row is 'Diffusion' with a black square icon, 'Metal1' text, 'drw' text, and a red dropdown arrow. The second row is 'Contact' with a gray square icon, 'Via1' text, 'drw' text, and a red dropdown arrow. The third row is 'Metal' with a dark gray square icon, 'Metal2' text, 'drw' text, and a red dropdown arrow. Below these rows is an 'Add Implant Layer' button. At the bottom of the dialog are four buttons: 'OK' (red), 'Cancel', 'Apply', and 'Help'.

**Technology Library** sets the library containing guard ring templates you want to view, or in which you want to install a new or modified guard ring template. You can view guard ring templates whether a library is editable or not; when a library is not editable, the *OK* and *Apply* buttons are grayed out. To define a new guard ring or modify an existing guard ring, select an editable library (the *OK* and *Apply* buttons are active).

**View Device Configuration** lets you display a small window showing the guard ring configuration. This button is active when all required rules are entered in the form or defined in the technology file. You can click this button to display or refresh the image of the current guard ring configuration.

**Device Type** lets you select the kind of device you want to view, update, or install; for guard rings, you would select *guardRing*. When you select *guardRing*, the form changes to display the fields appropriate for guard rings.

**Name** lets you see the names of the guard ring templates defined in your technology file. These guard rings consist of: one master path with flush or offset ends, one or two offset subpaths, and one set of subrectangles.

This form also lets you define a new guard ring template by typing a unique name in the *Name* field and entering or changing the values for the other fields as desired. When you

define a new guard ring and click *Apply* or *OK* to install it, the system saves the new template to the technology file in virtual memory.

**Delete** lets you remove the guard ring template from your technology file from virtual memory. You will be asked whether you want to save your technology file on disk when you click *OK* or *Cancel*.

**Layers** The *Layers* button is turned on by default when the Install Device form first appears. The lower portion of the form shows the layers defined for the selected device.

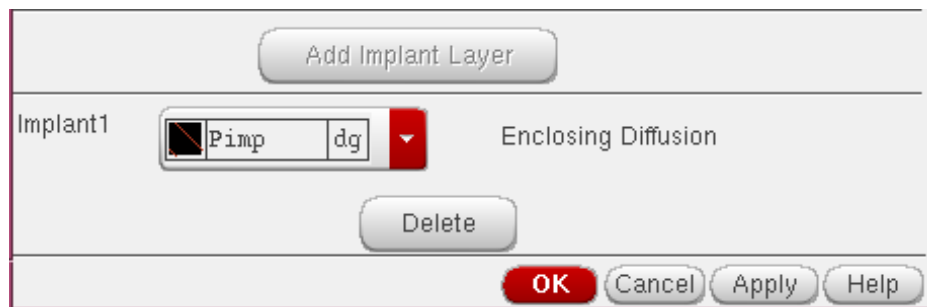
For guard rings, the layer fields are labeled *Diffusion*, *Contact*, and *Metal*. Each of three fields is associated with a specific part of the guard ring, as follows:

**Diffusion** lets you see or choose the layer for the diffusion layer (master path).

**Contact** lets you see or choose the layer for the via (set of subrectangles).

**Metal** lets you see or choose the layer for the offset subpath that surrounds the set of subrectangles.

**Add Implant Layer** displays a new section at the bottom of the Install Device form that allows you to see or define an implant layer to enclose the diffusion. Defining an implant layer is optional. When you click *Add Implant Layer*, the software displays the following section at the end of the Install Device form:



**Implant1** identifies the layer as an implant layer; this cyclic field lets you see or choose the implant layer to enclose the diffusion. You can only define one implant layer for a guard ring.

When you are defining a new guard ring, you might need to define an implant layer around the diffusion, depending on the material you chose for the diffusion (master path). For example, if your technology file does not contain n-diffusion or p-diffusion layers, you will need an implant layer.

**Enclosing Diffusion** tells you that the implant layer encloses the diffusion layer.

**Delete** lets you delete the implant layer for the current guard ring.

## Virtuoso Technology Data User Guide

### Defining and Installing Devices

**Rules** displays the Rules Browser window and redraws the lower part of the Install Device form to show the process design rules for the layers you specified with the *Layers* option. For example, the *Rules* section might look like this:

	Width	Length	TechfileDefault
Via1 Dimensions	0.1400	0.1400	✓
Via1 Spacing		0.1500	✓
Metal2 Offset		0.0	
Metal1 Width		0.1500	
Metal2 Width		0.1500	
Pimp Width		0.2400	

The top of the *Rules* section lets you see, enter, or change the values for *Dimensions* and *Spacing*. When *Techfile Default* is on, the values match the defaults in the technology file; when *Techfile Default* is off, the values do not match the defaults in the technology file. When the values are blank, there are no default values defined in the technology file.

**layerName Dimensions** specifies the width and length of vias on the layer specified for *Contact* in the *Layers* section. In the example above, this rule states that the size of vias is 0.1400 user units wide and 0.1400 user units long. *Techfile Default* indicates that these values match the default `minWidth` values in the technology file. You can change the via width and length to less than the technology file default value, and the form will not reset to the technology file value.

**layerName Spacing** specifies the space between vias on the layer specified for *Via*. In the example above, this rule states that the spacing between vias is 0.1500 user units. *Techfile Default* is on to indicate that this value matches the default `minSpacing` value in the technology file.



**Techfile Default** indicates whether the values currently displayed for the *Dimensions* and *Spacing* rules match the default values in your technology file. When the *Techfile Default* field is on, the values match.

When you enter or change to a value that does not match the default value in the technology file, and click anywhere else in the Install Device form, *Techfile Default* turns off to indicate that the value does not match the default value in the technology file. To restore the default values from the technology file for the *Dimensions* or *Spacing* fields, turn on *Techfile Default*.

The center of the *Rules* section displays the names of *Diffusion* or *Metal* enclosing via rules, if they are not defined in the technology file. The values are blank; you must enter a value for all missing rules.

**missingRule** is the name of a required enclosure rule that does not have a default value defined in the technology file. In the example above, the *Oxide Enclosing Via2* and *Metal1 Enclosing Via2* constraint fields are blank, meaning that there are no rules defined in the technology file for the layers enclosing Via2.

**Note:** After you enter values and click *OK* or *Apply* to save to the technology file in virtual memory, the *missingRule* fields are no longer displayed in the *Rules* section.

The bottom of the *Rules* section lets you see, enter, or change the values for *Offset* and *Width* fields.

For the *Width* fields, the system provides default values representing the minimum values required, calculated based on the via and enclosure layers. If you change values or enter new values and click *OK* or *Apply*, it verifies that the values of these fields are large enough to conform to the rules for guard rings: *Metal* layer encloses *Via* layer, *Diffusion* layer encloses *Metal* layer, and if there is an implant layer, *Implant1* layer encloses *Diffusion* layer.

**layerName Offset** specifies the distance between the centerline of the *Metal* layer (offset subpath) from the centerline of the *Diffusion* layer (master path). You can specify a positive or negative offset. The default is zero, meaning that the centerlines are coincident.

**layerName Width** specifies the width of the *Diffusion* layer (master path) specified in the *Layers* section.

**layerName Width** specifies the width of the *Metal* layer (offset subpath) specified in the *Layers* section.

**layerName Width** specifies the width of the *Implant1* layer (offset subpath), when an *Implant1* layer is specified in the *Layers* section.

### ***Adding or Modifying Rules: System Verification***

The system performs the following verification for the *Rules* section:

- When you click the *View Device Configuration* button, the system verifies that values have been entered for all blank fields. If a value is blank, the system displays a dialog box, stating that the rule must be specified; for example:

("diff" "drawing") Over ("cont" "drawing") Enclosure Rule must be specified.

- When you type a new or different value into the input field for the *Offset* or *Width* rules in the bottom part of the *Rules* section, and either click in a different *Rules* input field or click *Apply* or *OK* (or *Cancel* after having clicked *Apply*), the system compares the values for the required rules, and verifies that:

- The value for every rule is a multiple of the manufacturing grid in user units.
- The width of the *Metal* layer surrounding the set of subrectangles on the *Contact* layer is equal to or greater than the width of the set of subrectangles, plus two times the metal-to-via enclosure rule from the technology file:

$$\text{Metal layer width} \geq \text{via layer width} + (\text{metal-to-via enclosure rule} \times 2)$$

- The width of the *Diffusion* layer surrounding the *Metal* layer is equal to or greater than the width of the *Metal* layer, plus two times the diffusion-to-via enclosure rule from the technology file:

$$\text{Diffusion layer width} \geq \text{via layer width} + (\text{diffusion-to-via enclosure rule} \times 2)$$

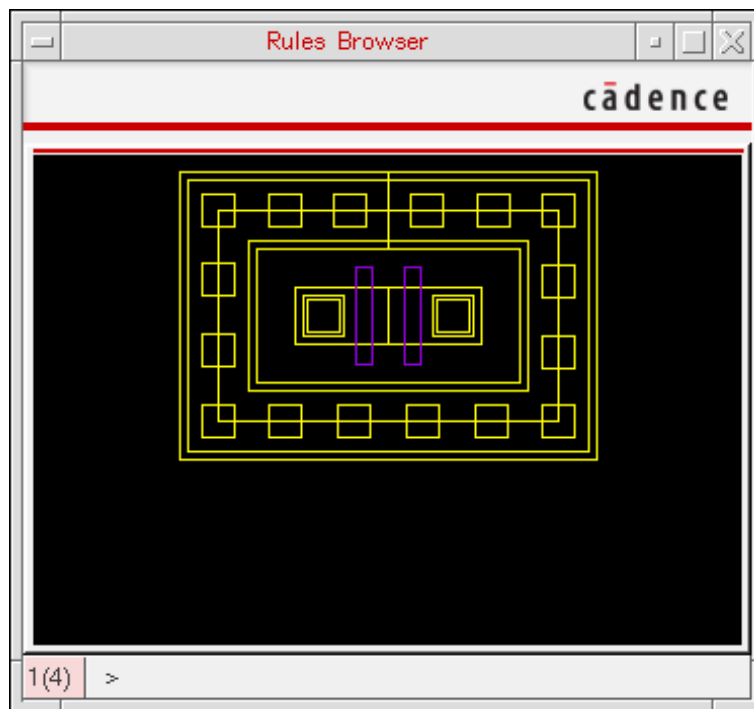
- The width of the *Diffusion* layer is greater than or equal to the width of the *Metal* layer.
- If there is an *Implant1* layer, the width of the *Implant1* layer is greater than or equal to the width of the *Diffusion* layer.

If the values do not pass the verification, the system adjusts the widths of the layers and replaces the numbers in the input fields so that they satisfy the rules, as described above.

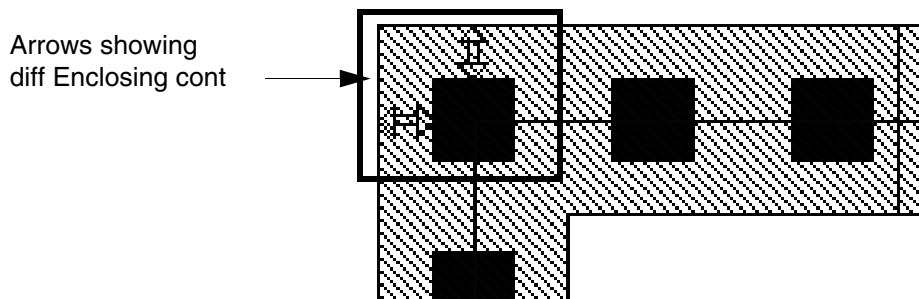
### Rules Browser Window

The Rules Browser window lets you see how design rules are applied graphically, using a double-ended arrow. The rules Browser is not an accurate picture of the current guard ring. For an accurate picture, see the [View Device Configuration Window](#) on page 156.

If you specified `diff`, `cont`, and `metal1` for layers, then the picture in the Rules Browser window might look like as shown below:



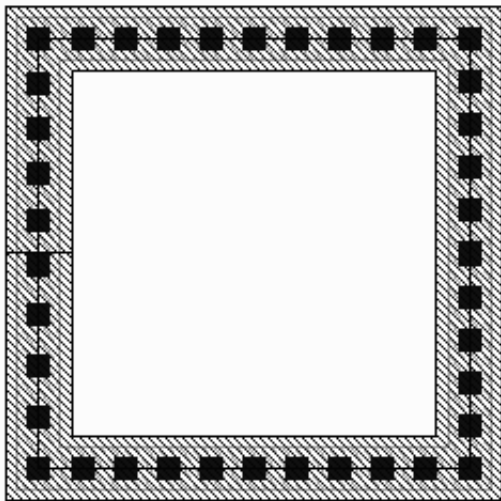
When you click in a design rule input field, the system shows you how the design rule is being measured by displaying double-ended arrows in the Rules Browser. For example, for *diff Enclosing cont*, the arrows look like this:



#### ***View Device Configuration Window***

The View Device Configuration button lets you display a small window showing the guard ring configuration. When any rule information is missing, clicking on this button displays a dialog box saying that you must specify the missing rule. When all required information is present, clicking on this button displays or refreshes the image of the guard ring configuration.

In the example of the guard ring with `diff`, `cont`, and `metal1` specified for layers, the View Device Configuration window might look like this:

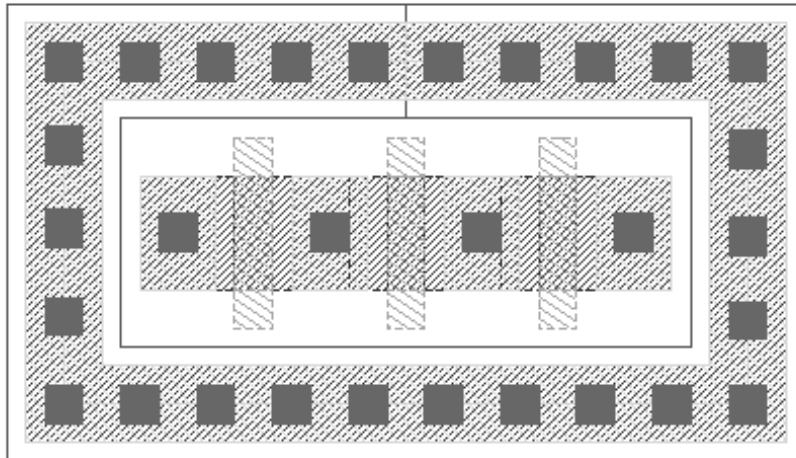


**Parameter Defaults** lets you see and change whether the *Diffusion* layer (master path) guard ring is choppable and the spacing method for vias. When you turn on *Parameter Defaults*, the lower part of the Install Device form is replaced with the following fields:

A screenshot of the 'Install Device' dialog box. The title bar is 'Install Device'. The 'Technology Library' is set to 'lib'. The 'Device Type' is 'guardRing'. The 'Name' is 'guardring1'. There is a 'View Device Configuration' button and a 'Delete' button. Below these are checkboxes for 'Layers', 'Rule', 'Stretch Handles', and 'Parameter Defaults' (which is checked). The 'Diffusion Path Choppable' checkbox is checked. The 'Contact Spacing Method' is set to 'distribute'. At the bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

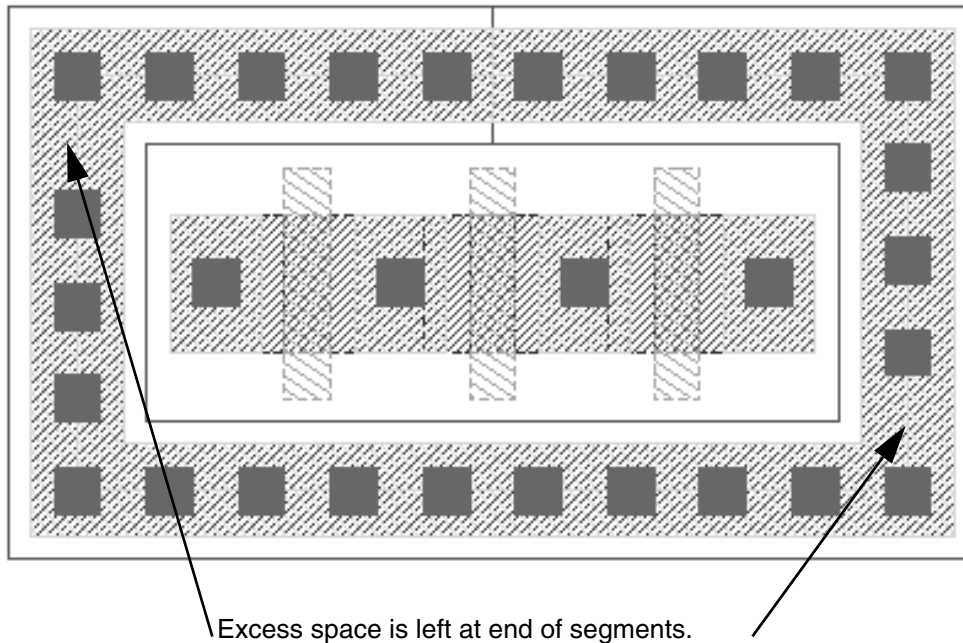
**Diffusion Path Choppable** lets you make the *Diffusion* layer (master path) choppable (on) or not (off). All other parts of the guard ring (*Metal*, *Contact*, and *Implant1* layers) are always choppable. The default for the *Diffusion* layer is choppable.

**Contact Spacing Method** lets you control spacing between vias. When the value is *distribute*, the system uses the defined spacing rule to place vias until there is no space for another via in each guard ring segment, then distributes remaining space as evenly as possible between the vias in multiples of the grid space specified by the `mfgGridResolution` rule in the technology file. If any space remains, it is placed after the last via in the segment.



Excess space is distributed evenly around vias.

When the value is *minimum*, the system uses the specified spacing rule to place vias in each guard ring segment until there is no space for another via, then leaves all excess space after the last via in the segment.



## Defining and Installing a New Guard Ring

Defining a new guard ring template and installing it in your technology file requires the following tasks:

[Choosing the Technology Library, Device Type, and Device Name on page 159](#)

[Specifying Guard Ring Layers on page 159](#)

[Specifying Guard Ring Rules on page 161](#)

[Specifying Guard Ring Parameter Defaults on page 163](#)

[Installing the Guard Ring Template in the Technology File on page 164](#)

## Choosing the Technology Library, Device Type, and Device Name

1. From the *Technology Library* cyclic field, choose the technology library in which to install the new guard ring.
2. For the *Name* field,
  - ❑ If the *Name* field is blank, type in a unique name for your new guard ring.
  - ❑ If the *Name* field already contains the name of a guard ring, click the arrow next to the name and choose the blank space at the top of the list of guard ring names.

The *Diffusion*, *Contact*, and *Metal* fields become blank.

The *Layers* button is turned on by default when the Install Device form first appears. Now you are ready to specify the layers for your new guard ring.

## Specifying Guard Ring Layers

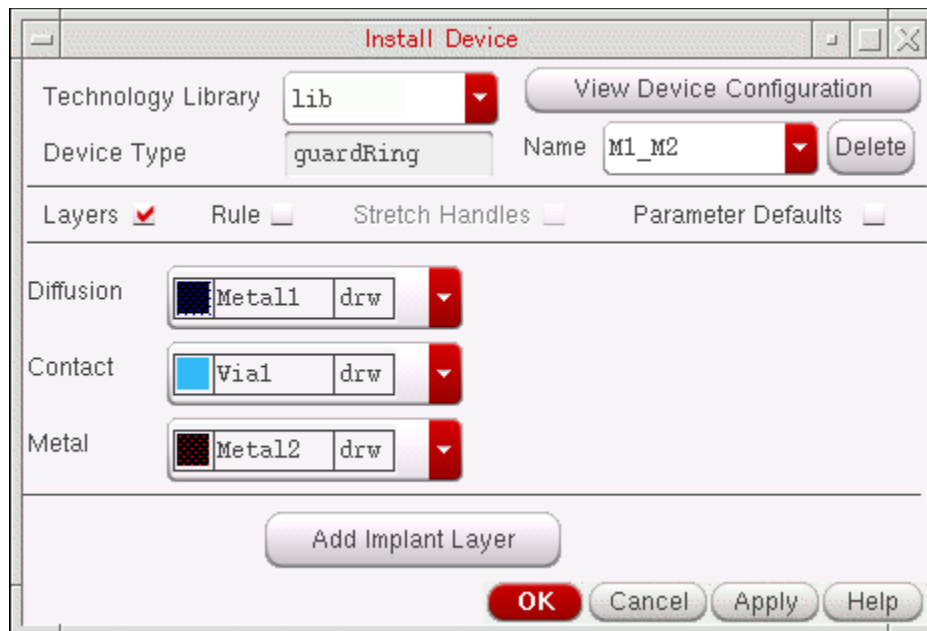
To specify the layers for the new guard ring, in the Install Device form, follow the steps below.

1. From the *Diffusion* cyclic field, choose the layer for the diffusion, which encloses the layer you choose for *Metal*.
2. From the *Contact* cyclic field, choose the layer for vias.
3. From the *Metal* cyclic field, choose the layer that encloses the layer you chose for *Contact*.

## Virtuoso Technology Data User Guide

### Defining and Installing Devices

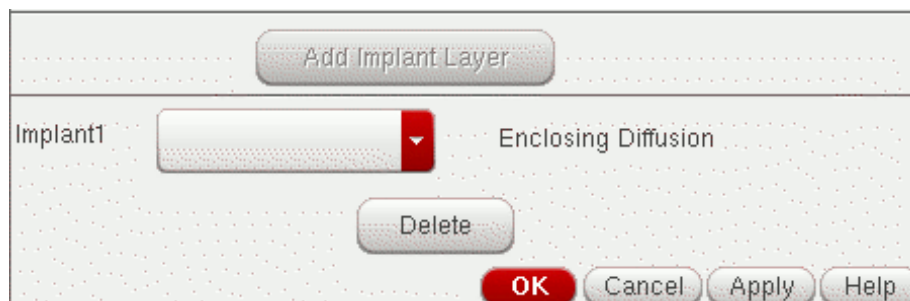
For example, you might choose the following layers for your new guard ring:



4. If you the material you chose for diffusion needs to be enclosed in an implant layer, follow these steps:

- a. Click *Add Implant Layer*.

The Add Implant Layer section appears at the bottom of the Install Device form.



- b. From the *Implant1* cyclic field, choose an implant layer.

**Note:** The *Rules* field is grayed out while the Add Implant Layer form is open, until you select a layer. If you decide you do not want to an implant layer, you need to click *Delete* to close the Add Implant layer section before you can click *Rules*.

Now you are ready to specify the rules for your new guard ring.



## Virtuoso Technology Data User Guide

### Defining and Installing Devices

#### Specifying Guard Ring Rules

To specify the rules for the new guard ring, in the Install Device form, follow the steps below.

**Note:** For guard rings, the layer specified for *Diffusion* must completely enclose the layer specified for *Metal*, and the layer specified for *Metal* must completely enclose the layer specified for *Contact*. When you specify an implant layer, it must completely enclose the layer specified for *Diffusion*.

1. Click to turn on *Rules*.

The lower part of the Install Device form is redrawn to show the applicable design rules from the technology file for the layers you selected.

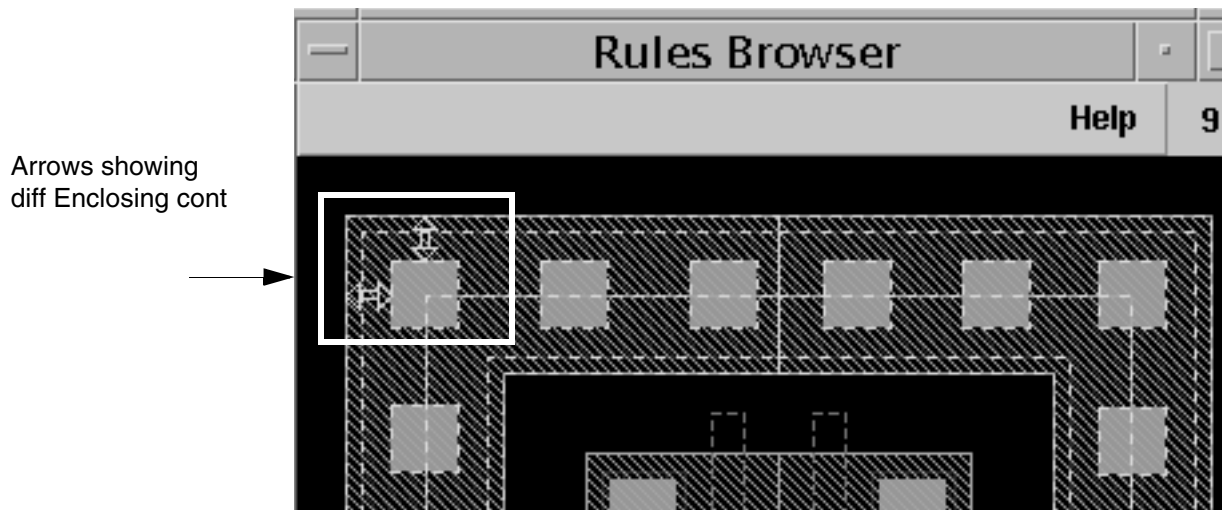
If a required enclosure rule for the selected layers is not defined in the technology file, the system displays the name of the rule and a blank input field. For example, if the *diff*, *cont*, and *metal1* layers are specified, the system checks the technology file for the *diff Enclosing cont* rule, and not finding it, added *diff Enclosing cont* to the Rules section, as shown below:

The 'Install Device' dialog box is shown with the 'Rules' section active. The 'Rule' checkbox is checked. The 'Rules' section contains a table with columns: Width, Length, and TechfileDefault. The 'Metal2 Offset' row has a blank input field in the 'Length' column, which is highlighted by an arrow and a text box stating 'Missing rule has blank input field.'

	Width	Length	TechfileDefault
Via1 Dimensions	0.1400	0.1400	✓
Via1 Spacing		0.1500	✓
Metal2 Offset		0.0	
Metal1 Width		0.1500	
Metal2 Width		0.1500	

When you click *Rules*, the system also displays the Rules Browser window. This window lets you see how each rule is measured: when you click the cursor in a design rule input field in the *Rules* section, the system displays double-ended arrows in the Rules

Browser to show how the rule is measured. For example, for *diff Enclosing cont*, the arrows look like this:



2. Check how any of the design rules are used by clicking in an input field and looking at the Rules Browser.

When the input area for a rule is blank, meaning the required rule is not defined in the technology file, you must provide a value.

3. If the input field for a design rule is missing, type in the value you want.

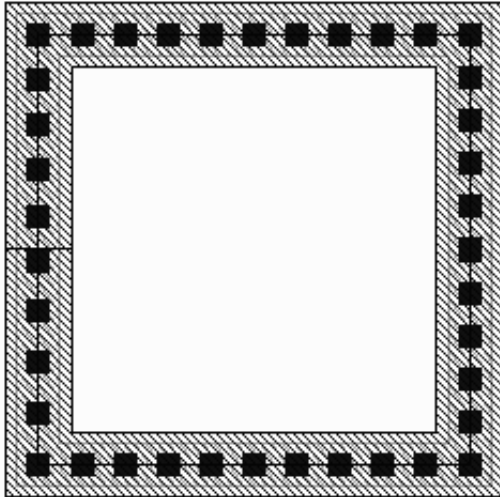
A missing rule appears in the *Rules* section until you click *Apply* or *OK* and save the new value to the technology file in virtual memory; then, since the rule is no longer missing, the *Rules* section no longer displays it.

Once there is a value for every rule, you can look at the configuration of the guard ring in the View Device Configuration window.

4. To see the current guard ring configuration, click the *View Device Configuration* button at the top of the Install Device form.

If the value for a rule is missing and you click *View Device Configuration*, the system displays a dialog box asking you to enter a value for the rule.

For the example using the *diff*, *cont*, and *metal1* layers, the View Device Configuration window might look like this:



Any time you make a change to the rules, you can click the *View Device Configuration* button to refresh the image in the View Device Configuration window. You do not have to click *Apply* or *OK* to see the results of a change you make to a design rule.

5. If you want to change the value for a design rule, type over the existing value.
6. To see the results of changing the value for a rule, click *View Device Configuration* at the top of the Install Device form.

The View Device Configuration window is refreshed, showing a current picture of the guard ring.

The system checks the values in the *Rules* section to make sure the guard ring will be created correct-by-construction. If it is not, the system adjusts the values. For a description of the verification performed, see [“Adding or Modifying Rules: System Verification”](#) on page 154.

Now you are ready to specify the parameter defaults for your guard ring.

### **Specifying Guard Ring Parameter Defaults**

To specify whether the *Diffusion* layer choppable and how the vias are spaced, follow the steps below.

1. Click to turn on *Parameter Defaults*.

## Virtuoso Technology Data User Guide

### Defining and Installing Devices

---

The lower part of the Install Device form is redrawn to show these fields:



The default is for the *Diffusion* layer (master path) to be choppable. All other parts of the guard ring (*Metal* layer, *Contact* layer, and *Implant1* layer, when there is one) are always choppable.

2. To prevent the *Diffusion* layer from being chopped (cut), turn off *Diffusion Path Choppable*.
3. For *Contact Spacing Method*, choose *distribute* or *minimum*.

For more information see [Contact Spacing Method](#).

4. To see what your change to the spacing between rectangles looks like, click *View Device Configuration* at the top of the Install Device form and look at the View Device Configuration window.

The View Device Configuration window refreshes, showing a current picture of the guard ring.

Now you are ready to install the new guard ring template in the technology file.

### Installing the Guard Ring Template in the Technology File

Once you have completely defined your new guard ring and are ready to save the guard ring template in the technology library, follow the steps below.

**Note:** The system checks the values in the *Rules* section whenever you click *Apply* or *OK* to make sure the guard ring will be created correct-by-construction. If it is not, the system adjusts the values. For a description of the verification performed, see [“Adding or Modifying Rules: System Verification”](#) on page 154.

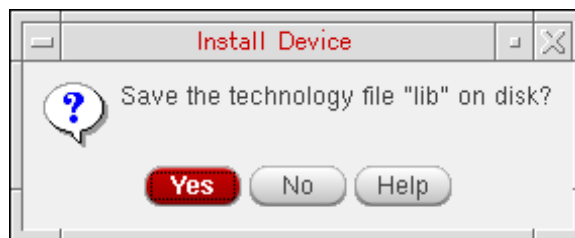
1. On the Install Device form, click *Apply* or *OK*.

If you entered a value for a rule that did not have a default in the technology file (a missing rule), then when you click *OK* or *Apply*, the system displays a dialog box asking if you want to add the rule to the technology file, like this:



2. If the Save All Rules dialog box appears, and you want to save all new rules to the technology file in virtual memory, click *Yes*; otherwise, click *No*.

When you click *OK* (or *Cancel*, after clicking on *Apply*), the system displays a dialog box, asking if you want to save the technology file to disk. When you click *Yes*, all work you have performed during the current editing session is saved from the technology file in virtual memory to the technology file on disk.



3. If you want to save your work to the technology file on disk, click *Yes* in the Install Device dialog box; otherwise, click *No*.

Your new guard ring has been saved to the technology file in virtual memory.

For messages that can appear after you click *Apply* or *OK*.

### Defining a New Guard Ring from an Existing Guard Ring

To define a new guard ring template based on the definition of an existing guard ring template, do the following on the Install Device form:

1. From the *Technology Library* cyclic field, choose the technology library containing the guard ring template you want to use as a base.
2. From the *Device Type* cyclic field, choose *guardRing*.
3. From the *Name* cyclic field, choose the name of the guard ring you want to use as a base.

The form displays the definition of the selected guard ring.

4. In the *Name* field, change the existing name to a unique name for the new guard ring template.
5. Change whatever layers, rules, and parameter defaults you want for the new guard ring. For specific instructions about specifying layers, rules, and parameter defaults, see [“Defining and Installing a New Guard Ring”](#) on page 158.
6. To see the results of your changes to the existing guard ring, click *View Device Configuration* at the top of the Install Device form.

The View Device Configuration window is refreshed, showing a current picture of the guard ring.

7. To install the new guard ring in your technology library,

If you entered a value for a rule that had a blank input field, the software displays a message box, asking if you want to add new rules to the technology file in virtual memory.

- a. To add the new rules to the technology file in virtual memory, click *Yes*; otherwise, click *No*.

When you click *OK* (or *Cancel*, after clicking on *Apply*), the system displays a dialog box, asking if you want to save the technology file to disk. When you click *Yes*, all work you have performed during the current editing session is saved from the technology file in virtual memory to the technology file on disk.

- b. If you want to save your work to the technology file on disk, click *Yes* in the Install Device dialog box; otherwise, click *No*.

**Note:** The system checks the values in the *Rules* section whenever you click *Apply* or *OK* to make sure the guard ring will be created correct-by-construction. If it is not, the system adjusts the values. For a description of the verification performed, see [“Adding or Modifying Rules: System Verification”](#) on page 154.

You have completed creating a new guard ring from an existing guard ring.

## Modifying an Existing Guard Ring

To change the definition of an existing guard ring template, do the following on the Install Device form:

1. From the *Technology Library* cyclic field, choose the technology library containing the guard ring template you want to redefine.
2. From the *Device Type* cyclic field, choose *guardRing*.
3. From the *Name* cyclic field, choose the name of the guard ring template you want to redefine.

The form displays the definition of the selected guard ring template.

4. Change whatever layers, rules, and parameter defaults you want for the redefined guard ring template. For specific instructions about specifying layers, rules, and parameter defaults, see [“Defining and Installing a New Guard Ring”](#) on page 158.
5. To see the results of your changes to the guard ring, click *View Device Configuration* at the top of the Install Device form.

The View Device Configuration window is refreshed, showing a current picture of the guard ring.

6. To install the redefined guard ring in your technology library,

If you entered a value for a rule that had a blank input field, the software displays a message box, asking if you want to add new rules to the technology file in virtual memory.

- a. To add the new rules to the technology file in virtual memory, click *Yes*; otherwise, click *No*.

When you click *OK* (or *Cancel*, after clicking on *Apply*), the system displays a dialog box, asking if you want to save the technology file to disk. When you click *Yes*, all work you have performed during the current editing session is saved from the technology file in virtual memory to the technology file on disk.

- b. If you want to save your work to the technology file on disk, click *Yes* in the Install Device dialog box; otherwise, click *No*.

**Note:** The system checks the values in the *Rules* section whenever you click *Apply* or *OK* to make sure the guard ring will be created correct-by-construction. If it is not, the system adjusts the values. For a description of the verification performed, see [“Adding or Modifying Rules: System Verification”](#) on page 154.

You have completed modifying an existing guard ring.

## Deleting a Guard Ring

To delete a guard ring template from the technology library, do the following on the Install Device form:

1. From the *Technology Library* cyclic field, choose the technology library containing the guard ring template you want to delete.
2. From the *Device Type* cyclic field, choose *guardRing*.



***Templates listed in the Name field could have been created by other users. Be careful not to delete a guard ring template that might be needed by other users.***

3. From the *Name* cyclic field, choose the name of the guard ring template you want to delete.
4. Click *Delete*.

The software displays a verification message box, asking if you want to remove the selected guard ring the technology library.



When you delete a guard ring template, it is immediately removed from the technology file in virtual memory. The next time the technology file is saved to disk, it is also removed from disk. Once you click Yes, you cannot retrieve this guard ring template; if you want it again, you must redefine and reinstall it.

5. To remove the guard ring template from the technology library in virtual memory, click Yes.

You have completed deleting a guard ring from the technology file in virtual memory.



---

## Form Descriptions

---

This appendix presents descriptions of the user interface forms discussed in this user guide.

## **New Technology Library Form**

**Technology Library Name** is the name of the new technology library to create.

**Load ASCII Technology File** lets you specify the name of the ASCII file you want to compile and install in the new technology library.

**Copy From Existing Technology Library** lets you choose the name of the technology library you want to copy and install in the new technology library.

**Reference Existing Technology Libraries** lets you establish a library containing only references to other libraries that you select and order.

**Browse** lets you browse your directories and files.

**Directory (non-library directories)** is the directory in which you want to place the new technology library.

**Design Manager** indicates the design management system you use. The options displayed in this cyclic field are those that are available to you.

## **SKILL Function to Display Form**

`tcDisplayNewTechForm()`

## Reference Existing Technology Libraries Form

**New Library** is the name of the technology library to which you want to add references to other technology libraries.

**Types** lets you list all technology libraries or selectively only IC or Package technology libraries.

**Technology Libraries** lists the other existing technology libraries and allows you to select one library at a time to add to the list of reference technology libraries.

**Reference Technology Libraries** lists the technology libraries referenced by the library in the *New Library* field and allows you to add a library selected in the *Technology Libraries* list box to the list of reference technology libraries.

### SKILL Function to Display Form

`tcNewLibDisplayRefTechForm(t_newLibName)`

## Technology Database Graph Form

**Library** is the name of the technology database at the top of the database graph you want to see.

**Effective** displays the effective graph, with multiple references to the same database resolved.

**As Defined** displays the graph of the technology databases as defined, showing multiple references to the same database by other databases.

### SKILL Function to Display Form

`tcDisplayTechGraphForm( [ d techID ] )`

## Attach Technology Library to Design Library Form

**Design Library** is the name of the DFII design library to which you want to attach the technology library.

**Types** lets you list all technology libraries or selectively only IC or Package technology libraries.

**Technology Library** is the name of the technology library you want to attach to the design library.

### SKILL Function to Display Form

`tcDisplayAttachTechForm()`

## Load Technology File Form

**ASCII Technology File** is the name of the ASCII technology file you want to compile and load into a technology library in virtual memory.

**Browse** lets you browse your directories and files.

**Classes** lets you select specific technology file sections to load.

**Select All** selects all sections and indicates that you want to load the entire file.

**Technology Library** is the name of the technology library you want to update. Select one of the libraries in your search path from this cyclic field.

**Merge** indicates that you want to combine the data in the ASCII technology file with the technology data already in the technology library in virtual memory.

**Note:** Merging data replaces data specifications that are order dependent. Non-order-dependent data is appended to the existing data in the technology library in virtual memory.

**Replace** indicates that you want to overwrite technology data loaded into virtual memory with the database compiled from the ASCII technology file.

## SKILL Functions to Display Form

[tcDisplayLoadTechForm\(\)](#)

[tcDisplayCompTechForm\(\)](#)

## Dump Technology File Form

**Technology Library** is the name of the library containing the technology file you want to dump. Select one of the libraries in your search path from this cyclic field.

**Classes** lets you select specific technology file sections to dump.

**Select All** selects all sections and indicates that you want to dump the entire technology database to the specified technology file.

**Dump Empty Section Headers** lets you dump empty sections in techDB. For example, if there is no `viaDef` in a techDB and the *Dump Empty Section Headers* option is selected, then the tech dumper dumps an empty header for the `viaDef`, as shown below:

```
;*****
; VIADEFS
;*****

viaDefs (

    standardViaDefs (
        ; ( viaDefName  layer1  layer2  (cutLayer cutWidth cutHeight
[resistancePerCut])
        ;   (cutRows   cutCol   (cutSpace))
        ;   (layer1Enc) (layer2Enc)   (layer1Offset) (layer2Offset) (origOffset)
        ;   [implant1   (implant1Enc) [implant2       (implant2Enc)
[well/substrate]]])
        ; (
        -----
    )
    ) ;standardViaDefs

    customViaDefs (
        ; ( viaDefName libName cellName viewName layer1 layer2 resistancePerCut)
          ; (viaVariantName viaDefName (paramName paramValue) ...) ; (
        -----
    )
```

```
) ;customViaVariants  
) ;viaDefs
```

**ASCII Technology File** is the name of the ASCII technology file to create.



***It is safest to specify a new technology filename rather than overwriting an existing technology file. An ASCII file produced with the Dump command does not contain any of the comments or SKILL programs that your original file might contain. You may want or need the original technology file later.***

**Open in Techfile IDE** lets you open the technology file in the user-friendly Techfile IDE.

**Browse** lets you browse your directories and files.

**OK** or **Apply** initiates the dump.

Sometimes the dump ASCII technology file is different from the load ASCII technology file. This is because of the following reasons:

1. The original technology file has system reserved layer:

In the IC610 release, every single `techLib` became an ITDB where `techLib` inherits the system reserved layers and other system specific information from `cdsDefTechLib` and technology file load ignores this information from ASCII `techFile`.

Therefore, when the technology file is dumped again following the standard ITDB convention, only the top level `techLib` which does not have the system reserved info is dumped back.

2. The original `techFile` has a `spacingTable` in which all the rule value for all the indexes of the table are not explicitly specified. Let's consider the example given below:

**Original technology file:**

```
( minNumCut          "Via8"  
  (( "width"   nil    nil ) 1 )  
  (  
    0.5        2  
    1.0        4  
  )  
)  
) ;spacingTables
```



#### After Load/Dump Roundtrip:

```
( minNumCut      "Via8"
  (( "width"      nil      nil ) 1 )]
  (
    0.36          1
    0.5           2
    1.0           4
  )
)
```

) ;spacingTables

In this case the minWidth of Via8 layer is 0.36 but the minNumCut rule does not explicitly specify how many cuts should be used when the width is 0.36. Therefore, the techFile loader estimates this rule and puts it into the techLib, which then gets dumped in the ASCII techFile.

#### SKILL Function to Display Form

[tcDisplayDumpTechForm\(\)](#)

## Technology File – Layer Browser Forms

**Technology Library** is the library that contains the technology file currently selected for editing.

**Filters** lets you choose whether to display user-defined layers, system layers, or both.

**Layer, Layer 1, Layer 2, Via, Implant Layer, Legal Region Layer** displays the existing layers of the type in the heading and lets you choose a layer or layer-purpose pair by clicking on it in the list. For the Layer Rules `equivalentLayers` subclass, which specifies multiple layers, you can choose more than one layer by holding down the `Ctrl` key while clicking on each layer.

## **Discard Edits To Technology File Form**

**Technology Library** is the name of the library containing the technology file you want to load from disk. From this cyclic field, select one of the libraries in your search path.

### **SKILL Function to Display Form**

`tcDisplayDiscardTechForm()`

## Save Technology File Form

**Technology Library** is the name of the library containing the technology file you want to save. From this cyclic field, select one of the libraries in your search path.

### SKILL Function to Display Form

`tcDisplaySaveTechForm()`

## **Install Device Form**

For information on this form, see [Bringing up the Install Device Form](#) on page 144.

### **SKILL Function to Display Form**

`tcOcInstallDevices()`

## Merge Display Resource Files (DRF) Form

**Select DRF to merge** is the section in which you specify the display resource files to merge.

**From Library** list box lets you select display resource files from your technology libraries.

**From File** lets you type in the path name of a display resource file to merge.

**Browse** lets you browse your directories and files to locate a file to merge.

**Add** adds the specified display resource file to the bottom of the *Merge DRF files in sequence* list.

**Note:** You add display resource files to the list one at a time. Add the last file to merge first. (The order is important. If a display resource is defined in more than one file being merged, the last definition loaded overwrites any earlier ones.) Add the files so that the first file you add is the last file to merge and the last file you add is the first file to merge. (Files listed in *Merge DRF files in sequence* are merged from the bottom up.)

**Merge DRF files in sequence** is the section that lists the files to be merged.

**Delete** lets you delete a selected file from the list of files to be merged.

**Destination DRF** lets you specify the path name of the new display resource file to create from the merged files. The filename must be `display.drf`.

**Browse** lets you specify the directory where you want to save the merged file.

**Load Merged DRF** lets you load the display resource file immediately after the merge operation is complete, making it possible for you to view the changes as soon as you run the *Redraw* command.

**OK** or **Apply** initiates the merge.

**Cancel** closes the form without performing a merge.

**Defaults** clears the *From File* and *Destination DRF* fields and sets the *Load Merged DRF* check box to the default value (selected).

## Display Resource Editor (DRE) Form

**File** lets you load a display resource file, save changes to a display resource file on disk, reinitialize, or exit Display Resource Editor.

**Help** opens the Cadence Help window.

**Device** lets you add a new display device or select a display device defined in the display resource file. The list of packets and layer-purpose pairs is refreshed when you select a different display device.

**View Mode** lets you specify whether you want to view a list of display packets or layer-purpose names. You can choose to display *All LPPs*, *All Valid LPPs*, or *Active Palette Filtered LPPs*.

**Technology Library Names** lets you select the technology file that defines the layers you want to edit.

**Search** lets you filter layer-purpose names and packet names.

**Associations** lists all LPPs that use the same display packet.

**Fill Style** lists the fill styles defined in the display resource file. Click a fill style to view associated fill color, outline color, stipple, and line style attributes.

**Fill Color** lists the fill colors defined in the display resource file.

**Outline Color** lists the outline colors defined in the display resource file.

**Stipple** lists the stipple patterns defined in the display resource file.

**Line Style** lists the line styles defined in the display resource file.

**Create New Packet** lets you create a new display packet.

**Modify Current Packet** updates in virtual memory the changes that you made to a display packet.

## SKILL Function to Display Form

[dreInvokeDre\(\)](#)

## Color Editor Form

**Search** lets you filter colors.

**Color** lists the colors already defined in the display resource file. Click a color to edit it.

**Create New** lets you create a new color with the specified name.

**Blink**, if selected, causes the color to blink.

**Add to Custom Colors** lets you add the new or modified color to the *Custom colors* section.

**Hue/Sat/Val/Red/Green/Blue** let you set the RGB values of the color. You can also drag the sliders to select a new color value.

**Swatch** displays the current and modified color. The *Swatch* area is updated as you drag the sliders or specify RGB values.



## Stipple Editor Form

**Search** lets you filter stipples.

**Stipple** lists the stipples already defined in the display resource file. Click a stipple to edit it.

**Create New** lets you create a new stipple with the specified name.

**Size** lets you specify the resolution of the grid. The larger the value, the higher the resolution.

**Clear** clears the grid so that you can start over.

**Invert** reverses the display of pixels in the grid and the *Swatch* area.

**Swatch** displays the current and modified stipple pattern. The *Swatch* area is updated as you click the pixels in the grid.

## Line Style Editor Form

**Search** lets you filter colors.

**Line Style** lists the line styles already defined in the display resource file. Click a line style to edit it.

**Create New** lets you create a new line style with the specified name.

**Size** lets you specify the resolution of the line. The larger the value, the higher the resolution.

**Thickness** lets you specify the thickness of the line.

**Clear** clears the grid so that you can start over.

**Invert** reverses the display of pixels in the grid and *Swatch* area.

**Swatch** displays the current and modified line style. The *Swatch* area is updated as you click the pixels in the grid.

## Load Form

**Look in** lists the current directory name. Click the directory name to view the files and directories stored in it. Select a `.drf` file and click *Open* to load it.

**Parent Directory** lets you move one level up to view the list of files and directories stored in the parent directory of the current directory.

**File name** lets you specify the name of the file that you want to load. The *Open* button is enabled only when you specify a valid `.drf` filename in the *File name* field.

**File of type** lets you filter the list of files to display only the files with extension `.drf`.

## Save As Form

**Look in** lists the current directory name. Click the directory name to view the files and directories stored in it.

**Parent Directory** lets you move one level up to view the list of files and directories stored in the parent directory of the current directory.

**File name** lets you specify the name of the file to which you want to save your changes. By default, the new `.drf` file is saved in the current directory.

**File of type** lets you automatically save the new file with extension `.drf`.

---

## Environment Variables

---

This appendix contains the Virtuoso® Technology Data environment variable names, descriptions, types, and values.

**Note:** Undocumented environment variables are private and are subject to change at any time.

### Working with Environment Variables

#### Checking the Value of an Environment Variable

To determine the current value of an environment variable,

- Type it the CIW

```
envGetVal( "cdba" "variable_name" )
```

For example

```
envGetVal( "cdba" "noTechUpRev" )
```

#### Changing the Default Settings of Environment Variables

You can change the default settings of environment variables as follows:

- 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.

## Virtuoso Technology Data User Guide

### Environment Variables

---

For example, to set the `noTechUpRev` variable, type the following in the CIW or include it in a file:

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

To determine the current value of any layout editor environment variable, type the `envGetVal()` function in the CIW as follows:

```
envGetVal("cdba" "variable_name")
```

### 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.

## Technology Data Environment Variables

This section describes environment variables in Virtuoso® Technology Data.

- AlternateFoundryCG
- defaultAttachTech
- noTechUpRev

### AlternateFoundryCG

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

#### Description

(ICADVM20.1 Only – 95511) Specifies the name of the constraint group that contains the foundry constraints. All constraint groups that are intended to be used as foundry constraint groups must be defined using the alternateFoundry constraint group definition. In the Layout Editor Options form, the Editor field will show only those constraint groups that are defined with the `alternateFoundry` constraint group definition.

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

#### GUI Equivalent

Command	<i>Options – Editor</i>
Form Field	<i>Foundry Constraints (<u>Layout Editor Options Form</u>)</i>

#### Examples

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

## defaultAttachTech

```
cdba defaultAttachTech string t_techLibraryName
```

### Description

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

For information about attaching new libraries, see [Attaching to an Existing Technology Library](#) in the *Cadence Library Manager User Guide*.

### GUI Equivalent

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

### Examples

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



## noTechUpRev

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

### Description

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

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

### GUI Equivalent

None

### Examples

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

## **Virtuoso Technology Data User Guide**

### **Environment Variables**

---

---

## Data Handling for Incremental Technology Databases

---

### Major Data Duplication Rules Summary

You can duplicate some technology data in more than one technology database if doing so accomplishes your design flow objectives. Duplicating most technology data, however, creates conflicts, which must be avoided.

The following paragraphs summarize the major rules of duplication to use to avoid creating conflicts; they delineate

- what can and cannot be duplicated in multiple technology databases in the same graph
- how the software handles specific conflicts between databases
- how the software handles situations when referenced objects are not where they need to be
- special allowance for creating layers that match Virtuoso reserved system layers using the environment variable CDS\_ALLOW\_VIRTSYSLAY Environment Variable on page 200.

This section does not deal with every possible data specification. For complete information, see the Data Handling Table, which provides complete information about how the software handles data in every situation, including how it detects and handles conflicts.

Multiple technology bases in a graph can contain duplicates of the following devices of the same name without conflict:

- `MPPTemplates` of the same name
- user-defined non-`viaDef` devices

The following subsections can also be duplicated in multiple technology databases:

- `foundry` constraint groups and their contents, with the requirement that the objects referenced exist somewhere in the subgraph
- default constraint groups and their contents, with the requirement that the objects referenced exist somewhere in the subgraph
- layer-purpose pairs and their drawing priority order
- `techLayerPurposePriorities` for the same named layers and purposes
- `equivalentLayers`
- `stampLabels`
- `labels`
- `leLswLayers`

You cannot duplicate most other technology data, such as `viaDefs` or any constraint group with the same name other than a `foundry` or `default` constraint group.

Although you can specify the `dbuPerUU` value in every technology database, the value must be the same in all of them.

In certain cases, related data must be specified in the same technology database:

- Any `cdsViaDevice` and its `viaDef` must be in the same technology database.
- For any `techDisplays` definition, the specified layer-purpose pair must be defined in the same technology database.

For information about conflict types and how they occur, see [“Conflict Avoidance”](#) on page 30.

## Data Handling Table

The table in this appendix summarizes how the software checks for data conflicts in a graph and handles the data, conflict or not. The conflict checking and resultant behavior defined in this table apply whenever you introduce a new technology database or make any additions or changes to an existing technology database in a graph, whether compiling a technology file or setting or creating data with a SKILL function. When defining technology data, it is important to be cognizant of the incremental technology database graph and how the data you introduce works with or can conflict with the other data in the graph.

# Virtuoso Technology Data User Guide

## Data Handling for Incremental Technology Databases

ASCII technology file section/ subsection	S/W checks graph for matching	Allow duplicates in graph?	Action taken when match is defined in:			Referenced objects	Action taken when referenced object is defined in:		
			No db	local db	reference db		No db	local db	reference db
Controls									
techParams	Parameter name	Disallowed. Issue warning	Create in local	Overwrite in local	Conflict	N/A	N/A	N/A	N/A
viewTypeUnits	View type name	Disallowed. Issue error	Create in local	Overwrite in local	Conflict	N/A	N/A	N/A	N/A
mfgGridResolution	Construct name	Disallowed. Issue error	Create in local	Overwrite in local	Conflict	N/A	N/A	N/A	N/A
refTechLibs	Construct name	N/A <sup>2</sup>	Create in local	Overwrite in local	Create in local	N/A	N/A	N/A	N/A
layerDefinitions									
techLayers	Layer number or name	Disallowed. Issue warning	Create in local	Warning or do nothing <sup>3</sup>	Conflict	N/A	N/A	N/A	N/A
techPurposes	Purpose number or name	Disallowed. Issue warning	Create in local	Warning or do nothing <sup>3</sup>	Conflict	N/A	N/A	N/A	N/A
techLayerPurposePriorities	Layer name and purpose name	Allowed <sup>1</sup>	Create in local	Reorder in local	Create in local	Layer, purpose	Disallowed	Update in local	Update in local
techDisplays	LPP name	Allowed <sup>1</sup>	Create in local	Overwrite in local	Create in local	LPP	Disallowed	Update in local	Dis-allowed
techLayerProperties	Layer name and property name	Disallowed	Create in local	Overwrite in local	Error <sup>4</sup>	Layers	Disallowed	Update in local	Dis-allowed
techDerivedLayers	Derived layer number or name	Disallowed. Issue warning	Create in local	Warning or do nothing <sup>3</sup>	Conflict	Layers	Disallowed	Update in local	Update in local
layerRules									
functions	Layer name	N/A <sup>5</sup>	Create in local	Overwrite in local	Error <sup>4</sup>	Layers	Disallowed	Update in local	Dis-allowed
mfgResolutions	Layer name	N/A <sup>5</sup>	Create in local	Overwrite in local	Error <sup>4</sup>	Layers	Disallowed	Update in local	Dis-allowed
routingDirections	Layer name	N/A <sup>5</sup>	Create in local	Overwrite in local	Error <sup>4</sup>	Layers	Disallowed	Update in local	Dis-allowed

# Virtuoso Technology Data User Guide

## Data Handling for Incremental Technology Databases

ASCII technology file section/ subsection	S/W checks graph for matching	Allow duplicates in graph?	Action taken when match is defined in:			Referenced objects	Action taken when referenced object is defined in:		
			No db	local db	reference db		No db	local db	reference db
currentDensity/ currentDensity-Tables	Layer name	N/A <sup>5</sup>	Create in local	Overwrite in local	Error <sup>4</sup>	Layers	Disallowed	Update in local	Disallowed
equivalent-Layers	Construct name	Allowed <sup>1</sup>	Create in local	Overwrite in local	Create in local	Layers	Disallowed	Update in local	Update in local
stampLabels	Construct name	Allowed <sup>1</sup>	Create in local	Overwrite in local	Create in local	Layers	Disallowed	Update in local	Update in local
labels	Construct name	Allowed <sup>1</sup>	Create in local	Overwrite in local	Create in local	Layers	Disallowed	Update in local	Update in local
<b>constraintGroups</b>									
constraintGroup (foundry/default)	constraint-Group name	Allowed	Create in local	Overwrite in local	Create in local	N/A	N/A	N/A	N/A
constraintGroup (non-foundry/default)	constraint-Group name	Disallowed	Create in local	Overwrite in local	Conflict	N/A	N/A	N/A	N/A
constraints (native OA, in foundry/default constraint group)	CG name, constraint name, layer number or name, or viaDef name	Allowed	Create in local	Overwrite in local	Create in local	Layers, viaDefs	Disallowed	Update in local	Update in local
constraints (native OA, in non-foundry/default constraint group)	CG name, constraint name, layer number or name, or viaDef name	Disallowed	Create in local	Overwrite in local	N/A <sup>6</sup>	Layers, viaDefs	Disallowed	Update in local	Update in local
LPP based / user defined rule (in foundry/default CG)	CG name, rule name, layer / LPP number or name	Disallowed	Create in local	Overwrite in local	Conflict	Layers, LPPs	Disallowed	Update in local	Update in local
LPP based / user defined rule (in non-foundry/default CG)	CG name, rule name, layer / LPP number or name	Disallowed	Create in local	Overwrite in local	N/A <sup>6</sup>	Layers, LPPs	Disallowed	Update in local	Update in local
<b>siteDefs</b>									
scalarSiteDefs	siteDef name	Disallowed	Create in local	Overwrite in local	Conflict	N/A	N/A	N/A	N/A

# Virtuoso Technology Data User Guide

## Data Handling for Incremental Technology Databases

ASCII technology file section/ subsection	S/W checks graph for matching	Allow duplicates in graph?	Action taken when match is defined in:			Refer- enced objects	Action taken when referenced object is defined in:		
			No db	local db	refer- ence db		No db	local db	refer- ence db
arraySiteDefs	siteDef name	Disallowed	Create in local	Overwrite in local	Conflict	siteDefs	Disallowed	Update in local	Update in local
<b>viaDefs</b>									
standardVia- Defs	viaDef name	Disallowed	Create in local	Overwrite in local	Conflict	Layers	Disallowed	Update in local	Update in local
customViaDefs	viaDef name	Disallowed	Create in local	Overwrite in local	Conflict	Layers	Disallowed	Update in local	Update in local
<b>viaSpecs</b>									
viaSpecs	Layer pair names	Disallowed	Create in local	Overwrite in local	Conflict	Layers, viaDefs	Disallowed	Update in local	Update in local
<b>Devices</b>									
cdsViaDevice	Mapped viaDefs name	Disallowed 7	Create in local	Overwrite in local	Conflict	Layers, LPPs	Disallowed	Update in local	Update in local
ruleContact- Device	Mapped viaDefs name	Disallowed 7	Create in local	Overwrite in local	Conflict	Layers, LPPs	Disallowed	Update in local	Update in local
multiplepart- PathTemplates	Device name	Allowed	Create in local	Overwrite in local	Create in local	Layers, LPPs	Disallowed	Update in local	Update in local
<b>leRules</b>									
leLswLayers	Construct name	Allowed 1	Create in local	Overwrite in local	Create in local	LPPs	Disallowed	Update in local	Update in local

# Virtuoso Technology Data User Guide

## Data Handling for Incremental Technology Databases

---

### Footnotes

<sup>1</sup> Different definitions are allowed in multiple technology databases in the graph. During lookup, the most local definition will be returned and will be applied to the effective technology database in the graph.

<sup>2</sup> Each reference technology library (`refTechLibs`) is exclusively applied to the techDB in which it was defined. Two definitions of `refTechLibs` in different techDBs are considered as different objects.

<sup>3</sup> Issue a warning if the definition in the ASCII technology file and the definition found in the local technology database have partially matched characteristics (e.g., two layers defined with the same layer number but different layer names, or with the same layer name but different layer numbers). Do nothing if the definition in the ASCII technology file and the definition found in the local technology database have fully matched characteristics. The above behavior is applied to technology file compilation in Merge mode and techSet APIs, not for technology file compilation in Replace mode.

<sup>4</sup> A layer property (or layer attribute) must exist in the same technology database with the layer. Since no duplicate layers are allowed in the graph, it is an error to define a layer property or a layer attribute with the same property/attribute name on the same layer.

<sup>5</sup> A layer attribute is not a database object because it does not have database ID.

<sup>6</sup> Native OpenAccess constraint and user-defined LPP-based rules only exist within a constraint group. Since it is not allowed to define two non-foundry/default constraint groups with the same constraint group name, it is impossible to have two native OpenAccess constraints or user-defined LPP-based rules with the same matching characteristics, because the constraint group name is part of the matching characteristics and it has already been checked.

<sup>7</sup> Some devices (`cdsViaDevice`, `ruleContactDevice`, `symContactDevice`, `symEnhContactDevice`) are mapped to `viaDefs` in an OpenAccess database with the same device name. The mapped `viaDef` must be unique in the graph.

## CDS\_ALLOW\_VIRTSYSLAY Environment Variable

A special allowance has been made to let you create, in a technology database, an OpenAccess layer that matches a Virtuoso reserved system layer using the environment variable `CDS_ALLOW_VIRTSYSLAY`.

### Creating layers

With `CDS_ALLOW_VIRTSYSLAY` enabled, create user-defined reserved layers with the SKILL function `techCreateLayer()`. After the layer is created, Virtuoso adds (mark internally) the specified user-defined reserved layers to the techDB.

Only layer creation is affected by this environment variable, no other Virtuoso operations are affected. This process should be done as a one-time setup for the entire DFII session in order to keep consistency in creating (marking) user-defined reserved layers.



As set by `CDS_ALLOW_VIRTSYSLAY`, the internally marked user-defined reserved layer will be temporarily created before saving the techDB. After the save operation is complete, the newly-created reserved layer is deleted and remarked to its original state as before the save operation.

## Deleting layers

The environment variable does not delete user-defined reserved layers rather `techDeleteLayer()` resets the marking of the layer, and the layer is not saved to disk.

## Modifying layers

Modifying user-defined reserved layers is not supported. You cannot use functions such as the following in this situation: `techSetLayerProp()`, `techSetLayerFunction()`, `techSetLayerMaskNumber()`, `techSetLayerName()`.

## Loading an ASCII technology file

With the environment variable set, the specified user-defined reserved layers in the ASCII techfile

- are marked as created in the techDB
- can be saved to disk
- can be dumped to another ASCII techfile

Only the specified user-defined reserved layers are dumped, not all reserved layers.

Without the environment variable set, the specified user-defined reserved layers in ASCII techfile

- are not marked as created in the techDB
- do not issue a warning message to be compatible with previous releases
- can not be saved to disk because layers are not marked
- can not be dumped to new ASCII techfile because layers are not marked

## Virtuoso Technology Data User Guide

### Data Handling for Incremental Technology Databases

---

#### Enabling CDS\_ALLOW\_VIRTSYSLAY

This Boolean environment variable is set in a shell window (not in the CIW or through a DFII GUI nor is it listed in the sample `.cdsenv` file). And it should be used by experienced Virtuoso and techDB users only.

To enable the environment variable, type in a Shell window

```
setenv CDS_ALLOW_VIRTSYSLAY t
```

To disable, type

```
unsetenv CDS_ALLOW_VIRTSYSLAY
```

or

```
CDS_ALLOW_VIRTSYSLAY nil
```

#### Example:

```
setenv CDS_ALLOW_VIRTSYSLAY t
env | grep CDS_ALLOW_VIRTSYSLAY
echo 'tfID = techOpenTechFile("tech_lib_name" "tech.db")' >> test1.au
echo 'layer = techCreateLayer(tfID 235 "prBoundary")' >> test1.au
virtuoso -log test1.log -test test1.au
unsetenv CDS_ALLOW_VIRTSYSLAY
```

---

## Incremental Technology File Examples

---

This appendix presents a simple example of creating multiple technology files for incremental technology databases from an existing, single technology file. It also illustrates an example of a more complicated incremental technology database structure designed for different design tasks to enter at the point in the graph that supplies all of the data needed for each task.

### Example 1

This is a simple example of creating multiple technology files for incremental technology databases from an existing, single technology file.

#### Single ASCII Technology File

The following is a single ASCII technology file containing all of the technology data required for all of the design tasks in a design flow. If another design task requires different technology data, an entirely new technology file must be created and loaded, even if it uses some of the technology data already defined in this technology file. For example, another task might require all of the same basic layers and purposes, but different constraints, `viaDefs`, or `viaSpecs`. Creating a single technology database would require copying and editing the entire original technology file. If the file is split into multiple files for incremental technology databases, then it is necessary to create an ASCII file only for what needs to be different.

# Virtuoso Technology Data User Guide

## Incremental Technology File Examples

---

```

;*****
; LAYER DEFINITIONS
;*****
layerDefinitions(
  techLayers(
    ;( LayerName          Layer#      Abbreviation )
    ;( -----          - - - - -    - - - - - )
    ;User-Defined Layers:
    ( Metall             7           METAL1      )
    ( Vial                8           VIA1        )
    ( Metal2              9           METAL2      )
  ) ;techLayers

```

```

techLayerPurposePriorities(
;layers are ordered from lowest to highest priority
;( LayerName          Purpose      )
;( -----          - - - - -    )
  ( Metall            drawing      )
  ( Vial              drawing      )
  ( Metal2            drawing      )
) ;techLayerPurposePriorities

```

```

techDisplays(
;( LayerName  Purpose  Packet  Vis Sel Con2ChgLy DrgEnbl Valid )
;( -----  - - - - -  - - - -  - - - - - - - - - - - - - - - )
  ( Metall    drawing   m1      t t t t t )
  ( Vial      drawing   v1      t t t t t )
  ( Metal2    drawing   m2      t t t t t )
) ;techDisplays

) ;layerDefinitions

```

```

;*****
; LAYER RULES
;*****
layerRules(
  functions(
    ;( layer          function      [maskNumber] )
    ;( -----          - - - - -    - - - - - )
    ( Metall          "metal"       1           )
    ( Vial             "cut"        2           )
    ( Metal2           "metal"      3           )
  ) ;functions

```

The `layerDefinitions` section defines the layers and purposes used throughout the single technology database. The data is required for all tasks in a design flow. For the incremental technology databases in this example, it is therefore defined in the base technology file containing data used by all other databases.

Destination: `techFile1.tf`

### Notes:

1. If there are any system-reserved layers and purposes defined in a technology file, they are discarded at compilation because they are defined in the default technology database, `cdsDefTechLib`, which is automatically referenced at the base of any graph (see [Simple Incremental Technology Database Structures](#) on page 20)

2. Additional or different layers or layer attributes required only for specific tasks in the design flow can be defined in technology databases providing other data

The `layerRules` section defines layer attributes used throughout the single technology database. The data is required for all tasks in a design flow. For the incremental technology databases in this example, it is therefore defined in the base technology file containing data used by all other databases. Destination: `techFile1.tf`

## Virtuoso Technology Data User Guide

### Incremental Technology File Examples

---

```

;*****
; VIADEFS
;*****

viaDefs(
  standardViaDefs(
    ( viaDefName  layer1  layer2  (cutLayer cutWidth cutHeight [resistancePerCut])
    ;   (cutRows  cutCol  (cutSpace))
    ;   (layer1Enc) (layer2Enc) (layer1Offset) (layer2Offset) (origOffset)
    ;   [implant1 (implant1Enc) [implant2 (implant2Enc) [well/substrate]]])
    ; ( ----- )
    ( M1_M2      Metall1 Metall2  ("Via1" 0.2 0.2)
      (1 1 (0.0 0.0))
      (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0) (0.0 0.0)
    )
  ) ;standardViaDefs

) ;viaDefs

```

viaDefs can be grouped in a second technology database, which can then be referenced by any number of other libraries to use them for their design tasks. Destination: techLib2.ttf, which uses data defined in techLib1 and therefore must reference techLib1.

```

;*****
; CONSTRAINT GROUPS
;*****
constraintGroups(
  ( group      [override] )
  ; ( ----- )
  ( "foundry"   nil

    spacings(
      ( minWidth  "Metall1"  0.3 )
    ) ;spacings
  ) ;foundry
) ;constraintGroups

```

Constraints can be broken out so that they can be referenced by other databases, as can the siteDefs section. Destination: techLib3.ttf, which uses data defined in techLib2 and techLib1 and therefore must reference techLib2, which references techLib1.

```

;*****
; SITEDEFS
;*****
siteDefs(
  scalarSiteDefs(
    ( siteDefName      type width  height  symInX symInY symInR90)
    ; ( ----- )
    ( core              core 10.0  5.0   t t nil)
    ( core2             core 20.0  5.0   t t nil)
    ( IO                pad  20.0  20.0  t t t)
  ) ;scalarSiteDefs

```

## Virtuoso Technology Data User Guide

### Incremental Technology File Examples

---

```
arraySiteDefs(  
; ( name      type  
; ((siteDefName dx      dy      orientation) ...)   
; [symX] [symY] [symR90] )  
; ( ----- )  
  ( tiledIO      pad  
    (   
      ( IO      0.0  0.0  R0  )  
      ( core    0.0  0.0  R90  )  
    )  
    nil t t  
  )  
);arraySiteDefs  
);siteDefs
```

## ITDB ASCII Technology Files Derived from the Single File

This example splits an existing simple technology file into three technology files:

- a base technology file specifying basic layer and purpose data used the other technology databases;
- a technology file defining `viaDefs`, constraints, and devices; and
- a technology file specifying `viaSpecs`.

**Note:** `viaSpecs` does not support `cdsGenViaDefs`.

The single technology file could, of course, be split up differently to satisfy specific design needs. For example, devices might be split into a separate technology file if they are to be used only for some tasks or if various tasks require different groups of devices and other technology databases containing devices are to be added. Also, additional technology files could be added with the appropriate references to create technology databases that use technology data in this graph.

The following are the incremental technology file and database names and their contents used in this example:

---

Graph level	Content	ASCII technology file name	Technology database name
base	layers	techLib1.tf	techLib1
second	viaDefs	techLib2.tf	techLib2
third	viaSpecs	techLib3.tf	techLib3

---

#### Base ASCII Technology File: Layers

Basic layer and purpose data applies to all of the rest of the technology data in the original, single technology file. It will also be used by any design tasks in the design flow, including any tasks requiring by the addition of technology data or, with the incremental technology databases, the addition of technology databases. Consequently, it can be useful to provide a technology database containing only the basic layer information.

The example base technology file `techLib1.tf` defines the basic layer and purpose data for user-defined layers. It is referenced by `techLib2.tf` and itself automatically references the Cadence default technology database, `cdsDefTechLib`, which defines the system-reserved layers and purposes.

## Virtuoso Technology Data User Guide

### Incremental Technology File Examples

---

```

;*****
; LAYER DEFINITIONS
;*****
layerDefinitions(
  techLayers(
    ;( LayerName          Layer#      Abbreviation )
    ;( -----          - - - - -    - - - - - )
    ;User-Defined Layers:
    ( Metall1            7           METAL1      )
    ( Via1               8           VIA1        )
    ( Metal2             9           METAL2      )
  ) ;techLayers

  techLayerPurposePriorities(
    ;layers are ordered from lowest to highest priority
    ;( LayerName          Purpose      )
    ;( -----          - - - - -    )
    ( Metall1            drawing      )
    ( Via1               drawing      )
    ( Metal2             drawing      )
  ) ;techLayerPurposePriorities

  techDisplays(
    ;( LayerName          Purpose      Packet      Vis  Sel  Con2ChgLy  DrgEnbl  Valid )
    ;( -----          - - - - -    - - - - -    ---  ---  - - - - -  - - - - -  - - - )
    ( Metall1            drawing      m1          t  t  t  t  t )
    ( Via1               drawing      v1          t  t  t  t  t )
    ( Metal2             drawing      m2          t  t  t  t  t )
  ) ;techDisplays

) ;layerDefinitions

;*****
; LAYER RULES
;*****
layerRules(
  functions(
    ;( layer              function      [maskNumber])
    ;( -----          - - - - -    - - - - - )
    ( Metall1            "metal"        1              )
    ( Via1               "cut"          2              )
    ( Metal2             "metal"        3              )
  ) ;functions
) ;layerRules

```



## Second ASCII Technology File: ViaDefs

A new controls section must be added to this technology file to reference the base technology database, which contains data required by this second ASCII technology file.

```
controls (
  refTechLibs (
    "techLib1"
  )
) ;controls

;*****
; VIADEFS
;*****

viaDefs (
  standardViaDefs (
    ;( viaDefName  layer1  layer2  (cutLayer cutWidth cutHeight [resistancePerCut])
    ;  (cutRows  cutCol  (cutSpace))
    ;  (layer1Enc) (layer2Enc) (layer1Offset) (layer2Offset) (origOffset)
    ;  [implant1 (implant1Enc) [implant2 (implant2Enc) [well/substrate]]])
    ;( ----- )
    ( M1_M2      Metal1  Metal2  ("Via1" 0.2 0.2)
      (1 1 (0.0 0.0))
      (0.1 0.1) (0.2 0.2) (0.0 0.0) (0.0 0.0) (0.0 0.0)
    )
  ) ;standardViaDefs
) ;viaDefs
```

### Third ASCII Technology File: Constraints and Site Definitions

A new controls section must be added to this technology file to reference the second technology database, which in turn references the base technology database. This third ASCII technology file requires data specified in the other databases.

```
controls(
refTechLibs(
    "techLib2"
)
) ;controls

;*****
; CONSTRAINT GROUPS
;*****
constraintGroups(
; ( group      [override] )
; ( -----    - )
    ( "foundry"    nil

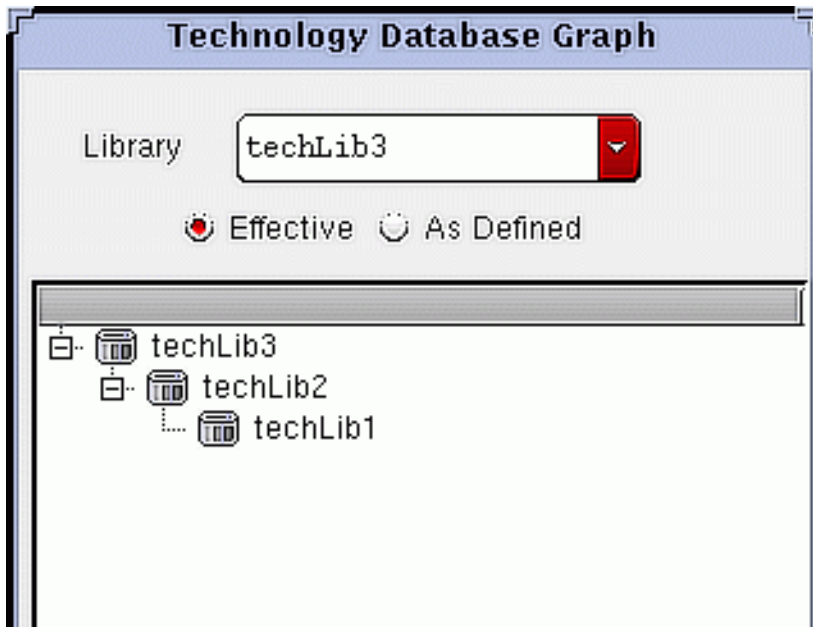
        spacings(
            ( minWidth    "Metall"    0.3 )
        ) ;spacings
    ) ;foundry
) ;constraintGroups

;*****
; SITEDEFS
;*****
siteDefs(
    scalarSiteDefs(
; ( siteDefName      type width  height  symInX symInY symInR90)
; ( -----          - - - - - - - - - - - - - - - - - - - - )
        ( core          core 10.0  5.0  t t nil)
        ( core2         core 20.0  5.0  t t nil)
        ( IO            pad  20.0  20.0 t t t)
    ) ;scalarSiteDefs

    arraySiteDefs(
; ( name      type
; ( ((siteDefName  dx      dy      orientation) ...)
; ( [symX] [symY] [symR90] )
; ( -----)
        ( tiledIO          pad
            (
                ( IO          0.0  0.0  R0  )
                ( core       0.0  0.0  R90 )
            )
            nil t t
        )
    ) ;arraySiteDefs
) ;siteDefs
```

## Graph of Incremental Technology Databases

Compiling the technology files in the proper order (`techLib1.tf` to database `techLib1`, then `techLib2.tf` to database `techLib2`, then `techLib3.tf` to database `techLib3`) results in the following technology database graph:



Other databases specifying other technology data can be added to the graph by referencing

Additional databases, referencing existing databases or providing additional technology data, can be added as needed.

## Example 2

This example presents six technology files that create separate databases for providing base data and data that can be applied to different tasks, such as different metal processes and place and route tasks. These illustrate the kind of data partitioning you can do with incremental technology databases.

The sample technology files are in the following directory:

```
<install_dir>/doc/techfileuser/examples
```

The following table summarizes the technology files, technology database names assumed by the references they contain, technology database references, and general contents. They are listed from the top of the graph down, as shown in the Technology Database Graph form. They are compiled from the bottom up.

Technology File	Technology Database	Technology Database References	Content
gpdk090_top.tf	gpdk090_top	techLibCustom4M prTechLib	database references only
techLibCustom4M.tf	techLibCustom4M	gpdk090	Data for 4-metal processes: viaDefs constraintGroups special setup constraint groups devices cdsViaDevices multipartPathTemplates
prTechLib.tf	prTechLib	gsclib090_tech techLib9M	Data specific to place and route tasks: controls additional techParams viaDefs constraintGroups LEFDefaultRouteSpec
gsclib090_tech.tf	gsclib090_tech	gpdk090	siteDefs

## Virtuoso Technology Data User Guide

### Incremental Technology File Examples

Technology File	Technology Database	Technology Database References	Content
techLib9M.tf	techLib9M	gpdK090	Data tailored for 9-metal processes: layerDefinitions layerRules viaDefs constraintGroups and constraints, including foundry constraints devices additional multipartPathTemplates
gpdK090.tf	gpdK090	(default reference to cdsDefTechLib added at compilation)	Base data: controls techParams mfgGridResolution layerDefinitions techPurposes techLayers techLayerPurpose-Priorities techDisplays layerRules functions routingDirections ViaDefs constraintGroups foundry devices ruleContactDevice multiPartPathTemplates

## Virtuoso Technology Data User Guide

### Incremental Technology File Examples

---

The following are the database graphs (As Defined and Effective) that result after loading the sample technology files in reference order to create the incremental technology databases.

