

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

**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>	5
<u>Scope</u>	6
<u>Licensing Requirements</u>	6
<u>Related Documentation</u>	6
<u>Installation, Environment, and Infrastructure</u>	6
<u>Virtuoso Tools</u>	6
<u>Virtuoso Tools</u>	8
<u>Additional Learning Resources</u>	8
<u>Video Library</u>	8
<u>Virtuoso Videos Book</u>	8
<u>Rapid Adoption Kits</u>	8
<u>Help and Support Facilities</u>	9
<u>Customer Support</u>	9
<u>Feedback about Documentation</u>	10
<u>Understanding Cadence SKILL</u>	11
<u>Using SKILL Code Examples</u>	11
<u>Sample SKILL Code</u>	11
<u>Accessing API Help</u>	12
<u>Typographic and Syntax Conventions</u>	13
<u>Identifiers Used to Denote Data Types</u>	14
 <u>1</u>	
<u>Import Functions</u>	17
<u>Verilog In Function</u>	18
<u>impHdlDisplay</u>	19
<u>VHDL In Functions</u>	20
 <u>2</u>	
<u>Connectivity-to-Schematic Functions</u>	21
<u>conn2Sch</u>	22

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

<u>conn2SchStartUp</u>	25
<u>conn2SchImpHdlDisplay</u>	26

Preface

This manual describes the SKILL APIs of tools used to import HDL files into the Virtuoso® design environment and convert netlists to schematic diagrams. It provides information on the SKILL functions that you can use with the following Virtuoso applications:

- Verilog In
- VHDL In
- Connectivity-to-Schematic

This manual is intended for the following users:

- Designers who want to use Verilog and VHDL files in designs maintained in the Virtuoso design environment.
- Designers who want to generate schematic views from netlist views.

This preface contains the following topics:

- [Scope](#)
- [Licensing Requirements](#)
- [Related Documentation](#)
- [Additional Learning Resources](#)
- [Customer Support](#)
- [Feedback about Documentation](#)
- [Understanding Cadence SKILL](#)
- [Typographic and Syntax Conventions](#)
- [Identifiers Used to Denote Data Types](#)

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node (for example, ICADVM18.1) releases.

Label	Meaning
(ICADVM18.1 Only)	Features supported only in the ICADVM18.1 advanced nodes and advanced methodologies release.
(IC6.1.8 Only)	Features supported only in mature node releases.

Licensing Requirements

For information on licensing in the Virtuoso design environment, see the [Virtuoso Software Licensing and Configuration User Guide](#).

Related Documentation

Installation, Environment, and Infrastructure

- [Cadence Installation Guide](#)
- [Virtuoso Design Environment SKILL Reference](#)
- [Cadence Application Infrastructure User Guide](#)
- [Virtuoso Software Licensing and Configuration Guide](#)

Virtuoso Tools

IC6.1.8 Only

- [Virtuoso Layout Suite L User Guide](#)
- [Virtuoso Layout Suite XL User Guide](#)
- [Virtuoso Layout Suite GXL Reference](#)

ICADVM18.1 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 ICADVM18.1

- [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](#)

- [Verilog In for Virtuoso Design Environment User Guide and Reference](#)
- [VHDL In for Virtuoso Design Environment User Guide and Reference](#)
- [Connectivity-to-Schematic User Guide](#)

Virtuoso Tools

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 the related features and to access the list of available videos, see [Virtuoso Videos](#).

Rapid Adoption Kits

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

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

- [SKILL Language Programming Introduction](#)
- [SKILL Language Programming](#)
- [Advanced SKILL Language Programming](#)

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

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

Help and Support Facilities

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

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

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

For more information, see [Getting Help](#) in *Virtuoso Design Environment User Guide*.

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

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

- Log on to Cadence Online Support

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

Feedback about Documentation

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

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

You can also submit feedback by using the following methods:

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

Understanding Cadence SKILL

Cadence SKILL is a high-level, interactive programming language based on the popular artificial intelligence language, Lisp. It lets you customize and extend your design environment. Using SKILL you can validate the steps of your algorithm incrementally before incorporating them into a larger program.

For more information about the SKILL language, see [Getting Started](#) in the *SKILL Language User Guide*.

Using SKILL Code Examples

The SKILL APIs in this user manual are explained with illustrative code examples.

You can copy these examples from the manual and paste them directly into the Command Interpreter Window (CIW) or use the code in non-graphical SKILL mode.

Sample SKILL Code

The following code sample shows the syntax of a SKILL API that accepts three arguments.

axlGetRunStatus

```
axlGetRunStatus(  
    t_sessionName      ← Required argument  
    [ ?optionName t_optionName ] ← Optional keyword argument  
    [ ?historyName t_historyName ] ← Optional keyword argument  
)  
=> l_statusValues      ← Return value
```

The first argument `t_sessionName` is a required argument, where `t` signifies the data type of the argument. The second and third arguments `?optionName t_optionName` and `?historyName t_historyName` are optional keyword arguments (identified by a question mark), which are specified in name-value pairs and can be placed in any order during the function call.

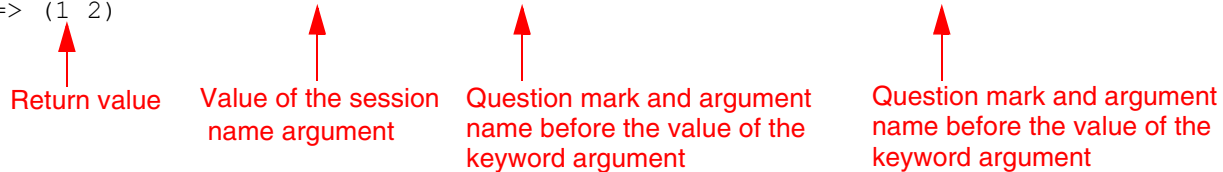
HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Preface

The return value is the value that the SKILL API returns after evaluating the expression. In this case, it is a list of status values, *l_statusValues*.

Example

```
axlSession=axlGetWindowSession( hiGetCurrentWindow() )
=> "session0"
axlGetRunStatus("session0" ?historyName "Interactive.10" ?optionName "tests")
=> (1 2)
```



Return value

Value of the session name argument

Question mark and argument name before the value of the keyword argument

Question mark and argument name before the value of the keyword argument

Accessing API Help

Quick reference information for SKILL APIs is available from the CIW and the SKILL API Finder. To access the reference information for a particular SKILL API, do one of the following:

- Type `help <function_name>` in the CIW.
- Type `startFinder ([?funcName t_functionName])` in the CIW.
- Start the **SKILL API Finder** from the CIW by choosing *Tools – Finder* or type `cdsFinder` on the UNIX command line.

In the *Search in* field of the displayed Cadence SKILL API Finder window, type the SKILL API name for which you want to display the help information and click *Go*.

The matches for the searched SKILL API appear in the *Results* area.

To view the complete documentation of the searched SKILL API, select the API name in the *Results* area and click the *More Info* button. The complete documentation of the selected SKILL API appears in a new Cadence Help window.

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.
<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.
<i>z_argument</i>	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, <i>z_</i>) indicates the data type the argument can accept and must not be typed.
	Separates a choice of options.
{ }	Encloses a list of choices, separated by vertical bars, from which you must choose one.
[]	Encloses an optional argument or a list of choices separated by vertical bars, from which you may choose one.
[?argName <i>t_arg</i>]	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.
...	Indicates that you can repeat the previous argument.
	Used with brackets to indicate that you can specify zero or more arguments.
	Used without brackets to indicate that you must specify at least one argument.
, ...	Indicates that multiple arguments must be separated by commas.
=>	Indicates the values returned by a Cadence® SKILL® language function.
/	Separates the values that can be returned by a Cadence SKILL language function.

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

Identifiers Used to Denote Data Types

Data type identifiers are used to indicate the type of value required by an API argument. These data types are denoted by a single letter that is prefixed to the argument label and is separated from the argument by an underscore; for example, t is the data type in `t_viewName`. Data types and underscores are used only as identifiers; they must not be typed when specifying the argument in a function.

Prefix	Internal Name	Data Type
<i>a</i>	array	array
<i>A</i>	amsobject	AMS object
<i>b</i>	ddUserType	DDPI object
<i>B</i>	ddCatUserType	DDPI category object
<i>C</i>	opfcontext	OPF context
<i>d</i>	dbobject	Cadence database object (CDBA)
<i>e</i>	envobj	environment
<i>f</i>	flonum	floating-point number
<i>F</i>	opffile	OPF file ID
<i>g</i>	general	any data type
<i>G</i>	gdmSpecIIUserType	generic design management (GDM) spec object
<i>h</i>	hdbobject	hierarchical database configuration object
<i>I</i>	dbgenobject	CDB generator object
<i>K</i>	mapioobject	MAPI object
<i>l</i>	list	linked list
<i>L</i>	tc	Technology file time stamp
<i>m</i>	nmplIIUserType	nmplII user type
<i>M</i>	cdsEvalObject	cdsEvalObject
<i>n</i>	number	integer or floating-point number
<i>o</i>	userType	user-defined type (other)
<i>p</i>	port	I/O port
<i>q</i>	gdmSpecListIIUserType	gdm spec list

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Preface

Prefix	Internal Name	Data Type
<i>r</i>	defstruct	defstruct
<i>R</i>	rodObj	relative object design (ROD) object
<i>s</i>	symbol	symbol
<i>S</i>	stringSymbol	symbol or character string
<i>t</i>	string	character string (text)
<i>T</i>	txobject	transient object
<i>u</i>	function	function object, either the name of a function (symbol) or a lambda function body (list)
<i>U</i>	funobj	function object
<i>v</i>	hdbpath	hdbpath
<i>w</i>	wtype	window type
<i>sw</i>	swtype	subtype session window
<i>dsw</i>	dswtype	subtype dockable window
<i>x</i>	integer	integer number
<i>y</i>	binary	binary function
<i>&</i>	pointer	pointer type

For more information, see *Cadence SKILL Language User Guide*.

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Preface

Import Functions

This chapter describes the SKILL functions used for importing

- Verilog In Function
- VHDL In Functions

Verilog In Function

Virtuoso Verilog In lets you convert structural Verilog netlists into one of the following forms:

- Virtuoso schematics
- Netlists in Cadence OpenAccess format
- Verilog text views in a Cadence-format library

In each case, the design is converted into a data format that can be used by Cadence tools. The following API lets you access the Verilog In tool.

For details on Verilog In, see the [Verilog In for Virtuoso Design Environment User Guide and Reference](#).

impHdlDisplay

```
impHdlDisplay(  
    impHdlOptionsForm  
)  
=> nil
```

Description

Displays the *Verilog In* form. Alternatively, in the CIW, choose *File – Import – Verilog* to open the *Verilog In* form.

Arguments

impHdlOptionsForm The *Verilog In* form name.

Values Returned

nil Returns *nil* on failure.

VHDL In Functions

Virtuoso VHDL In lets you convert a VHDL structural or behavioral description into one of the following forms:

- Schematic view
- Netlist view
- VHDL text views

In every case, VHDL In imports the design from the VHDL format into a Virtuoso database format—a data format that can be used by Cadence tools. You can import the following into a Virtuoso library:

- VHDL designs
- VHDL ASIC libraries
- VHDL designs, minus modules that already exist in the Virtuoso® Design Environment library
- Pieces of a hierarchical design
- A combination of the above architecture

For details, see the *VHDL In for Virtuoso Design Environment User Guide and Reference*.

You can use the following VHDL In functions, which are detailed in *Digital Design Netlisting and Simulation SKILL Reference*.

- vhdlHiImport
- vhdlImport
- vhdlToPinList
- vhdlPinListToVHDL
- vhdlRegisterSimulator

Connectivity-to-Schematic Functions

The Virtuoso Connectivity-to-Schematic tool is used to generate digital and analog schematic views from netlist views. For details, see the [*Connectivity-to-Schematic User Guide*](#).

Using the Connectivity-to-Schematic SKILL APIs, you can perform the following tasks:

- Generates a schematic view from an imported netlist view.
- Invokes graphical user interface of Connectivity-to-Schematic.
- Accept the user interface name of Connectivity-to-Schematic as an argument and displays the graphical user interface for the tool.

conn2Sch

```
conn2Sch(  
    t_srcLibName  
    t_srcCellName  
    t_srcViewName  
    [ ?destLibName t_destLibName ]  
    [ ?destCellName t_destCellName ]  
    [ ?destViewName t_destViewName ]  
    [ ?block t_block ]  
    [ ?paramFile t_paramFile ]  
    [ ?cdslib t_cdslib ]  
)  
=> t / nil
```

Description

Generates a schematic view from an imported netlist view.

The values of the following environment variables are taken from the `.cdsenv` file, by default. However, you can modify the values of these variables to customize the appearance of a schematic.

- `dest_symbol_view_name`
- `ref_lib_list`
- `log_file_name`
- `import_if_exists`
- `sheet_symbol`
- `line_line_spacing`
- `line_component_spacing`
- `density_level`
- `label_height`
- `page_col_limit`
- `page_row_limit`
- `pin_placement`
- `full_place_and_route`
- `optimize_wire_label_locn`

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Connectivity-to-Schematic Functions

- *minimize_crossovers*
- *generate_square_schematic*
- *extract_schematic*
- *verbose*
- *asg_options*
- *power_net*
- *ground_net*
- *power_symbol*
- *ground_symbol*

Arguments

<i>t_srcLibName</i>	Specifies the source library name.
<i>t_srcCellName</i>	Specifies the cell name to be imported.
<i>t_srcViewName</i>	Specifies the view name to be imported.
<i>?destLibName t_destLibName</i>	Specifies the destination library name. This is an optional argument.
<i>?destCellName t_destCellName</i>	Specifies the destination cell name. This is an optional argument.
<i>?destViewName t_destViewName</i>	Specifies the destination view name. This is an optional argument. The default value is <i>schematic</i> .
<i>?block t_block</i>	Specifies if conn2Sch needs to wait until import is complete. This is an optional argument. The default value is <i>nil</i> .
<i>?paramFile t_paramFile</i>	Specifies the parameter file. This is an optional argument. The default value is <i>nil</i> .
<i>?cdslib t_cdslib</i>	Specifies the cdslib file. This is an optional argument. The default value is <i>nil</i> .

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Connectivity-to-Schematic Functions

Values Returned

<code>t</code>	Returns <code>t</code> on successful execution of the function.
<code>nil</code>	Returns <code>nil</code> if the function encounters an error.

Example

```
conn2Sch("testLib" "testCell" "netlist")
```

Additional Information

- If you do not specify the *dest_sch_lib*, *dest_cell_name*, or *dest_view_name*, or if these parameters are empty or set to `nil`, then the values for these parameters are taken from the `.cdsenv` file. If the values for these parameters in the `.cdsenv` file are invalid, then the schematic is saved in the specified *src_sch_lib* with the specified *src_cell_name*.
- If you specify a parameter file, `conn2sch` ignores the *log_file_name* parameter, if specified in the file. The tool saves the errors and log messages in the file specified using the *log_file_name* environment variable. If you set the *log_file_name* environment variable to `" "`, its default value, `conn2sch.log`, is used.

conn2SchStartUp

```
conn2SchStartUp(  
    )  
=> t / nil
```

Description

Invokes GUI of the Connectivity-to-Schematic tool. This is a CIW menu callback function.

Arguments

None

Values Returned

<code>t</code>	Returns <code>t</code> if the GUI of the tool is invoked.
<code>nil</code>	Returns <code>nil</code> if the function encounters an error while invoking the GUI of the tool.

conn2SchImpHdlDisplay

```
conn2SchImpHdlDisplay(  
    conn2schOptionsForm  
)  
=> t / nil
```

Description

Accepts the user interface name of Connectivity-to-Schematic as an argument and displays the GUI for the tool.

Arguments

conn2schOptionsForm

Refers to the user interface of Connectivity-to-Schematic. The user interface accepts user inputs and other optional settings.

Values Returned

<i>t</i>	Returns <i>t</i> if the GUI of the tool is invoked.
<i>nil</i>	Returns <i>nil</i> if the function encounters an error while invoking the GUI of the tool.

Note: The functions, `conn2SchImpHdlDisplay()` and `conn2SchStartUp()` perform the same task. The `conn2SchImpHdlDisplay()` function will be removed in the next release.