

Virtuoso® ADE Verifier SKILL Reference

**Product Version ICADV20.1
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>	11
<u>Scope</u>	11
<u>Licensing Requirements</u>	12
<u>Related Documentation</u>	12
<u>What's New and KPNS</u>	12
<u>Installation, Environment, and Infrastructure</u>	12
<u>Technology Information</u>	12
<u>Virtuoso Tools</u>	13
<u>Additional Learning Resources</u>	13
<u>Video Library</u>	13
<u>Virtuoso Videos Book</u>	13
<u>Rapid Adoption Kits</u>	14
<u>Help and Support Facilities</u>	14
<u>Customer Support</u>	15
<u>Feedback about Documentation</u>	15
<u>Understanding Cadence SKILL</u>	16
<u>Using SKILL Code Examples</u>	16
<u>Sample SKILL Code</u>	16
<u>Accessing API Help</u>	17
<u>Typographic and Syntax Conventions</u>	18
<u>Identifiers Used to Denote Data Types</u>	19

1

<u>Requirement Functions</u>	21
<u>Addition and Deletion Functions</u>	22
<u>verifAddReq</u>	23
<u>verifMoveReq</u>	25
<u>verifRemoveReq</u>	27
<u>Manual Signoff Functions</u>	28
<u>verifDeleteReqSignoff</u>	29
<u>verifGetReqSignoff</u>	31

Virtuoso ADE Verifier SKILL Reference

<u>verifSignOffReq</u>	32
<u>Data Extraction Functions</u>	34
<u>verifGetReqs</u>	35
<u>verifGetReqParent</u>	37
<u>verifGetReqProp</u>	38
<u>verifGetReqProps</u>	39
<u>verifGetReqStatus</u>	40
<u>verifGetReferencedCellViews</u>	41
<u>Requirement Setup Functions</u>	42
<u>verifCreateRandomId</u>	43
<u>verifSetReqId</u>	44
<u>verifSetReqTitle</u>	45
<u>verifSetReqType</u>	46
<u>verifSetReqCellviewHolder</u>	48
<u>verifSetReqProp</u>	50
<u>Export and Import Functions</u>	51
<u>verifExportReqsToFile</u>	52
<u>verifGetImportedFiles</u>	54
<u>verifImportFile</u>	55
<u>verifCompareImportedFiles</u>	57
<u>verifMergeImportedFiles</u>	58
<u>verifExportJson</u>	59
<u>Custom Fields Functions</u>	60
<u>verifSetCustomFieldValue</u>	61
<u>verifSetReqCustomFieldValue</u>	63
<u>verifGetCustomFieldNames</u>	65
<u>verifGetCustomFieldValue</u>	66
<u>verifGetReqCustomFieldNames</u>	67
<u>verifGetReqCustomFieldValue</u>	68
<u>vManager Functions</u>	70
<u>verifCreateVPlan</u>	71
<u>verifCreateVsifScript</u>	73
<u>verifDownloadFromVManager</u>	75
<u>verifGetVManager</u>	77
<u>verifGetVManagerProjects</u>	79
<u>verifIsVManagerConnected</u>	80

Virtuoso ADE Verifier SKILL Reference

<u>verifIsVManagerEnabled</u>	81
<u>verifPostResultsToVManager</u>	82
<u>verifRemoveVManager</u>	84
<u>verifSetVManager</u>	86
<u>verifUploadToVManager</u>	89
<u>verifVPlanExists</u>	90

2

Verifier Session and Setup Functions 93

<u>Batch Functions</u>	94
<u>verifCreateBatchScript</u>	95
<u>verifIsBatchRunProcess</u>	96
<u>Startup, Exit, and Session Information Functions</u>	97
<u>verifOpenCellView</u>	98
<u>verifIsSessionReadOnly</u>	100
<u>verifSaveSession</u>	101
<u>verifSaveSessionAs</u>	102
<u>verifCloseSession</u>	103
<u>verifGetAllSessions</u>	104
<u>verifGetCellViewSession</u>	105
<u>verifIsSessionModified</u>	106
<u>verifIsValidSession</u>	107
<u>verifGetWindow</u>	108
<u>verifGetSessionCellView</u>	109
<u>verifRegisterCallback</u>	110
<u>verifRemoveCallback</u>	113
<u>verifGetCallbacks</u>	114
<u>Preferences Functions</u>	115
<u>verifGetOptions</u>	116
<u>verifGetOptionVal</u>	118
<u>verifSetOptionVal</u>	119

3

Implementation Functions 121

Implementation Management Functions 122

Virtuoso ADE Verifier SKILL Reference

<u>verifAddImp</u>	123
<u>verifMoveImp</u>	125
<u>verifRemoveImp</u>	127
<u>verifOverwriteSpec</u>	129
<u>verifGetImpEstRunTime</u>	131
<u>verifSetImpEstRunTime</u>	132
<u>verifGetImpPriority</u>	134
<u>verifSetImpPriority</u>	135
<u>verifUpdateImpEstRunTime</u>	137
<u>verifGetImpSetPreRunScript</u>	139
<u>verifSetImpSetPreRunScript</u>	140
<u>verifGetMappableType</u>	142
<u>Data Extraction Functions</u>	144
<u>verifGetImps</u>	145
<u>verifGetImpData</u>	148
<u>verifGetImpTestOutputs</u>	151
<u>verifGetImpTests</u>	153
<u>verifUpdate</u>	155

4

Requirement-to-Implementation Mapping Functions..... 157

<u>verifMapping</u>	158
<u>verifGetImpMapping</u>	162
<u>verifGetReqMapping</u>	164
<u>verifExportMapping</u>	165
<u>verifImportMapping</u>	166

5

Simulation and Results Extraction Functions..... 169

<u>Implementation Set Functions</u>	170
<u>verifAddImpSet</u>	171
<u>verifAddImpToImpSet</u>	172
<u>verifGetImpSets</u>	174
<u>verifGetImpsInImpSet</u>	176
<u>verifSetImpSetName</u>	177

Virtuoso ADE Verifier SKILL Reference

<u>verifRemoveImpFromImpSet</u>	178
<u>verifRemoveImpSet</u>	180
<u>Results Extraction Functions</u>	181
<u>verifGetResultDataForImp</u>	182
<u>verifGetResultDataForReq</u>	185
<u>verifReloadAllRes</u>	187
<u>verifEvaluateResults</u>	188
<u>verifCopyAndUpdateResultsFromUserDefinedDirectory</u>	189
<u>Simulation Functions</u>	191
<u>verifImplsRun</u>	192
<u>verifRun</u>	193
<u>verifSetImpRun</u>	195
<u>verifCheck</u>	196
<u>verifCheckImp</u>	199
<u>verifStop</u>	201

6

<u>Verification Status and Reports Functions</u>	203
<u>verifPublishHTML</u>	204

7

<u>Setup Library Assistant Functions</u>	205
<u>slaOpenOrCreateView</u>	207
<u>slaAddCornerModelFile</u>	209
<u>slaAddCornerVariable</u>	210
<u>slaAddDocument</u>	211
<u>slaAddSweepVariable</u>	212
<u>slaCreateCornerSetup</u>	213
<u>slaCreateSweepSetup</u>	214
<u>slaCreateVerificationSpace</u>	215
<u>slaGetAllDocuments</u>	216
<u>slaGetCornerModels</u>	217
<u>slaGetCornerSetupCorners</u>	218
<u>slaGetCornerSetups</u>	219
<u>slaGetCornerVars</u>	220

Virtuoso ADE Verifier SKILL Reference

<u>slaGetDocumentAbsolutePath</u>	221
<u>slaGetSweepSetupVars</u>	222
<u>slaGetSweepSetups</u>	223
<u>slaGetVerificationSpaces</u>	224
<u>slaImportCorners</u>	225
<u>slaImportSweeps</u>	226
<u>slaRemoveCorner</u>	227
<u>slaRemoveCornerModel</u>	228
<u>slaRemoveCornerSetup</u>	229
<u>slaRemoveCornerVariable</u>	230
<u>slaRemoveSweepSetup</u>	231
<u>slaRemoveSweepVariable</u>	232
<u>slaRemoveVerificationSpace</u>	233
<u>slaSaveAndCloseView</u>	234

8

Snapshot Functions 237

<u>verifAreSnapshotsEnabled</u>	239
<u>verifCreateSnapshot</u>	240
<u>verifCreateSnapshotConfiguration</u>	242
<u>verifDeleteSnapshot</u>	244
<u>verifDeleteSnapshotConfiguration</u>	245
<u>verifExportSnapshotsToExcel</u>	246
<u>verifGetReferenceSnapshot</u>	248
<u>verifGetSnapshot</u>	249
<u>verifGetSnapshotAbsoluteTolerance</u>	250
<u>verifGetSnapshotComment</u>	251
<u>verifGetSnapshotRelativeTolerance</u>	252
<u>verifGetSnapshots</u>	253
<u>verifGetSnapshotsData</u>	254
<u>verifIsSnapshotLocked</u>	256
<u>verifIsSnapshotVisible</u>	257
<u>verifRenameSnapshot</u>	258
<u>verifRestoreFromSnapshot</u>	259
<u>verifSetReferenceSnapshot</u>	260

Virtuoso ADE Verifier SKILL Reference

<u>verifSetSnapshotAbsoluteTolerance</u>	261
<u>verifSetSnapshotComment</u>	262
<u>verifSetSnapshotConfiguration</u>	263
<u>verifSetSnapshotLocked</u>	264
<u>verifSetSnapshotRelativeTolerance</u>	265
<u>verifSetSnapshotVisible</u>	266
<u>verifSetSnapshotsEnabled</u>	267

9

<u>Debugging Functions</u>	269
<u>verifEnableDebug</u>	270
<u>verifDisableDebug</u>	272
<u>verifGetDebug</u>	274

A

<u>Additional Information</u>	277
<u>Custom Function to Copy Implementation Units to Requirements</u>	278
<u>Callback Function to Save Verifier Cellviews Automatically</u>	280
<u>Callback Function to Automatically Create Snapshots for Each Simulation Run</u>	281
<u>Callback Function to Automatically Create Snapshots for Backup</u>	282
<u>Function to Customize Menu Banners</u>	284
<u>Function to Customize Context Menus</u>	285
<u>Removed SKILL functions</u>	289

Virtuoso ADE Verifier SKILL Reference

Preface

This manual describes the SKILL functions of Cadence® Virtuoso® ADE Verifier (Verifier).

Note: Only the functions and arguments described in this manual are supported for public use. Any undocumented functions or arguments are likely to be private and could be subject to change without notice. It is recommended that you check with your Cadence representative before using them.

This SKILL API reference is meant for verification project managers and designers who want to use Verifier SKILL APIs for requirements-based verification of their analog designs.

This manual assumes that users are familiar with the Cadence SKILL language and Virtuoso Verifier.

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 mature node, advanced nodes, and advanced methodologies releases.

Licensing Requirements

Verifier is available with the Virtuoso_ADE_Verifier license (95270) in the ICADV12.3 and higher releases.

Appropriate licenses are required to simulate implementations from Verifier. These include the licenses for the following:

- Virtuoso ADE Assembler
- Virtuoso ADE Explorer
- A supported simulator for simulating the designs

Related Documentation

What's New and KPNS

- [*Virtuoso ADE Verifier What's New*](#)
- [*Virtuoso ADE Verifier Known Problems and Solutions*](#)

Installation, Environment, and Infrastructure

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

Technology Information

- [*Virtuoso Technology Data User Guide*](#)
- [*Virtuoso Technology Data ASCII Files Reference*](#)
- [*Virtuoso Technology Data SKILL Reference*](#)

Virtuoso Tools

- [*Virtuoso ADE Verifier User Guide*](#)
- [*Tutorial: Virtuoso ADE Verifier Workflow*](#)
- [*Virtuoso ADE Assembler User Guide*](#)
- [*Virtuoso ADE Explorer User Guide*](#)
- [*Virtuoso Schematic Editor User Guide*](#)
- [*Virtuoso UltraSim Simulator User Guide*](#)
- [*Spectre AMS Designer Simulator User Guide*](#)
- [*Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide*](#)
- [*Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*](#)
- [*Spectre Circuit Simulator Reference*](#)

Additional Learning Resources

Video Library

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

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

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

Virtuoso Videos Book

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

Rapid Adoption Kits

Cadence provides 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.

Virtuoso ADE Verifier Rapid Adoption Kit

In addition, Cadence offers the following training courses on ADE Verifier:

- [Virtuoso ADE Verifier](#)
- [Virtuoso ADE Explorer S1: Set Up and Run Analog Simulations Using the Spectre Simulator](#)
- [Virtuoso ADE Explorer S2: Analyzing Simulations Using the ViVA XL Waveform Tool](#)
- [Virtuoso ADE Explorer S3: Corner Analysis and Monte Carlo Simulations](#)
- [Virtuoso ADE Explorer S4: Real-Time Tuning, Checks/Asserts, and Reliability Analysis](#)
- [Virtuoso ADE Assembler S1: Introducing the Assembler Environment](#)
- [Virtuoso ADE Assembler S2: Sweeping Variables, Simulating Corners, and Creating Run Plans](#)
- [Virtuoso ADE Assembler S3: Circuit Checks, Device Asserts, and Reliability Analysis](#)
- [Variation Analysis Using the Virtuoso ADE Assembler](#)

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.

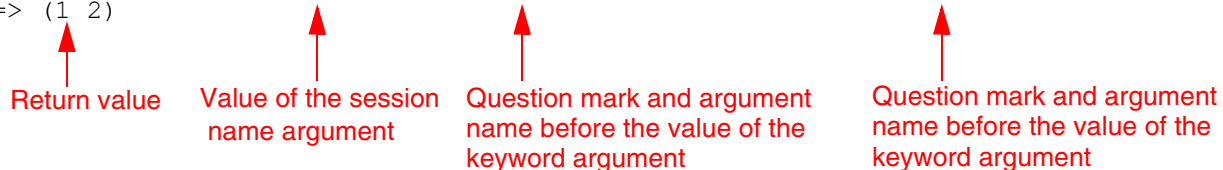
Virtuoso ADE Verifier SKILL Reference

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.
text	Indicates text that you must type 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 t_arg]	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, τ is the data type in $\tau_viewName$. 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

Virtuoso ADE Verifier 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>d_w</i>	dwtype	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*.

Requirement Functions

This chapter describes the following SKILL functions:

- Addition and Deletion Functions
- Manual Signoff Functions
- Data Extraction Functions
- Requirement Setup Functions
- Export and Import Functions
- Custom Fields Functions
- vManager Functions

Addition and Deletion Functions

Use the following functions to add and delete requirements:

- verifAddReq
- verifMoveReq
- verifRemoveReq

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifAddReq

```
verifAddReq(  
    g_sessionId  
    t_title  
    [ ?reqId t_reqId ]  
    [ ?parentId t_parentId ]  
    [ ?pos x_pos ]  
)  
=> t / nil
```

Description

Adds a new requirement with the title to the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_title</i>	Title of the requirement.
<i>?reqId t_reqId</i>	Unique ID of the new requirement.
<i>?parentId t_parentId</i>	ID of an existing requirement that will be the parent of the new requirement.
<i>?pos x_pos</i>	Position where the requirement will be inserted. By default, the new requirement is added at the end.

Value Returned

<i>t</i>	Requirement is added to the Verifier session.
<i>nil</i>	Requirement is not added to the Verifier session.

Examples

The following example opens a Verifier cellview and adds a requirement:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=>0
```

The following example creates a requirement with the given title:

```
verifAddReq(sess "full_diff_opamp")
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

=>t

The following example creates a requirement with ID "C1" under requirement "ID1"):

```
verifAddReq(sess "A child req" ?reqId "C1" ?parentId "ID1")
```

=>t

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifMoveReq

```
verifMoveReq(  
    g_sessionId  
    t_reqId  
    [ ?parentReqId g_parentReqId ]  
    [ ?itemIndex x_itemIndex ]  
)  
=> t / nil
```

Description

Changes the position of a requirement in the specified Verifier session.

Arguments

g_sessionId Integer, string number, or window specifying the Verifier session ID.
For example, 0, "0", or window(2).

t_reqId ID of the requirement to be moved.

?parentReqId *g_parentReqId*

g_parentReqId accepts one of the following values:

- nil - Parent does not change (default)
- " " - Moves the requirement specified with *t_reqId* to the top-level design so that the given *g_parentReqId* is no longer the parent
- ID of a requirement with type "Note" - The specified requirement is moved to be a child of this *Note* requirement.

?itemIndex *x_itemIndex*

New position of the requirement. If the index is 0 (default) the requirement is moved to the bottom.

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Value Returned

t	The requirement was moved successfully.
nil	The requirement was not moved successfully.

Example

Open Verifier cellview and move requirement with ID "ID1.1" to be the first child of requirement "ID1".

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifMoveReq(sessionId "ID1.1" ?parentReqId "ID1" ?itemIndex 1)
=> t
```

verifRemoveReq

```
verifRemoveReq(  
    g_sessionId  
    g_reqId  
)  
=> t / nil
```

Description

Deletes one single requirement or a list of requirements in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_reqId</i>	The single requirement ID or list of requirement IDs.

Value Returned

t	Specified requirements are deleted from the Verifier session.
nil	Specified requirements are not deleted from the Verifier session.

Examples

The following example opens a Verifier cellview and deletes the specified requirements.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0
```

Delete a list of requirements:

```
verifRemoveReq(uid list("ID1.1" "ID1.2"))  
=> t
```

Delete a requirement:

```
verifRemoveReq(uid "ID1")  
=> t
```

Manual Signoff Functions

Use the following functions to manually sign off requirements:

- verifDeleteReqSignoff
- verifGetReqSignoff
- verifSignOffReq

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifDeleteReqSignoff

```
verifDeleteReqSignoff(  
    g_sessionId  
    [ ?reqId g_reqId ]  
)  
=> t / nil
```

Description

Deletes the signoff setup of the specified requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>?reqId g_reqId</i>	Single requirement ID or list of requirement IDs.

Value Returned

<i>t</i>	Signoff setup of the specified requirement is deleted.
<i>nil</i>	There is no signoff information to delete, or the command failed.

Examples

The following example opens a Verifier cellview and retrieves a requirement that has the verification type manual. The example then retrieves the sign-off details of that requirement (ID1). Then, it removes the sign-off and checks its removal.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetReqSignoff(uid "ID1")  
(nil signoffUser "Tom" signoffTime "Jun 14 15:25:59 2018"  
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"  
signoffComments  
"Manully passing this requirement until next run."  
)
```

Deletes a signoff setup for a single requirement:

```
verifDeleteReqSignoff(uid ?reqId "ID1")  
=> t  
verifGetReqSignoff(uid "ID1")  
=> nil
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Deletes signoff setup for a list of requirements:

```
verifDeleteReqSignoff(uid ?reqId list("ID1" "ID2"))  
=> t
```

Deletes signoff setup for all requirements:

```
verifDeleteReqSignoff(uid)  
=> t
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifGetReqSignoff

```
verifGetReqSignoff(  
    g_sessionId  
    t_reqId  
)  
=> l_reqSignoff / nil
```

Description

Retrieves the signoff setup of the specified requirement in a Verifier session. The retrieved information includes the name of the person who signed off the signoff validity period, comments, and if the signoff is enabled for use or disabled.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of requirement.

Value Returned

<i>l_reqSignoff</i>	Disembodied property list (DPL) containing the sign-off setup of the specified requirement.
<i>nil</i>	Specified requirement does not have any sign-off setup.

Examples

The following example opens a Verifier cellview and retrieves a requirement that has the signoff setup:

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0
```

Retrieves the disembodied property list of signoff data for the specified requirement:

```
dpl = verifGetReqSignoff(uid "ID1")  
(nil signoffUser "Tom" signoffTime "Jun 14 15:25:59 2018"  
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"  
signoffComments  
"Manully passing this requirement until next run."  
)
```

Retrieves the value for 'signoffUser':

```
dpl->signoffUser  
"Tom"
```

verifSignOffReq

```
verifSignOffReq(
    g_sessionId
    [ ?reqId g_reqId ]
    [ ?userName t_userName ]
    [ ?lifetime t_lifetime ]
    [ ?expDate t_expDate ]
    [ ?comments t_comments ]
    [ ?disableState g_disableState ]
)
=> t / nil
```

Description

Signs off the specified requirements using the provided information in a Verifier session. You can sign off the requirements that have the verification type 'Manual' or the requirements that failed verification (the overall status of requirement is 'Fail', 'Not Mapped', 'Spec Check Fail', 'Signoff Required' or 'Signed Off').

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>?reqId g_reqId</i>	Single requirement ID or list of requirement IDs. Default is nil and it signs off all requirements. Note: The requirement that cannot be signed off will be ignored.
<i>?userName t_userName</i>	Name of the user who is signing off the specified requirements.
<i>?lifetime t_lifetime</i>	Validity period of the signoff. The supported values are: <ul style="list-style-type: none"> ■ Once: Signoff is valid until next run. ■ Until Date: Signoff is valid up to the date specified using the option <i>?expiration t_expiration</i>. ■ Indefinitely: Signoff is valid for an infinite time.
<i>?expDate t_expDate</i>	String value indicating the expiration date of the signoff. Use it to specify the value of <i>t_lifetime</i> as 'Until Date'. Specify the date format 'MMM d yyyy', such as 'Jan 9 2018' and ensure that the specified date is not earlier than the current date.
<i>?comments t_comments</i>	Any comment about the manual signoff.

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

<code>?disableState</code> <code>g_disableState</code>	Status indicating if the signoff is enabled or disabled for use in the verification project.
---	--

Value Returned

<code>t</code>	Specified requirements are signed off.
<code>nil</code>	Specified requirements are not signed off.

Examples

Opens a Verifier cellview and then signs off the requirements.

```
uid = verifOpenCellView("test" "results" "verifier")
=>0
verifGetReqSignoff(uid "ID1")
=>nil
```

Signs off a requirement.

```
verifSignOffReq(uid ?reqId "ID1" ?comments "Manually passing this requirement until
next run")
=> t
verifGetReqSignoff(uid "ID1")
(nil signoffUser "Tom" signoffTime "Jun 14 16:07:56 2018"
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"
signoffFullUserName
"" signoffExpireDate "" signoffComments "Manually passing this requirement until
next run"
)
```

Signs off a list of requirements.

```
verifSignOffReq(uid ?reqId list("ID1" "ID2"))
=> t
verifGetReqSignoff(uid "ID1")
(nil signoffUser "Tom" signoffTime "Jun 14 16:09:31 2018"
signoffLocation "/ADE1/users/harry/tst/EAV_RAK" signoffLifetime "Once"
signoffFullUserName
"" signoffExpireDate "" signoffComments "")
)
```

Signs off all requirements that can be signed off.

```
verifSignOffReq(uid)
=> t
```

Data Extraction Functions

Use the following functions to extract data from requirements:

- verifGetReqs
- verifGetReqParent
- verifGetReqProp
- verifGetReqProps
- verifGetReqStatus
- verifGetReferencedCellViews

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifGetReqs

```
verifGetReqs(  
    g_sessionId  
    [ ?type t_type ]  
    [ ?owner t_owner ]  
    [ ?inHier t_parentReqId ]  
    [ ?mappable g_mappable ]  
    [ ?mapped g_mapped ]  
    [ ?domain g_domain ]  
)  
=> l_reqs / nil
```

Description

Retrieves the list of requirements from a Verifier session that have the same type, owner, hierarchy, mappable or mapped property, and domain as specified.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
?type <i>t_type</i>	Type of requirement.
?owner <i>t_owner</i>	Owner of requirement.
?inHier <i>t_parentReqId</i>	ID of parent requirement.
?mappable <i>g_mappable</i>	Boolean value that specifies if the requirement must be mappable.
?mapped <i>g_mapped</i>	Boolean value that specifies if the requirement must be mapped to the implementation items.
?domain <i>g_domain</i>	String value that specifies the domain of the requirements.

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Value Returned

<i>l_reqs</i>	A list of requirements, all or specific ones that match the specified criteria.
<i>nil</i>	There are no requirements in the Verifier session or no requirements match the conditions.

Examples

The following example opens a Verifier cellview and retrieves the list of the requirements from the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Get all requirements:

```
verifGetReqs(uid)
("ID1" "ID2" "ID3" "ID3.1" "ID3.2")
```

Get requirements that has type of 'Spec Pass':

```
verifGetReqs(uid ?type "Spec Pass")
=> ("ID3")
```

Get unmapped requirement:

```
verifGetReqs(uid ?mapped nil)
=> ("ID2" "ID3" "ID3.1")
```

Get requirements that has type of 'Ran Ok', belongs to 'Tom', has parent 'ID3', is mappable and mapped:

```
verifGetReqs(uid ?type "Ran Ok" ?owner "Tom" ?inHier "ID3" ?mappable t ?mapped t
?domain "SIMULATION")
=> ("ID3.2")
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifGetReqParent

```
verifGetReqParent(  
    g_sessionId  
    t_reqId  
)  
=> t_parentReqId / nil
```

Description

Returns the ID of the parent of the specified requirement.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of the requirement.

Value Returned

<i>t_parentReqId</i>	ID of the parent of the specified requirement ID.
<i>nil</i>	ID of the parent of the specified requirement ID is not returned.

Example

The following example opens a Verifier cellview and gets the ID for the parent of the specified requirement.

```
sessionId=verifOpenCellView("test" "setup" "verifier")  
=> 0  
verifGetReqParent(sessionId "ID_1.1")  
=> "ID_1"
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifGetReqProp

```
verifGetReqProp(  
    g_sessionId  
    t_reqId  
    t_propName  
)  
=>t_propValue / nil
```

Description

Returns the value of the specified property of a requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of the requirement.
<i>t_propName</i>	Name of the requirement property. The valid property names are Parent, HierId, ID, Title, Type, MinSpec, MaxSpec, Unit, Owner, ReadOnly, Description, and VerificationSpace.

Value Returned

<i>t_propValue</i>	Specified property of a requirement in a Verifier session.
<i>nil</i>	Specified property of a requirement in a Verifier session was not found.

Example

The following example opens a Verifier cellview and retrieves the MinSpec of a requirement.

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
verifGetReqProp(1 "ID_1" "MinSpec")  
=> "0.98"
```

verifGetReqProps

```
verifGetReqProps(  
    g_sessionId  
    t_reqId  
)  
=>o_reqProps / nil
```

Description

Returns all the properties of a requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of the requirement.

Value Returned

<i>o_reqProps</i>	Specified properties of a requirement in a Verifier session.
<i>nil</i>	Specified properties of a requirement in a Verifier session were not found.

Example

The following example opens a Verifier cellview and retrieves all properties for the requirement "ID1_1":

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 0  
tableToList(verifGetReqProps(sessionId "ID_1.1"))  
(("Unit" "")  
 ("Description" "")  
 ("MinSpec" "0.98")  
 ("ID" "ID_1.1")  
 ("Parent" "ID_1")  
 ("MaxSpec" "")  
 ("Title" "Swing")  
 ("Type" "Spec Pass")  
 ("Owner" "Tom")  
 ("ReadOnly" "false")  
 ("VerificationSpace" "")  
 ("HierId" "1.1")  
)
```

verifGetReqStatus

```
verifGetReqStatus(  
    g_sessionId  
    t_reqId  
)  
=> t_status / nil
```

Description

Returns the overall status of a requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of the requirement.

Value Returned

<i>t_status</i>	Overall status of the specified requirement ID. This is the text on <i>Overall Status</i> column of the <i>Results</i> tree.
<i>nil</i>	Overall status of the specified requirement ID is not returned.

Example

The following example opens a Verifier cellview and retrieves the overall status of the specified requirements.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetReqStatus(uid "ID1")  
=> "0%"  
verifGetReqStatus(uid "ID1.1")  
=> "Not Mapped"
```


verifGetReferencedCellViews

```
verifGetReferencedCellViews(  
    g_sessionId  
)  
=> l_referencedCellViews | nil
```

Description

Returns a list of all the referenced Verifier cellviews that can be associated with the specified session. If no referenced Verifier cellviews are defined, nil is returned.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>l_referencedCellViews</i>	List of all referenced Verifier cellviews that can be associated with the specified session.
nil	No referenced Verifier cellviews have been defined.

Example

The following example opens a Verifier cellview and retrieves the referenced Verifier cellviews from the specified session:

```
uid = verifOpenCellView("test" "ref" "verifier")  
0  
verifGetReferencedCellViews(uid)  
(("test" "ref" "verifier_OpAmp")  
 ("test" "ref" "verifier_amsPLL")  
)
```

Requirement Setup Functions

Use the following functions to set requirements:

- verifCreateRandomId
- verifSetReqId
- verifSetReqTitle
- verifSetReqType
- verifSetReqCellviewHolder
- verifSetReqProp

verifCreateRandomId

```
verifCreateRandomId(  
    [ ?idLength x_idLength ]  
    [ ?prefix t_prefix ]  
    [ ?suffix t_suffix ]  
)  
=> t_id
```

Description

Creates a string of random characters that can be used as a unique requirement ID.

Arguments

<code>?idLength</code> <code>x_idLength</code>	The length of the generated portion of the ID. This does not include the given prefix or the suffix. Default is 5.
<code>?prefix t_prefix</code>	The prefix of generated IDs.
<code>?suffix t_suffix</code>	The suffix of generated IDs.

Value Returned

<code>t_id</code>	Generated random ID.
-------------------	----------------------

Examples

In your .cdsinit file, specify the following:

```
envSetVal("verifier.requirement" "idCreationFunction" 'string  
"verifCreateRandomId")
```

Newly created requirements will have an ID of five random characters, such as NUX8L or pWnT8.

The following example opens a Verifier cellview and creates a random ID for a new requirement:

```
verifCreateRandomId()  
=> "pWnT8"
```

The following example opens a Verifier cellview and creates a random ID for a new requirement with the prefix "id_" and a suffix "_x":

```
verifCreateRandomId(?prefix "id_" ?suffix "_x")  
=> "id_hhVGj_x"
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifSetReqId

```
verifSetReqId(  
    g_sessionId  
    t_reqId  
    t_newReqId  
)  
=> t / nil
```

Description

Sets the ID property for the specified requirement.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	Requirement ID.
<i>g_newReqId</i>	New requirement ID.

Value Returned

t	Requirement ID has been updated to <i>t_newReqId</i> .
nil	Requirement ID has not been updated to <i>t_newReqId</i> because of errors.

Example

The following example opens a Verifier cellview and sets the ID of a requirement to "ID4".

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
verifSetReqId(sessionId "ID3" "ID4")  
=> t
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifSetReqTitle

```
verifSetReqTitle(  
    g_sessionId  
    t_reqId  
    t_newTitle  
)  
=> t / nil
```

Description

Sets the ID property for the specified requirement.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_reqId</i>	Requirement ID.
<i>t_newTitle</i>	New title for the requirement.

Value Returned

<i>t</i>	Requirement title has been updated to <i>t_newTitle</i> .
<i>nil</i>	Requirement title has not been updated to <i>t_newTitle</i> because of errors.

Example

The following example opens a Verifier cellview and sets the title of a requirement to "DC gain".

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
verifSetReqTitle(sessionId "ID3" "DC gain")  
=> t
```

verifSetReqType

```
verifSetReqType(  
    g_sessionId  
    t_redId  
    t_newType  
)  
=> t / nil
```

Description

Sets the 'Type' property for the specified requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_redId</i>	The requirement ID.
<i>t_newType</i>	<p>The value of the requirement verification type. The type value can be one of the following:</p> <ul style="list-style-type: none">■ <code>Note</code> for notes■ <code>Spec Pass</code> for specification checks■ <code>Ran Ok</code> for simulation run status■ <code>Manual</code> for manual sign-off■ <code>ExtRef</code> for external references

Value Returned

<i>t</i>	Requirement type has been updated to <i>t_newType</i> .
<i>nil</i>	Requirement type has not been updated to <i>t_newType</i> because of errors.

Example

The following example opens a Verifier cellview and changes the type of specified requirement:

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
uid = verifOpenCellView("test" "setup" "verifier")
=> 0
verifSetReqType(uid "ID3" "Manual")
=> t
```

verifSetReqCellviewHolder

```
verifSetReqCellviewHolder(  
    g_sessionId  
    t_reqId  
    g_owned  
)  
=> t / nil
```

Description

If *g_owned* is specified, sets the 'Cellview Holder' property of specified requirement to current Verifier cellview, else clears the value of the property.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of the requirement.
<i>g_owned</i>	Owned state of the specified requirement.

Value Returned

<i>t</i>	Sets 'Cellview Holder' of requirement to current cellview or clears 'Cellview Holder' successfully.
<i>nil</i>	vManager is disabled or vManager is not setup in the specified Verifier session.

Example

The following example starts a Verifier session with a cellview and sets or clears the 'Cellview Holder' property of the specified requirement:

```
uid = verifOpenCellview("test" "example" "verifier")  
=>0  
verifSetReqCellviewHolder(uid "ID1" t)  
=>t  
verifGetReqProp(uid "ID1" "cvHolder")  
"test example verifier"  
verifSetReqCellviewHolder(uid "ID1" nil)  
=>t  
verifGetReqProp(uid "ID1" "cvHolder")
```


Virtuoso ADE Verifier SKILL Reference

Requirement Functions

'''

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifSetReqProp

```
verifSetReqProp(  
    g_sessionId  
    t_reqId  
    t_propertyName  
    t_propertyValue  
)  
=> t / nil
```

Description

Sets the property for the specified requirement.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	Requirement ID.
<i>t_propertyName</i>	Name of the property to be set.
<i>t_propertyValue</i>	New value of property.

Value Returned

t	Property value was set.
nil	Property value was not set because of errors.

Example

The following example opens a Verifier cellview and sets the type to Note:

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
verifSetReqProp(sessionId "ID3" "type" "Note")  
=> t
```

Export and Import Functions

Use the following functions to export and import requirements:

- verifExportReqsToFile
- verifGetImportedFiles
- verifImportFile
- verifCompareImportedFiles
- verifMergeImportedFiles
- verifExportJson

verifExportReqsToFile

```
verifExportReqsToFile(  
    g_sessionId  
    t_fileName  
    [ ?fileType t_fileType ]  
    [ ?fields t_fields ]  
)  
=> t / nil
```

Description

Exports the requirements to either an Excel or CSV file. Can also export the results imported file.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	The name of the file to export to.
<i>?fileType</i> <i>t_filetype</i>	A case insensitive string from: CSV, Excel or Imported. The default is "CSV".
<i>?fields</i> <i>t_fields</i>	The list of requirement fields to be exported. These fields can contain the following the case-insensitive names, listed in the default sequence: Overall Status, Mapping, History, Min Value, Max Value, Result Data Age, Passed, Failed, Not Run, Disabled. The requirement is always exported by default and cannot be disabled.

Value Returned

<i>t</i>	Successfully exported the requirements.
<i>nil</i>	Failed to export the requirements.

Example

The following example shows how to use this function to exports requirements:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 1
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
verifExportReqsToFile(sess "reqs.csv")  
=> t  
verifExportReqsToFile(sess "reqs.xlsx" ?fileType "Excel")  
=> t  
verifExportReqsToFile(sess "reqs2.csv" ?fields "Overall Status,Mapping,Unmapped")  
=> t
```

verifGetImportedFiles

```
verifGetImportedFiles(  
    g_sessionId  
)  
=> l_importedFiles / nil
```

Description

Returns a list of all imported files that are associated with the specified session. If no imported files have been defined, `nil` is returned.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
--------------------	--

Value Returned

<i>l_importedFiles</i>	List of all imported files that are associated with the specified session.
<code>nil</code>	No imported files are defined in the specified session.

Example

The following example shows how the function can be used:

```
sessionId = verifOpenCellView("test" "export" "verifier")  
=> 0  
verifGetImportedFiles(sessionId)  
=> ("./test_export.csv")
```

verifImportFile

```
verifImportFile(  
    g_sessionId  
    t_fileName  
    [ ?fileType t_fileType ]  
    [ ?headerRows x_headerRows ]  
    [ ?columnHeaders l_columnHeaders ]  
    [ ?sheetNames l_sheetNames ]  
    [ ?ignoreInavaliidRows g_ignoreInavaliidRows ]  
)  
=>t / nil
```

Description

Imports the requirements from a CSV or Excel file into a Verifier session. You can import requirements by referencing them or by copying them from the file.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	The name of the Excel or CSV file.
<i>t_fileType</i>	The type of the file, that can be "CSV" or "Excel".
?headerRows <i>x_headerRows</i>	The number of the row which contains the column header definitions.
?columnHeaders <i>l_columnHeaders</i>	The list of the header column names that identify the kind of data being loaded from the file and their sequence. The list can contain the following names: Parent, ID, Title, Type, MinSpec, MaxSpec, Unit, Owner, Description. Note: Do not use this optional key argument if you specify ?headerRows <i>x_headerRow</i> .
?sheetName <i>l_sheetName</i>	A list of the names of the Excel sheets to be imported.
?ignoreInavaliidRows <i>g_ignoreInavaliidRows</i>	Ignores the rows containing invalid properties. The default value is nil.

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Value Returned

t	The requirements are imported from the specified file successfully.
nil	The file contains errors or there are no valid requirements in the file.

Example

The following example opens a Verifier cellview and imports the requirements from a CSV file.

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifImportFile(sessionId "test.csv" ?fileType "CSV" ?headerRows 1
?ignoreInvalidRows t)
=> t
```


verifCompareImportedFiles

```
verifCompareImportedFiles(  
    g_sessionId  
    l_files  
)  
=> t / nil
```

Description

Compares the requirements from the Verifier session with the requirements from the imported files.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>l_files</i>	The list of files to be compared. If no files are specified, all imported files are compared.

Value Returned

t	The comparison is successful and the differences are displayed.
nil	There specified imported files do not exist in Verifier session or the command is unsuccessful.

Example

The following example opens a Verifier cellview and compares the requirements in the session with the imported requirements from a CSV file:

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 0  
  
verifCompareImportedFile(sessionId "test.csv" ?fileType "CSV" ?headerRows 1  
?ignoreInvalidRows t)  
=> t
```

verifMergeImportedFiles

```
verifMergeImportedFiles(  
    g_sessionId  
    l_files  
    [ ?interactive g_interactive ]  
)  
=> t / nil
```

Description

Merges the requirements from the Verifier session with the requirements from the imported files. If differences exist, the *Compare and Merge* form is displayed.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>l_files</i>	The list of files to be merged. If no files are specified, all imported files are merged.
<i>?interactive</i> <i>g_interactive</i>	<i>g_interactive</i> accepts one of the following values: <ul style="list-style-type: none">■ <i>nil</i> - Requirements are merged(default)■ <i>t</i> - Displays the <i>Compare and Merge</i> form if there are differences.■ The default value is <i>nil</i>.

Value Returned

<i>t</i>	Files are merged successfully.
<i>nil</i>	There are no imported files defined in the specified session.

Example

The following example opens a Verifier cellview and merges the requirements in the session with the imported requirements from a CSV file:

```
list1 = list("./csv/111.csv")  
verifMergeImportedFiles("0" list1)  
=> t
```

verifExportJson

```
verifExportJson(  
    g_sessionId  
    [ ?filename t_filename ]  
    [ ?compact g_compact ]  
)  
=> t_filename / nil
```

Description

Exports the Verifier session as a `json` file. The `json` file contains details about the session, including the requirements, implementations, mappings, and simulation results.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>?filename</i> <i>t_filename</i>	Name of the file to export to. If no filename is given, the details are exported to a file in the format <code><lib>_<cell>_<view>.json</code> .
<i>?compact</i> <i>g_compact</i>	Exports the <code>json</code> file in compact format if <code>nil</code> . Otherwise, it is indented and printed over multiple lines.

Value Returned

<i>t_filename</i>	Name of the file that was created.
<i>nil</i>	The command is not successful or an error occurred.

Example

The following example shows how to use this function:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 0  
  
verifExportJson(sess)  
=> "/home/user/dir1/test_sample_verifier.json"  
  
verifExportJson(sess ?filename "v1.json")  
=> "/home/user/dir1/v1.json"  
  
verifExportJson(sess ?filename "v2.json" ?compact t)  
=> "/home/user/dir1/v2.json"
```

Custom Fields Functions

Use the following functions to work with custom fields in requirements:

- [verifSetCustomFieldValue](#)
- [verifSetReqCustomFieldValue](#)
- [verifGetCustomFieldNames](#)
- [verifGetCustomFieldValue](#)
- [verifGetReqCustomFieldNames](#)
- [verifGetReqCustomFieldValue](#)

Overview

Verifier lets you include additional information for your design verification project, and for each requirement in the project. For example, you can include the name and e-mail address of the verification project manager. For each requirement in the project, you can include the name and e-mail address of the engineer as well.

To use this feature, first define the custom field tags, names, and tool tips in a `.csv` file for the project and for the requirements. See the following examples:

The `infoProject.csv` file contains the custom field details for the project:

```
Manager,Manager Name,The name of the project manager.  
Email,E-mail Address,The e-mail address of the project manager.
```

The `infoReq.csv` file contains the custom field details for the requirements in the project:

```
Engineer,Engineer Name,The name of the lead engineer.  
Email,E-mail Address,The e-mail address of the engineer.
```

Set the Verifier environment variable `customFieldConfig` and `customReqFieldConfig` to indicate the `.csv` files so that Verifier can start using the custom fields. For example, you can set these variables in the `.cdsinit` file, as illustrated below.

```
envSetVal("verifier.customFieldConfig" "csvfile" 'string' "./infoProject.csv")  
envSetVal("verifier.customReqFieldConfig" "csvfile" 'string' "./infoReq.csv")
```

verifSetCustomFieldValue

```
verifSetCustomFieldValue(  
    g_sessionId  
    t_fieldName  
    t_fieldValue  
)  
=> l_set / nil
```

Description

Sets the value of a custom field in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fieldName</i>	Name of the custom field.
<i>t_fieldValue</i>	Value of the custom field.

Value Returned

<i>l_set</i>	Value of the custom field is successfully set to the new value.
<i>nil</i>	Value of the custom field is not set.

Example

Consider that you have defined the custom fields *Manager Name* and *E-mail Address* for a verification project, as indicated by the following figure. For details on defining custom fields, see [“Overview”](#).

Note: To access the Virtuoso ADE Verifier Preferences form from the Verifier graphical user interface, choose *Edit — Preferences*.

The following example opens a Verifier cellview of the project and sets the values in the custom fields.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetCustomFieldNames(uid)  
=> ("Email" "VerificationManager")  
verifSetCustomFieldValue(uid "VerificationManager" "Harry")
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
=> t  
verifGetCustomFieldValue(uid "VerificationManager")  
=> "Harry"
```

verifSetReqCustomFieldValue

```
verifSetReqCustomFieldValue(  
    g_sessionId  
    t_reqId  
    t_fieldName  
    t_fieldValue  
)  
=> t / nil
```

Description

Sets the custom field value of the specified requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_reqId</i>	The ID of the requirement.
<i>t_fieldName</i>	The name of the custom field.
<i>t_fieldValue</i>	The value of the custom field.

Value Returned

<i>t</i>	The value of the custom field is successfully set to the new value.
<i>nil</i>	The value of the custom field is not set.

Example

Consider that you have defined the custom fields *Engineer Name* and *E-mail Address* for the requirements of a verification project. For details on defining custom fields, see [“Overview”](#).

Note: To access the requirement editor form from the Verifier graphical user interface, select the requirement and choose *Edit — Open Requirement Editor*.

The following example opens a Verifier cellview of the verification project and sets the values in the custom fields for the specified requirement.

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
verifGetReqCustomFieldNames(uid)
=> ("Email" "Engineer")
verifGetReqCustomFieldValue(uid "ID1" "Engineer")
=> ""
verifSetReqCustomFieldValue(uid "ID1" "Engineer" "Harry")
=> t
verifGetReqCustomFieldValue(uid "ID1" "Engineer")
=> "Harry"
```


verifGetCustomFieldNames

```
verifGetCustomFieldNames(  
    g_sessionId  
)  
=> l_fieldList / nil
```

Description

Retrieves a list of custom field names for a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>l_fieldList</i>	List of the custom fields available to the specified Verifier session.
nil	There are no custom fields in the specified Verifier session, or the command failed.

Example

The following example opens a Verifier cellview and retrieves the custom fields available to the session.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetCustomFieldNames(uid)  
=> ("Email" "VerificationManager")
```

verifGetCustomFieldValue

```
verifGetCustomFieldValue(  
    g_sessionId  
    t_fieldName  
)  
=> l_fieldvalue / nil
```

Description

Retrieves the value of a custom field available in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fieldName</i>	String specifying the name of the custom field.

Value Returned

<i>l_fieldvalue</i>	List of the value of the specified custom field in the session.
nil	There is no custom field in the session, or the command failed.

Example

The following example opens a Verifier cellview and retrieves the value of the custom field:

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetCustomFieldValue(uid "Email")  
=> tom@cadence.com"
```

verifGetReqCustomFieldNames

```
verifGetReqCustomFieldNames(  
    g_sessionId  
)  
=> l_fieldList / nil
```

Description

Retrieves the list of the custom fields available to the requirements in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>l_fieldList</i>	List of the custom fields available to the requirements in the Verifier session.
nil	There are no custom fields available to the requirements in the Verifier session, or the command failed.

Example

The following example opens a Verifier session and retrieves the names of the custom field:

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetCustomFieldNames(uid)  
=> ("Email" "VerificationManager")
```

verifGetReqCustomFieldValue

```
verifGetReqCustomFieldValue(  
    g_sessionId  
    t_reqId  
    t_fieldName  
)  
=> l_fieldValue / nil
```

Description

Retrieves the value of the specified custom field of a requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	ID of the requirement.
<i>t_fieldName</i>	String name of the custom field for requirements.

Value Returned

<i>l_fieldValue</i>	List value of the specified custom field of the requirement.
nil	Specified custom field of the requirement does not have a value, or the command failed.

Examples

The following example opens a Verifier cellview and retrieves the value of the custom field for a requirement.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetReqCustomFieldNames(uid)  
=> ("Email" "Engineer")
```

Custom field with empty value:

```
verifGetReqCustomFieldValue(uid "ID1" "Email")  
=> ""
```

Custom field with specified value:

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
verifGetReqCustomFieldValue(uid "ID1" "Engineer")  
=> "Fred"
```

Non-existent custom field name:

```
verifGetReqCustomFieldValue(uid "ID1" "Project")  
*WARNING* (VERIFIER-1239): unknown custom property name 'Project' on requirement  
'ID1'  
=> nil
```

vManager Functions

Use the following functions to use the vManager features:

- verifCreateVPlan
- verifCreateVsifScript
- verifDownloadFromVManager
- verifGetVManager
- verifGetVManagerProjects
- verifIsVManagerConnected
- verifIsVManagerEnabled
- verifPostResultsToVManager
- verifRemoveVManager
- verifSetReqCellviewHolder
- verifSetVManager
- verifUploadToVManager
- verifVPlanExists

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifCreateVPlan

```
verifCreateVPlan(  
    g_sessionId  
    t_fileName  
)  
=> t / nil
```

Description

Creates a vPlan file (*.vplanx) for specified Verifier session that is connected to the vManager server.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	vPlan file name.

Value Returned

<i>t</i>	vPlan file is created successfully.
<i>nil</i>	vPlan file exists, or vManager server does not have the permissions to write to that file, or the command is unsuccessful.

Example

The following example starts a Verifier session with vManager setup and creates a vPlan file:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080  
APB_UART_noref.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '1'.  
=>0  
verifIsVManagerEnabled(uid)  
=>t  
verifCreateVPlan(uid "/tmp/myPlan.vplanx")  
=>t  
verifVPlanExists(uid "/tmp/myPlan.vplanx")  
=>t  
verifCreateVPlan(uid "/home/User/myPlan.vplanx")  
*WARNING* (VERIFIER-9001): An error occurred during 'Create Plan':  
'Error transferring https://vmgr:letmein@vm-verifier-team:8080/vmgr/vapi/rest/  
planning/create - server replied: Bad Request  
Failed to open file for writing /home/User/myPlan.vplanx'
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Fix the error and try again.
=>nil

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifCreateVsifScript

```
verifCreateVsifScript(  
    g_sessionId  
    t_vsifFileName  
    [t_batchFileName]  
)  
=> t / nil
```

Description

Creates a Verification Simulation Input File (*.vsif) and batch script file for the specified Verifier session that is connected to the vManager server. This allows running Verifier from vManager.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_vsifFileName</i>	VSIF file name.
<i>t_batchFileName</i>	Batch script file name that is used to launch Verifier and run simulations. If this value is not defined, the VSIF basename with "_run" appended is used.

Value Returned

t	VSIF and batch script files are created successfully
nil	File exists, or Verifier does not have the permissions to write to the file, or the command is unsuccessful.

Example

The following example starts a Verifier session with vManager setup and creates the required files:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080  
APB_UART noref.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
verifCreateVsifScript(uid "myRun.vsif")  
=>t  
verifCreateVsifScript(uid "scripts/myRun.vsif")
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
;;Creates myRun.vsif and myRun_run
*WARNING* (VERIFIER-5009): Failed to open file 'scripts/myRun_run' for writing.
Check the file permissions and try again.
nil
```

verifDownloadFromVManager

```
verifDownloadFromVManager(  
    g_sessionId  
    ?g_waitUntilDone waitUntilDone  
    ?t_action action  
)  
=> t / nil
```

Description

Downloads the data from specified vPlan file through vManager server.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_waitUntilDone</i>	Ensures that the specified Verifier session waits for the download to complete before proceeding further. This is enabled by default.
<i>t_action</i>	Action for downloading data from vPlan in Verifier. The possible actions are merging the requirements, mapping and implementations from vManager to those in Verifier or replacing them. Specifying 'Ask' displays a dialog to ask what to do. The default value is 'Merge'. Possible values are 'Ask', 'Merge', and 'Replace'.

Value Returned

<i>t</i>	The data is downloaded from the specified vPlan file through vManager server successfully.
<i>nil</i>	The command is not successful.

Example

The following example starts a Verifier session with vManager setup and downloads data:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080  
APB_UART_noref.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

;; The vPlan file is modified and downloads the data

```
verifDownloadFromVManager(uid)
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080
APB_UART_noref.vplanx.
=>t
```

verifGetVManager

```
verifGetVManager(  
    g_sessionId  
    [g_addDetails]  
)  
=> l_settings / nil
```

Description

Retrieves the vManager settings from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_addDetails</i>	If enabled, the 'rootName' and 'versionSupported' will be shown up in the DPL. By default, only retrieve the basic settings: username, password, project, hostname, port, timeout, vPlan, enabled, version, configured, connected.

Value Returned

<i>l_settings</i>	DPL for the settings of vManager in the specified Verifier session.
<i>nil</i>	vManager is not setup in the specified Verifier session.

Example

The following example starts a Verifier session with vManager setup and retrieves the vManager settings:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
verifGetVManager(uid)  
(nil username "vmgr" password "****"  
project "vmgr" hostname "vm-verifier-team" port  
8080 timeout 30 vPlan "/tmp/myPlan.vplanx"  
enabled t version "19.01-a001" configured  
t connected t  
)
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
verifGetVManager(uid t)
(nil versionSupported t rootName "APB_UART"
username "vmgr" password "****" project
"vmgr" hostname "vm-verifier-team" port 8080
timeout 30 vPlan "/tmp/myPlan.vplanx" enabled
t version "19.01-a001" configured t
connected t
)
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifGetVManagerProjects

```
verifGetVManagerProjects(  
    g_sessionId  
)  
=> l_projects / nil
```

Description

Retrieves a list of projects from vManager server that are configured in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>l_projects</i>	List of projects from vManager that are configured in the specified Verifier session.
<i>nil</i>	vManager is not setup in the specified Verifier session or the command is not successful.

Example

The following example starts a Verifier session with vManager setup and gets the projects from vManager sever:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
verifGetVManagerProjects(uid)  
("vmgr")
```

verifIsVManagerConnected

```
verifIsVManagerConnected(  
    g_sessionId  
)  
=> t / nil
```

Description

Checks if the Verifier session is connected to the vManager server.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	Verifier is connected to the vManager server.
nil	Verifier is not connected to the vManager server, or vManager is not fully configured, or the command is not successful.

Example

The following example starts a Verifier session with a cellview and checks if Verifier is connected to the vManager server:

```
uid = verifOpenCellView("test" "example" "verifier")  
0  
verifIsVManagerConnected(uid)  
*WARNING* (VERIFIER-9018): verifIsVManagerConnected - no vManager setup specified  
for session 0  
Use menu entry Tools->vManager->Setup vManager to configure vManager and try again  
nil
```


verifIsVManagerEnabled

```
verifIsVManagerEnabled(  
    g_sessionId  
)  
=> t / nil
```

Description

Checks if vManager is enabled in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	vManager is enabled.
nil	vManager is disabled or vManager is not setup in the specified Verifier session.

Example

The following example starts a Verifier session with vManager setup and checks if vManager is enabled or not:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
;; Return the check state of 'Enabled' on vManager setup form  
verifIsVManagerEnabled(uid)  
=>t
```

verifPostResultsToVManager

```
verifPostResultsToVManager(  
    g_sessionId  
    [t_vmSessionName]  
)  
=> t / nil
```

Description

Posts the verification results of a simulation to the specified vManager session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_vmSessionName</i>	Name of the vManager session that contains all the verification and simulation results for each implementation in Verifier. The default value is verifier_session_<lib>_<cell>_<view>_<history>_<username>_<time>.

Value Returned

<i>t</i>	Post verification results to the new vManager session successfully.
<i>nil</i>	vManager is not setup in the specified Verifier session or the command is not successful.

Example

The following example starts a Verifier session with vManager setup and post results:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
verifPostResultsToVManager(uid)  
*WARNING* (VERIFIER-9014): The implementation 'amsPLL/PLL_VCO_320MHZ_tb/maestro/  
Active' has not yet been run by Verifier.  
Therefore, no results can be posted to vManager.  
INFO (VERIFIER-9010): Posting results to  
verifier_session_amsPLL_TOP_verification_verification_User_2019_03_04_22_15_20_24
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
10.  
INFO (VERIFIER-9011): Successfully post results to  
verifier_session_amsPLL_TOP_verification_verification_User_2019_03_04_22_15_20_24  
10.  
=>t
```

verifRemoveVManager

```
verifRemoveVManager(  
    g_sessionId  
)  
=> t / nil
```

Description

Remove all the vManager settings from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>t</i>	All the vManager settings are removed successfully.
<i>nil</i>	The command is not successful.

Example

The following example starts a Verifier session with a cellview and remove the vManager settings:

```
uid = verifOpenCellView("test" "example" "verifier")  
=>0  
verifSetVManager(uid ?enabled t ?project "vmgr")  
*WARNING* (VERIFIER-9019): vManager setup for session 0 is not fully configured.  
Use menu entry Tools->vManager->Setup vManager to fully configure vManager and try  
again.  
=>t  
verifGetVManager(uid)  
(nil error "(VERIFIER-9019): vManager setup for session 0 is not fully  
configured.\nUse menu entry Tools->vManager->Setup vManager to fully configure  
vManager and try again." username ""  
password "" project "vmgr" hostname  
"" port 0 timeout 30  
vPlan "" enabled t version  
"" configured nil connected nil  
)  
verifRemoveVManager(uid)  
=>t  
verifGetVManager(uid)  
*WARNING* (VERIFIER-9018): verifGetVManager - no vManager setup specified for
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
session 0
Use menu entry Tools->vManager->Setup vManager to configure vManager and try again
=>nil
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

verifSetVManager

```
verifSetVManager(  
    g_sessionId  
    ?username g_username  
    ?password g_password  
    ?hostname g_hostname  
    ?port g_port  
    ?project g_project  
    ?timeout g_timeout  
    ?vPlan g_vPlan  
    ?enabled g_enabled  
    ?action t_action  
)  
=> t / nil
```

Description

Configures or updates the vManager settings within the specified Verifier session.

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_username</i>	Specifies the user name.
<i>g_password</i>	Specifies the password.
<i>g_hostname</i>	Specifies the host name.
<i>g_port</i>	Specifies the port number.
<i>g_project</i>	Specifies the project name.
<i>g_timeout</i>	Specifies the timeout duration.
<i>g_vPlan</i>	Specifies the vPlan file name.
<i>g_enabled</i>	State of 'Enabled'.
<i>t_action</i>	Action for downloading data from vPlan in Verifier. The possible actions are merging the requirements, mapping and implementations from vManager to those in Verifier or replacing them. Specifying 'Ask' displays a dialog to ask what to do. The default value is 'Merge'. Possible values are 'Ask', 'Merge', and 'Replace'.

Value Returned

<i>t</i>	Configured or modified the vManager settings successfully.
<i>nil</i>	Command is unsuccessful.

Example

The following example starts a Verifier session with vManager setup and modifies the vManager settings:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
INFO (VERIFIER-8215): Started Verifier session '0'.
=>0
verifSetVManager(uid ?enabled nil)
=>t
verifIsVManagerEnabled(uid)
nil
;; Creates a new Verifier cellview and configures the vManager settings
win = deNewCellView("amsPLL" "TOP_verification" "verifier" "verifier" nil)
INFO (VERIFIER-8215): Started Verifier session '2'.
window:4
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

```
uid = win->verifSession
=>2
verifSetVManager(uid ?username "vmgr" ?password "****" ?hostname "vm-verifier-team"
?port 8080 ?vPlan "/tmp/myPlan.vplanx" ?enabled t)
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
=>t
;; Updates an existing setup
verifSetVManager(uid ?username "new-user" ?password "newPassword")
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/
myPlan.vplanx.
=>t
```


verifUploadToVManager

```
verifUploadToVManager(  
    g_sessionId  
    [g_waitUntilDone]  
)  
=> t / nil
```

Description

Upload the Verifier data to the specified vPlan file using the vManager server.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_waitUntilDone</i>	Ensures that the specified Verifier session must wait for the upload to complete before proceeding further. This is enabled by default.

Value Returned

<i>t</i>	Data is uploaded to specified vPlan file through vManager server successfully.
<i>nil</i>	Command is not successful.

Example

The following example starts a Verifier session with vManager setup and uploads data:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
;; The requirements or mappings are modified in specified Verifier session  
verifUploadToVManager(uid)  
INFO (VERIFIER-9005): Uploading session to vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-9006): Successfully uploaded session to vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
=>t
```

verifVPlanExists

```
verifVPlanExists(  
    g_sessionId  
    t_fileName  
)  
=> t / nil
```

Description

Checks if the vPlan file exists and can be accessed by the vManager configured in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	vPlan file name.

Value Returned

<i>t</i>	Specified vPlan file exists and is available.
<i>nil</i>	Specified Verifier session does not contain vManager setup or the command is not successful.

Example

The following example starts a Verifier session with vManager setup and checks for the vPlan file:

```
uid = verifOpenCellView("amsPLL" "TOP_verification" "verification")  
INFO (VERIFIER-9008): Successfully downloaded from vm-verifier-team.8080 /tmp/  
myPlan.vplanx.  
INFO (VERIFIER-8215): Started Verifier session '0'.  
=>0  
verifVPlanExists(uid "/tmp/myPlan.vplanx")  
=>t  
verifVPlanExists(uid "myPlan.vplanx")  
=>nil
```

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Virtuoso ADE Verifier SKILL Reference

Requirement Functions

Verifier Session and Setup Functions

This chapter describes the following SKILL functions:

- Batch Functions
- Startup, Exit, and Session Information Functions
- Preferences Functions

Batch Functions

Use the following function to create batch scripts in Verifier:

- verifCreateBatchScript
- veriflsBatchRunProcess

verifCreateBatchScript

```
verifCreateBatchScript(  
    g_sessionId  
    t_scriptFileName  
)  
=> t | nil
```

Description

Creates a batch script to rerun the saved Verifier cellview from the command line.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_scriptFileName</i>	The batch script file name.

Value Returned

t	The batch script file is created successfully.
nil	The command is not successful.

Example

The following example opens a Verifier cellview and create the batch script file for the specified session:

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifCreateBatchScript(uid "run")  
=> t
```

To use the script:

```
virtuoso -replay run -log run.log
```

verifIsBatchRunProcess

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

Description

Returns `t` if the code is currently running in a remote child process for ADE Verifier. You can use this function in your `.cdsinit` file or in custom SKILL code.

Arguments

None.

Value Returned

<code>t</code>	Child IC remote process is running.
<code>nil</code>	Child IC remote process is not running.

Example

The following example shows how to use this function:

;In Virtuoso CIW:

```
axlIsICRPPProcess( )  
=> nil
```

;In the .cdsinit file:

```
if(verifIsBatchRunProcess() then  
    info("## Is batch run process ##\n")  
    ;; Do something for batch run here  
else  
    info("## Is not batch run process ##\n")  
    ;; Do something for non-batch run here  
)
```


Startup, Exit, and Session Information Functions

Use the following functions to start and exit Verifier:

- [verifOpenCellView](#)
- [verifIsSessionReadOnly](#)
- [verifSaveSession](#)
- [verifSaveSessionAs](#)
- [verifCloseSession](#)
- [verifGetAllSessions](#)
- [verifGetCellViewSession](#)
- [verifIsSessionModified](#)
- [verifIsSessionModified](#)
- [verifGetWindow](#)
- [verifGetSessionCellView](#)
- [verifRegisterCallback](#)
- [verifRemoveCallback](#)
- [verifGetCallbacks](#)

verifOpenCellView

```
verifOpenCellView(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?mode g_mode ]  
    [ ?openWindow g_openWindow ]  
)  
=> x_sessionId / nil
```

Description

Opens the Verifier cellview and returns the session ID.

Arguments

<i>t_libName</i>	The library name of the cellview.
<i>t_cellName</i>	The cell name.
<i>t_viewName</i>	The view name.
<i>?mode g_mode</i>	<p>The mode in which the cellview is opened. The available options are:</p> <p><i>nil</i>: The session will be opened either for editing or as read-only depending on whether the cellview is editable or not.</p> <p><i>r</i>: The session will be opened as read-only.</p> <p><i>w</i>: The session will be opened with write-access. This will create a new cellview if it does not exist, or overwrite an existing cellview.</p> <p><i>a</i>: The session will be opened for edit.</p>
<i>?openWindow g_openWindow</i>	Boolean to specify whether to open a window in the cellview or not. If it is <i>nil</i> then no window is opened, but the session is still opened in non-graphical mode. The default value is <i>t</i> .

Virtuoso ADE Verifier SKILL Reference

Verifier Session and Setup Functions

Value Returned

<code>x_sessionId</code>	The session ID of the opened cellview.
<code>nil</code>	The specified cellview does not open successfully.

Examples

The following example opens a Verifier cellview in a window:

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
```

If a window is already open then brings the window to the front and returns the session ID:

```
sess2 = verifOpenCellView("test" "sample" "verifier")
=> 0
```

In the Design Management flow, if a window is already open, the function returns `nil`. No window is opened, but the session is still opened in non-graphical mode.

```
sess2 = verifOpenCellView("test" "sample" "verifier")
=> nil
```

Creates a new cellview:

```
sess3 = verifOpenCellView("test" "sample" "verifier2" ?mode "w")
=> 1
verifSaveSession(sess3)
verifCloseSession(sess3)
```

Reopens the cellview for edit without a window:

```
sess3 = verifOpenCellView("test" "sample" "verifier2" ?mode "a" ?openWindow nil)
=> 2
```

verifIsSessionReadOnly

```
verifIsSessionReadOnly(  
    g_sessionId  
)  
=> t / nil
```

Description

Determine if the given session is read-only.

Argument

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	The session is open read-only.
nil	The session is open for editing or does not exist.

Example

The following example shows how to use this function:

```
sess = verifOpenCellView("test" "sample" "verifier" ?mode "r")  
=> 1  
verifIsSessionReadOnly(sess)  
=> t  
verifIsSessionReadOnly(99)  
*WARNING* (VERIFIER-5004): verifIsSessionReadOnly - no such session: 99  
=> nil
```

verifSaveSession

```
verifSaveSession(  
    g_sessionId  
)  
=> t / nil
```

Description

Saves the active setup of the specified Verifier session to the cellview from where the setup was loaded.

Argument

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	Successfully saved the session.
nil	Failed to save the session.

Example

The following example shows how to save a Verifier session in its cellview:

```
sess = verifOpenCellView("test" "setup" "verifier")  
=> 0  
verifSaveSession(sess)  
INFO (VERIFIER-1507): Saved cellview test.setup:verifier  
=> t
```

verifSaveSessionAs

```
verifSaveSessionAs(  
    g_sessionId  
    t_libName  
    t_cellName  
    t_viewName  
)  
=> t / nil
```

Description

Saves the active setup of specified Verifier session as a cellview of the type 'verifier'.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_libName</i>	The library name - this must already exist
<i>t_cellName</i>	The new cell name
<i>t_viewName</i>	The new view name

Value Returned

t	Session was successfully saved as a new cellview.
nil	Failed to save the cellview.

Example

The following example shows how to save a Verifier session to a new cellview:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 1  
verifSaveSessionAs(sess "test" "sample" "verifier_new")  
=> t
```

verifCloseSession

```
verifCloseSession(  
    g_sessionId  
    [ ?saveIfModified g_saveIfModified ]  
)  
=> t / nil
```

Description

Closes a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>[?saveIfModified g_saveIfModified]</i>	A boolean which controls whether to save the session, or discard any changes if it is modified. If any value is not specified and the session has been modified, then a question dialog prompts to either save or discard the changes.

Value Returned

t	Specified Verifier session was closed successfully.
nil	Specified Verifier session was not opened or the command was unsuccessful.

Example

The following example starts a Verifier cellview and adds a requirement. It then saves the cellview and closes the session using `verifCloseSession`.

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 0  
verifCloseSession(sess) ; Closes the unmodified session  
sess = verifOpenCellView("test" "sample" "verifier")  
=> 1  
verifAddReq(0 "A Requirement") ; Now modified it by adding a requirement  
=> t  
verifCloseSession(sess ?saveIfModified t) ; Will save then close the session - if  
it was ?saveIfModified nil, then the changes would be discarded before closing  
=> t
```

verifGetAllSessions

```
verifGetAllSessions(  
    )  
=> l_ids / nil
```

Description

Returns a list of all the open Verifier session Ids.

Value Returned

<i>l_ids</i>	List of integers, string numbers, or windows specifying the Verifier session IDs.
<i>l_ids</i>	Failed to find any open sessions.

Example

The following example opens a Verifier cellview and returns all open session IDs:

```
sess = verifOpenCellView("test" "setup" "verifier")  
=> 0  
verifGetAllSessions()  
=> (0)
```


verifGetCellViewSession

```
verifGetCellViewSession(  
    t_libName  
    t_cellName  
    t_viewName  
)  
=> x_sessionId / nil
```

Description

Returns the integer session ID for the specified opened Verifier cellview.

Arguments

<i>t_libName</i>	Library name for the opened Verifier cellview.
<i>t_cellName</i>	Cell name for the opened Verifier cellview.
<i>t_viewName</i>	View name for the opened Verifier cellview.

Value Returned

<i>x_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>nil</i>	Specified Verifier cellview does not open.

Example

The following example retrieves the session ID of a Verifier cellview.

```
verifOpenCellView("test" "setup" "verifier")  
=> 1  
sessionId = verifGetCellViewSession("test" "setup" "verifier")  
=> 1
```

verifIsSessionModified

```
verifIsSessionModified(  
    g_sessionId  
)  
=> t / nil
```

Description

Checks whether the setup has been modified after the last time it was saved in the given session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>t</i>	The setup of the specified session has been modified after the last time it was saved.
<i>nil</i>	The specified session does not exist, or the setup of the specified session has not been modified after the last time it was saved.

Example

The following example shows how to use this function:

```
uid = verifOpenCellView("test" "new" "verifier")  
=> 0
```

```
verifIsSessionModified(uid)  
=> nil
```

Adds a new requirement

```
verifAddReq(uid "MyReq")  
=> "3300663f-d165-4796-8518-b103cb7490d2"
```

```
verifIsSessionModified(uid)  
=> t
```

verifIsValidSession

```
verifIsValidSession(  
    g_sessionId  
)  
=> t / nil
```

Description

Confirms if the given session is a valid ADE Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	The specified session is a valid ADE Verifier session.
nil	The specified session is not a valid ADE Verifier session.

Example

The following example shows how to use this function:

```
uid = verifOpenCellView("test" "new" "verifier")  
=> 0  
  
verifIsValidSession(uid)  
=> t  
  
verifIsValidSession("0")  
=> t  
  
verifIsValidSession(hiGetCurrentWindow())  
=> t  
  
verifIsValidSession(1)  
=> nil
```

verifGetWindow

```
verifGetWindow(  
    g_sessionId  
)  
=> w_windowId / nil
```

Description

Returns the window ID related to the specified session ID.

Argument

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>w_windowId</i>	Window ID of the specified session.
nil	Session does not exist or there is no corresponding window.

Example

The following example retrieves the window ID of a newly created Verifier cellview:

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
windowID = verifGetWindow(sessionId)  
window:2
```

verifGetSessionCellView

```
verifGetSessionCellView(  
    g_sessionId  
)  
=> l_libCellView / nil
```

Description

Returns the cellview name for the specified session.

Argument

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>l_libCellView</i>	A list containing the library, cell and view for the specified session.
<i>nil</i>	The session does not exist.

Example

The following example retrieves the library, cell, and view of a newly opened Verifier session:

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
verifGetSessionCellView(sessionId)  
("test" "setup" "verifier")
```

verifRegisterCallback

```
verifRegisterCallback(  
    g_callback  
)  
=> g_callback
```

Description

Registers a callback with Verifier which is executed at various times, such as, when a session is opened, saved, closed, simulations are started, finished, and when percentage changes.

Callback registration can be done at any time, and even before any Verifier sessions are opened.

The callback function takes three arguments, `x_session`, `s_signal`, and `l_args`. The `x_session` is the session identifier that calls the callback. `x_signal` is a symbol containing the type of callback, and `args` is a list containing data specific to the signal. For example, for the 'sessionSaved signal the `args` is a list

'("opamp090.full_diff_opamp_AC:verifier"), whereas for 'simulationPercentage the list is '("opamp090/full_diff_opamp_AC/maestro/Active" 49 98 200), and for 'requirementStatusChange it is '("ID6" "Passed").

The following table describes the supported signals:

Signal	args	When the signal is sent
'sessionCreated	session_num	When a session is first created, either when it is opened from a cellview, or created from scratch. Example: '(0)
'sessionDeleted	session_num	When the session has finally been deleted, which is typically done by closing the window. Example: '(0)
'sessionSaved	libName cellName viewName	When the session is saved to disk. Example: '("myLib" "myCell" "verifier")
'sessionLoaded		

Virtuoso ADE Verifier SKILL Reference

Verifier Session and Setup Functions

Signal	args	When the signal is sent
	libName cellName viewName	When the session has been loaded from disk. Example: ' ("myLib" "myCell" "verifier")
'simulationStatus	imp_name sim_status	When the simulation status of the given implementation changes. The <code>sim_status</code> is a string such as "Running", "Finished". Example: ' ("myLib/myCell/maestro/Active" "Finished")
'simulationPercentage	imp_name int_percentageComplete int_numCompleted int_numSubmitted	When the implementation simulation percentage changes. The three numbers are the calculated percentage complete, the number of points that have completed, and the total number of points. Example: ' ("myLib/myCell/maestro/Active" 75 3 4).
'simulationsDone		When all of the simulations for the implementations have completed. Example: <code>nil</code> .
'simulationsAdded	Int_number_added	The number of implementations that have been added to the simulation queue. Typically this will happen when the user clicks Run. Example: ' (3).
'requirementStatusChange	req_id result_status	The simulation results status of the requirement has changed. Typically sent out when the simulation finishes, but is also sent out when opening a Verifier session with results available. The <code>result_status</code> is a string such as "No Results", "Not Run", "Pass", "Fail". Example: ' ("req_1" "Pass").

Virtuoso ADE Verifier SKILL Reference

Verifier Session and Setup Functions

Arguments

g_callback A symbol or function object, where the symbol is the name of a function that takes three arguments, and the function object is a callable function that also takes three arguments.

Value Returned

g_callback The callback is registered.

Example

The following example shows how to register a callback with Verifier:

```
defun(myCallback (sess sig args)
info("Callback: %L %L %L\n" sess sig args)
)
myCallback

verifRegisterCallback('myCallback)
=> myCallback

verifRegisterCallback(lambda((sess sig args) info("Lambda CB: %L %L %L\n" sess sig
args)))
funobj@0x254db9a8
```


verifRemoveCallback

```
verifRemoveCallback(  
    g_callback  
)  
=> t / nil
```

Description

Remove a callback that was previously registered using `verifRegisterCallback(cb)`.

Argument

<i>g_callback</i>	A symbol or function object.
-------------------	------------------------------

Value Returned

<i>t</i>	The callback is removed.
<i>nil</i>	The callback is not registered and therefore is not removed.

Example

The following example shows how to remove a registered callback from Verifier:

```
defun(myCallback (sess sig args)  
  
    info("Callback: %L %L %L\n" sess sig args)  
    )  
myCallback  
  
verifRegisterCallback('myCallback)  
myCallback  
  
fn = verificRegisterCallback(lambda((sess sig args) info("Lambda CB: %L %L %L\n" sess  
sig args)))  
funobj@0x254db9a8  
  
verifRemoveCallback('myCallback)  
=> t  
  
verifRemoveCallback('myCallback)  
=> nil  
verifRemoveCallback(fn)  
=> t
```

verifGetCallbacks

```
verifGetCallbacks(  
    )  
=> l_callbacks
```

Description

Return the list of callbacks that have been registered by `verifRegisterCallback(sess cb)`.

Value Returned

l_callbacks Successfully exported the mapping.

Example

The following example starts a Verifier cellview and returns the registered callbacks:

```
defun(myCallback (sess sig args)  
info("Callback: %L %L %L\n" sess sig args)  
)  
myCallback  
  
verifRegisterCallback('myCallback)  
myCallback  
  
verifRegisterCallback(lambda((sess sig args) info("Lambda CB: %L %L %L\n" sess sig  
args)))  
funobj@0x254db9a8  
  
verifGetCallbacks()  
(myCallback funobj@0x254db9a8)
```

Preferences Functions

Use the following functions to set your Verifier preferences:

- verifGetOptions
- verifGetOptionVal
- verifSetOptionVal

verifGetOptions

```
verifGetOptions(  
    g_sessionId  
    [g_getValues]  
)  
=> l_optionNameList | nil
```

Description

Returns the list of Verifier preference options. These options are also available in the Virtuoso ADE Verifier Preferences form.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_getValue</i>	When non-nil the return value will be a list of pairs, where each pair will be the name of the option and its current value.

Value Returned

<i>l_optionNameList</i>	The list of all preference options of Verifier.
<i>nil</i>	The command is not successful.

Example

The following example opens a Verifier cellview and obtains the list of preference options:

```
uid=verifOpenCell("myLibrary" "myCell" "verifier")  
0  
verifGetOptions(uid)  
("appendimpinfo" "autocreaterreports" "batchhtmlreport" "displaycpk"  
 "displaymax" "displaymean" "displaymin" "displaysigmatotarget"  
 "displaystandarddeviation"  
 "displayyield" "enableCoverage" "enablecrossselection" "impjobpolicy"  
 "impresepperiod"  
 "impuimode" "linkdatasheets" "mappingonetoone" "openhtmldialog"  
 "overwritereports"  
 "palsdir" "palswhere" "reqspecs" "reqspecscheckunits" "simparallelcount"  
 "userreportidenticalhistory" "weightedAverageForCoverage"  
)  
verifGetOptions(uid t)
```

Virtuoso ADE Verifier SKILL Reference

Verifier Session and Setup Functions

```
(("appendimpinfo" t)
 ("displaycpk" nil)
 ("displaymax" nil)
 ("displaymean" nil)
 ("displaymin" nil)
 ("displaysigmatotarget" nil)
 ("displaystandarddeviation" nil)
 ("displayyield" nil)
 ("enableCoverage" t)
 ("enablecrossselection" t)
 ("impjobpolicy" "")
 ("impresexppperiod" 0)
 ("impuimode" nil)
 ("linkdatasheets" nil)
 ("mappingonetoone" nil)
 ("openhtmldialog" nil)
 ("overwritereports" t)
 ("palsdir" "")
 ("palswhere" "currentcellview")
 ("reqspecs" "Use Requirement Spec and Check")
 ("reqspecscheckunits" t)
 ("simparallelcount" 1)
 ("usereportidenticalhistory" nil)
 ("weightedAverageForCoverage" t)
)
```

verifGetOptionVal

```
verifGetOptionVal(  
    g_sessionId  
    t_optionName  
)  
=> t_optionValue / nil
```

Description

Returns the value of a preference option set in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_optionName</i>	The name of the preference option. The name must be one of the values returned by verifGetOptions .

Value Returned

<i>l_optionValue</i>	The value of the specified preference option.
nil	The specified option does not have a value, or the command is not successful.

Example

The following example opens a Verifier cellview, and retrieves the values of the specified options. For options that are switched on or off, Verifier returns YES or NO, respectively. For options with other values, Verifier returns the set values. The example also illustrates some warnings and errors that Verifier issues based on incorrect inputs.

```
uid=verifOpenCell("test" "sample" "verifier")  
0  
verifGetOptionVal(uid "enableCoverage")  
t  
  
verifGetOptionVal(uid "simparallelcount")  
2  
  
verifGetOptionVal(uid "reqspecs")  
"Use Requirement Spec and Check"
```

verifSetOptionVal

```
verifSetOptionVal(  
    g_sessionId  
    t_optionName  
    g_value  
)  
=> t/nil
```

Description

Sets the value of the specified preference option in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_optionName</i>	The name of the option.
<i>g_value</i>	The value to assign to the option.

Value Returned

<i>t</i>	Successfully sets the new value.
<i>nil</i>	The specified option does not exist or the value is invalid.

Example

The following example shows how to rename an implementation:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 0  
  
verifSetOptionVal(sess "simparallelcount" 4)  
=> t  
  
verifSetOptionVal(0 "displaymax" t)  
=> t  
verifSetOptionVal(sess "simparallelcount" "abc")  
  
*WARNING* (VERIFIER-1803): verifSetOption: invalid value '"abc"' for integer option  
- check the value and try again.  
  
=> nil  
  
verifSetOptionVal(sess "xxx" "1")  
  
*WARNING* (VERIFIER-1800): verifSetOption: unknown option: 'xxx'
```

Virtuoso ADE Verifier SKILL Reference

Verifier Session and Setup Functions

=> nil

Implementation Functions

This chapter describes the following SKILL functions:

- Implementation Management Functions
- Data Extraction Functions

For information on the functions for setting implementation sets, simulating implementations, and extracting results, see Chapter 5, “Simulation and Results Extraction Functions.”

Implementation Management Functions

Use the following functions to add, move, and delete implementations:

- verifAddImp
- verifMoveImp
- verifRemoveImp
- verifOverwriteSpec
- verifGetImpEstRunTime
- verifGetImpPriority
- verifSetImpEstRunTime
- verifSetImpPriority
- verifUpdateImpEstRunTime
- verifGetImpSetPreRunScript
- verifSetImpSetPreRunScript
- verifGetMappableType

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

verifAddImp

```
verifAddImp(  
    g_sessionId  
    t_lib  
    t_cell  
    t_view  
    t_history  
    [ ?runState g_runState ]  
    [ ?runPlanState g_runPlanState ]  
    [ ?createRequirements g_createRequirements ]  
)  
=> t / nil
```

Description

Adds a new implementation, that is a `maestro` cellview, to the specified Verifier session.

Arguments

<code>g_sessionId</code>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<code>t_lib</code>	The library name of the <code>maestro</code> cellview added to the specified Verifier session.
<code>t_cell</code>	The cell name of the <code>maestro</code> cellview added to the specified Verifier session.
<code>t_view</code>	The view name of the <code>maestro</code> cellview added to the specified Verifier session.
<code>t_history</code>	The history name of the <code>maestro</code> cellview added to the specified Verifier session. If you do not specify the history name, Verifier uses the <code>Active</code> history of the <code>maestro</code> cellview.
<code>?runState g_runState</code>	Sets the check state of the <i>Run</i> check box while adding the new implementation to the specified Verifier session. The default value is <code>t</code> .
<code>?runPlanState g_runPlanState</code>	

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

Sets the check state of the *Run Plan* check box while adding the new implementation to the specified Verifier session. The default value is `nil`.

`?createRequirements g_createRequirements`

If enabled, create the requirements for the implementation hierarchically. The default value is `nil`.

Value Returned

<code>t</code>	The <code>maestro</code> cellview is added to the specified Verifier session.
<code>nil</code>	The <code>maestro</code> cellview is not added to the specified Verifier session.

Examples

The following example opens a Verifier cellview and adds an implementation cellview.

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
```

Adds the given implementation to the session:

```
verifAddImp("0" "opamp090" "full_diff_opamp_AC" "maestro" "Active")
=> t
```

Adds a second implementation to the session, but does not enable the Run check box and create requirements from it:

```
verifAddImp(0 "opamp090" "full_diff_opamp_AC" "maestro" "Interactive.1" ?runState
nil ?createRequirements t)
=> t
```

To turn on the *Run* check box:

```
verifAddImp(uid "opamp090" "full_diff_opamp" "maestro" "Active" ?runState t)
Verifier: Adding implementation cellview opamp090/full_diff_opamp/maestro
=> t
```

To turn off the *Run* check box:

```
verifAddImp(uid "opamp090" "full_diff_opamp" "maestro" "Active" ?runState nil)
Verifier: Adding implementation cellview opamp090/full_diff_opamp/maestro
=> t
```

verifMoveImp

```
verifMoveImp(  
    g_sessionId  
    t_lib  
    t_cell  
    t_view  
    t_history  
    [ x_itemIndex ]  
)  
=> t / nil
```

Description

Moves an implementation to the specified location in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_lib</i>	The library name of the implementation.
<i>t_cell</i>	The cell name of the implementation.
<i>t_view</i>	The view name of the implementation.
<i>t_history</i>	The history name of the implementation.
<i>x_itemIndex</i>	The item index of the implementation in the implementations list. If you do not specify the index, its default value 0 is used, which moves the specified implementation as the last entry.

Value Returned

<i>t</i>	The implementation is moved to the specified location.
<i>nil</i>	The implementation is not moved to the specified location.

Examples

The following example opens a Verifier cellview and lists the implementation cellviews. It moves the last implementation cellview to the specified location.

```
uid = verifOpenCellView("test" "sample" "verifier")
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

=> 0

The implementation will be moved to be the last item:

```
verifMoveImp(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active")  
=> t
```

The implementation will be moved to be the first item:

```
verifMoveImp(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active" 1)
```

verifRemoveImp

```
verifRemoveImp(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
)  
=> t / nil
```

Description

Deletes an implementation from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impLib</i>	The library name of the implementation.
<i>t_impCell</i>	The cell name of the implementation.
<i>t_impView</i>	The view name of the implementation.
<i>t_impHistory</i>	The history name of the implementation.

Value Returned

<i>t</i>	The specified implementation is deleted from the Verifier session.
<i>nil</i>	The specified implementation does not exist in the Verifier session for deletion, or the command is not successful.

Examples

The following example opens a Verifier cellview that has an implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/Active. This implementation is then deleted from the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0
```

Get all implementations:

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

```
verifGetImps(uid)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "MonteCarlo.3.R0")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active")
("amsPLL" "CP_TB" "maestro" "Active")
)
```

Delete the specified implementation:

```
verifRemoveImp(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=> t
```


verifOverwriteSpec

```
verifOverwriteSpec(  
    g_sessionId  
    g_impLib  
    g_impCell  
    g_impView  
    g_impHistory  
    [ ?testName t_testName ]  
    [ ?outputName t_outputName ]  
    [ ?pushToImp g_pushToImp ]  
)  
=> t / nil
```

Description

Pull/push the specification and unit from or to the mapped implementation items hierarchically.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impLib</i>	Library name of the implementation.
<i>t_impCell</i>	Cell name of the implementation.
<i>t_impView</i>	View name of the implementation.
<i>t_impHistory</i>	History name of the implementation.
?testName <i>t_testName</i>	Name of the implementation test.
?outputName <i>t_outputName</i>	Name of the implementation output.
?pushToImp <i>g_pushToImp</i>	If enabled, pushes the requirement's specification and unit to mapped implementation items. This is equivalent to 'Overwrite Implementation Specification'. Default is <i>t</i> .

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

Value Returned

<code>t</code>	Successfully overwrite the requirement / implementation's specification and unit.
<code>nil</code>	Failed to overwrite the requirement / implementation's specification and unit.

Examples

The following example opens a Verifier cellview that has an implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/Active'.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Overwrite the specification and unit for outputs belongs to the implementation:

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
=> t
```

Overwrite the specification and unit for outputs belongs to the test:

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
?testName "AC")
=> t
```

Overwrite the specification and unit for output 'Op_Region' belongs to the test 'AC':

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
?testName "AC" ?outputName "Op_Region")
=> t
```

Overwrite the specification and unit for requirements that are mapped to outputs belongs to the implementation:

```
verifOverwriteSpec(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"
?pushToImp nil)
=> t
```

verifGetImpEstRunTime

```
verifGetImpEstRunTime(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
)  
=> x_estRunTime / nil
```

Description

Retrieves the `EstRunTime` of specified implementation in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impLib</i>	Library name of the implementation.
<i>t_impCell</i>	Cell name of the implementation.
<i>t_impView</i>	View name of the implementation.
<i>t_impHistory</i>	History name of the implementation.

Value Returned

<i>x_estRunTime</i>	Integer number of seconds for estimated run time.
<i>nil</i>	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and retrieves the `EstRunTime` of specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=>0  
verifGetImpEstRunTime(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")  
=>55
```

verifSetImpEstRunTime

```
verifSetImpEstRunTime(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
    x_estRunTime  
)  
=> t / nil
```

Description

Sets the `EstRunTime` of specified implementation in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impLib</i>	Library name of the implementation.
<i>t_impCell</i>	Cell name of the implementation.
<i>t_impView</i>	View name of the implementation.
<i>t_impHistory</i>	History name of the implementation.
<i>x_estRunTime</i>	Integer number of seconds for estimated run time.

Value Returned

<i>t</i>	<code>EstRunTime</code> of specified implementation is set to <code>x_estRunTime</code> successfully.
<code>nil</code>	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the `EstRunTime` of the specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=>0
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

```
verifSetImpEstRunTime(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active" 70)  
=>t
```

verifGetImpPriority

```
verifGetImpPriority(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
)  
=> x_priority / nil
```

Description

Get the 'Priority' of specified implementation in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impLib</i>	Library name of the implementation.
<i>t_impCell</i>	Cell name of the implementation.
<i>t_impView</i>	View name of the implementation.
<i>t_impHistory</i>	History name of the implementation.

Value Returned

<i>x_priority</i>	Integer number for priority.
<i>nil</i>	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and retrieves the Priority of specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=>0  
verifGetImpPriority(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")  
=>70
```

verifSetImpPriority

```
verifSetImpPriority(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
    x_priority  
)  
=> t / nil
```

Description

Sets the `Priority` of specified implementation in a Verifier session.

Arguments

<code>g_sessionId</code>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<code>t_impLib</code>	Library name of the implementation.
<code>t_impCell</code>	Cell name of the implementation.
<code>t_impView</code>	View name of the implementation.
<code>t_impHistory</code>	History name of the implementation.
<code>x_priority</code>	Integer number for priority.

Value Returned

<code>t</code>	<code>Priority</code> of specified implementation is set to <code>x_priority</code> successfully.
<code>nil</code>	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the `Priority` of the specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=>0
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

```
verifSetImpPriority(1 "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active" 70)  
=>t
```


verifUpdateImpEstRunTime

```
verifUpdateImpEstRunTime(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
)  
=> t / nil
```

Description

Updates the `EstRunTime` to be the actual recorded run time from run summary data.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impLib</i>	Library name of the implementation.
<i>t_impCell</i>	Cell name of the implementation.
<i>t_impView</i>	View name of the implementation.
<i>t_impHistory</i>	History name of the implementation.
<i>x_priority</i>	Integer number for priority.

Value Returned

<i>t</i>	<code>EstRunTime</code> of specified implementation is updated successfully.
<i>nil</i>	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the `Priority` of the specified implementation.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=>0
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

```
verifUpdateImpEstRunTime(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")  
=>t
```

verifGetImpSetPreRunScript

```
verifGetImpSetPreRunScript(  
    g_sessionId  
    t_impSetName  
)  
=> t_preRunScriptFileName
```

Description

Retrieves the pre-run script file name specified for an implementation set.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impSetName</i>	Name of the implementation set.

Value Returned

<i>t_preRunScriptFileName</i>	File name for pre-run script attached to the given implementation set.
nil	Implementation does not exist in the Verifier session, or no script is set for the given implementation set.

Examples

The following example starts a Verifier session with a cellview and retrieves the pre-run script file name specified for an implementation set.

```
uid = verifOpenCellView("test" "pre-run" "verifier")  
=>0  
verifGetImpSetPreRunScript(uid "Weekly")  
=>"prerun.il"
```

verifSetImpSetPreRunScript

```
verifSetImpSetPreRunScript(  
    g_sessionId  
    t_impSetName  
    t_preRunScriptFileName  
)  
=> t / nil
```

Description

Sets the specified pre-run script file name for an implementation set.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impSetName</i>	Name of the implementation set.
<i>t_preRunScriptFileName</i>	File name of the pre-run script attached to an implementation set.

Value Returned

<i>t</i>	The pre-run script for the given implementation set is set successfully.
nil	Implementation does not exist in the Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview and sets the pre-run script for an implementation set.

```
uid = verifOpenCellView("test" "pre-run" "verifier")  
=>0  
verifSetImpSetPreRunScript(uid "Weekly" "prerun.il")  
=>t
```

;; When the pre-run script file does not exist

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

```
verifSetImpSetPreRunScript(uid "Weekly" "abc.il")  
*WARNING* (VERIFIER-1368): The pre-run script file 'abc.il' for implementation set  
'Weekly' does not exist.  
Either update the path to the script, or ensure that the script exists in the  
location before running this implementation set.  
=>nil
```

verifGetMappableType

```
verifGetMappableType(  
    g_sessionId  
    t_path  
)  
=> t_type / nil
```

Description

Returns the type of the mappable object.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_path</i>	Hierarchical path and name of the mappable object. Each level of the hierarchy is separated by a '/

Value Returned

<i>t_type</i>	The mappable object type. The type can be one of these: Implementation, Test, Output, Run, StatOutput, and StatTest.
<i>nil</i>	The operation was unsuccessful.

Examples

The following set of examples show how to retrieve the mappable object from a cellview in different scenarios:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 0  
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active")  
=> "Implementation"  
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active/  
NoSuchTest")  
=> nil  
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active/  
opamp090:full_diff_opamp_AC:1")  
=> "Test"  
verifGetMappableType(sess "opamp090/full_diff_opamp_AC/maestro/Active/  
opamp090:full_diff_opamp_AC:1/Current")  
=> "Output"
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

Data Extraction Functions

Use the following functions to extract data from implementations:

- verifGetImps
- verifGetImpData
- verifGetImpTestOutputs
- verifGetImpTests
- verifUpdate

verifGetImps

```
verifGetImps(  
    g_sessionId  
    [ ?library t_impLib ]  
    [ ?cell t_impCell ]  
    [ ?view t_impView ]  
    [ ?history t_impHistory ]  
    [ ?runState g_runState ]  
    [ ?runPlanState g_runPlanState ]  
    [ ?extRef g_extRef ]  
    [ ?priority x_priority ]  
    [ ?estRunTime x_estRunTime ]  
)  
=> limps / nil
```

Description

Retrieves the list of the implementations from a Verifier session that have the same library, cell, view or history name as specified, the same check state for the 'Run' or 'RunPlan' check box.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>?library t_impLib</i>	The library name of the implementation.
<i>?cell t_impCell</i>	The cell name of the implementation.
<i>?view t_impView</i>	The view name of the implementation.
<i>?history t_impHistory</i>	The history name of the implementation.
<i>?runState g_runState</i>	The check state of the 'Run' check box.
<i>?runPlanState g_runPlanState</i>	The check state of the 'RunPlan' check box.
<i>?extRef g_extRef</i>	The referenced state of the implementation.
<i>?priority x_priority</i>	The integer number of implementation priority.

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

`?estRunTime x_estRunTime`

The integer number of implementation estimated run time.

Value Returned

`limps`

The list of implementations that match the conditions;

`nil`

There is no matched implementations or the command is not successful.

Examples

The following example opens a Verifier cellview and retrieves the implementations:

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Gets all implementations:

```
verifGetImps(uid)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active")
 ("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
 ("Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active")
 ("amsPLL" "CP_TB" "maestro" "Active"))
```

Gets the implementations belonging to library 'amsPLL':

```
verifGetImps(uid ?library "amsPLL")
(("amsPLL" "CP_TB" "maestro" "Active"))
```

Gets the implementation with enabled 'Run' check box:

```
verifGetImps(uid ?runState t)
(("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
 ("amsPLL" "CP_TB" "maestro" "Active"))
```

Gets the implementations that belong to cell 'OpAmp' and have the 'RunPlan' check box disabled:

```
verifGetImps(uid ?cell "OpAmp" ?runPlanState nil)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active")
 ("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"))
```

;; Gets the referenced implementations:

```
verifGetImps(uid ?extRef t)
=> nil
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

:: Gets the implementations that have priority equal to '50':

```
verifGetImps(uid ?priority 50)
(("Two_Stage_Opamp" "OpAmp" "maestro_sweeps-corners" "Active"))
```

:: Get the implementations that have priority greater than '40':

```
verifGetImps(uid ?priority lambda((x) x>40))
(("Two_Stage_Opamp" "OpAmp" "maestro_sweeps-corners" "Active")
("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
)
```

verifGetImpData

```
verifGetImpData(  
    g_sessionId  
    g_impLib  
    g_impCell  
    g_impView  
    g_impHistory  
    [ ?runName t_runName ]  
    [ ?testName t_testName ]  
    [ ?outputName t_outputName ]  
    [ ?statName t_statName ]  
    [ ?dataName t_dataName ]  
)  
=> o_dataTable/ t_dataValue/ nil
```

Description

Get the data information for implementation. The basic data is 'type', 'name' and 'map'. For implementation, there are extra data 'runState', 'runPlanState', and 'runMode'; for run, there are extra data 'runMode'; for output, there are extra data 'spec' and 'unit'; for statistical output, there are extra data 'statType'.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impLib</i>	The library name of the implementation.
<i>t_impCell</i>	The cell name of the implementation.
<i>t_impView</i>	The view name of the implementation.
<i>t_impHistory</i>	The history name of the implementation.
<i>?runName</i> <i>t_runName</i>	The name of the implementation run plan run.
<i>?testName</i> <i>t_testName</i>	The name of the implementation test.
<i>?outputName</i> <i>t_outputName</i>	The name of the implementation output.
<i>?statName</i> <i>t_statName</i>	The name of the implementation statistical output.

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

`?dataName` The data name is the key in each table element.
`t_dataName`

Value Returned

`o_dataTable` Table of implementation object data.
`t_dataValue` String value for the specified data name.
`nil` The implementation does not exist or the operation does not successful.

Examples

The following example opens a Verifier cellview and retrieves the implementation object data.

```
uid = verifOpenCellView("test" "results" "verifier")
=> 0
```

Get data for implementation:

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"))
(("runMode" "Monte Carlo Sampling")
 ("name" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"))
 ("runState" "false")
 ("type" "Implementation")
 ("map" "8509cd48-5cb7-4f65-a975-048afb5a32ef")
)
```

Get data for implementation test:

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"
?testName "AC"))
(("name" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" AC))
 ("type" "Test")
 ("map" "863951f4-3424-4315-b6fc-edeadbed2b1f")
)
```

Get data for implementation output:

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"
?testName "AC" ?outputName "Current"))
(("unit" "A")
 ("name" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" "AC" "Current"))
 ("spec" "< 1.5m")
 ("type" "Output")
 ("map" "dac5ef00-cbfa-4e7a-b19d-6a089e776efc")
)
```

Get data for implementation statistical output:

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"
?testName "AC" ?outputName "Current"?statName "Current::Mean"))
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

```
(("name" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" "AC" "Current"
"Current::Mean"))
("statType" "Mean")
("type" "StatOutput")
("map" "87d04770-c703-456c-be78-99afd4dbb447")
)
```

Get data for implementation run:

```
tableToList(verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_run_plan"
"Active" ?runName "Nominal"))
(("runMode" "Single Run, Sweeps and Corners")
("name" list("Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active" "Nominal"))
("type" "Run")
)
```

Get data value by data name:

```
verifGetImpData(uid "Two_Stage_Opamp" "OpAmp" "maestro_run_plan" "Active"
?dataName "runMode")
"Run Plan"
```

verifGetImpTestOutputs

```
verifGetImpTestOutputs(  
    g_sessionId  
    t_lib  
    t_cell  
    t_view  
    t_history  
    t_test  
    [ t_runName ]  
)  
=> l_impTestOutputs / nil
```

Description

Returns the list of the outputs for an implementation test in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_lib</i>	Library name of the implementation.
<i>t_cell</i>	Cell name of the implementation.
<i>t_view</i>	View name of the implementation.
<i>t_history</i>	History name of the implementation.
<i>t_test</i>	Name of the test in the specified implementation.
<i>t_runName</i>	Run name for the implementation is in 'Run Plan' mode. This argument is optional.

Value Returned

<i>l_impTestOutputs</i>	The list of the outputs in the specified test.
<i>nil</i>	The specified test does not exist or there is no outputs in the specified test.

Example

Note: For an example of how `verifGetImpTestOutputs` can be used in a custom function, see [“Custom Function to Copy Implementation Units to Requirements”](#).

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

The following example opens a Verifier cellview and retrieves all the outputs of a test of the specified implementation:

```
uid = verifOpenCell("test" "sample" "verifier")
=>t
tests=verifGetImpTests(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active")
("opamp090:full_diff_opamp_AC:1" "opamp090:full_diff_opamp_TRAN:1")
outputs=verifGetImpTestOutputs(uid "opamp090" "full_diff_opamp_AC" "maestro"
"Active" "opamp090:full_diff_opamp_AC:1")
("Bandwidth" "Current" "DCGain" "InputRandomOffset" "MAC_DCGain")
```


verifGetImpTests

```
verifGetImpTests(  
    g_sessionId  
    t_lib  
    t_cell  
    t_view  
    t_history  
    [ t_runName ]  
)  
=> l_impTests / nil
```

Description

Returns the list of all the tests for an implementation in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_lib</i>	The library name of the implementation.
<i>t_cell</i>	The cell name of the implementation.
<i>t_view</i>	The view name of the implementation.
<i>t_history</i>	The history name of the implementation.
<i>t_runName</i>	The run name when used with a run plan of the implementation. This is optional.

Value Returned

<i>l_impTests</i>	The list of all the tests of the specified cellview history.
<i>nil</i>	The specified implementation or run does not exist or there is no test in the specified implementation or run.

Example

Note: For an example of how `verifGetImpTests` can be used in a custom function, see [“Custom Function to Copy Implementation Units to Requirements”](#).

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

The following example opens a Verifier cellview and retrieves all the tests belonging to the specified implementation:

```
uid=verifOpenCellView("test" "sample" "verifier")
0
tests=verifGetImpTests(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active")
("opamp090:full_diff_opamp_AC:1" "opamp090:full_diff_opamp_TRAN:1")
```

verifUpdate

```
verifUpdate(  
    g_sessionId  
)  
=> t / nil
```

Description

Updates the Verifier setup from local implementations, and referenced Verifier cellviews.

Arguments

<i>t_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	Verifier setup was updated successfully.
nil	Verifier setup failed to update successfully.

Example

The following example opens a Verifier cellview, updates the Verifier setup from local implementations, and referenced Verifier cellviews:

```
sessionId = verifOpenCellView("test" "sample" "verifier")  
=> 0  
verifUpdate(sessionId)  
=> t
```

Virtuoso ADE Verifier SKILL Reference

Implementation Functions

Requirement-to-Implementation Mapping Functions

This chapter describes the following SKILL functions:

- [verifMapping](#)
- [verifGetImpMapping](#)
- [verifGetReqMapping](#)
- [verifExportMapping](#)
- [verifImportMapping](#)

verifMapping

```
verifMapping(  
    g_sessionId  
    g_reqsList  
    gimpsList  
    [ g_operation ]  
)  
=> t / nil
```

Description

Maps one or more requirements to one or more implementations in a Verifier session. Each of the requirement map objects can include the cellview, run, test, and output information. This function is also used to modify or delete the existing requirement-implementation mappings. For example, if the requirements list is empty, all the mapping information for the specified implementations is deleted.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_req</i>	Single requirement ID or list of requirement IDs to be mapped. The requirements can be nil, a string, or a list of strings.
<i>l_impsList</i>	The implementations to be mapped to the specified requirements. The implementations can be nil, or a string, a list of strings, or a list of lists of strings.

Virtuoso ADE Verifier SKILL Reference

Requirement-to-Implementation Mapping Functions

g_operation

Specifies whether to merge the specified mapping with any existing mapping, or to overwrite any existing mapping.

The following string values are used to specify the task to be performed for outdated mapping information:

- `merge`: Merges the old mapping information. Similar to the behavior of *Include Old Mappings* on the *Requirements or Implementations Mapped but Not Selected* dialog box.
- `overwrite` (default value): Removes the old mapping information and maps only to the new implementations; Similar to the behavior of *Continue* on the *Requirements or Implementations Mapped but Not Selected* dialog box.

An implementation name is specified as a list of strings or as string with each item separated by "/" from the following components:

```
<lib> <cell> <view> <history> [<run>] [<test> [<statisticalTest>] ||  
[<output> [<statisticalOutput>]]]
```

e.g.

```
"lib/cell/view/history"  
list("lib" "cell" "view" "history")  
"lib/cell/view/history/run"  
"lib/cell/view/history/run/test"  
"lib/cell/view/history/test/statTest"  
"lib/cell/view/history/run/test/statTest"  
"lib/cell/view/history/test/output"  
"lib/cell/view/history/test/output/statOut"
```

Virtuoso ADE Verifier SKILL Reference

Requirement-to-Implementation Mapping Functions

Value Returned

<code>t</code>	The specified requirements have been mapped to the specified implementations.
<code>nil</code>	The specified requirements and implementations could not be mapped.

Examples

The following example opens a Verifier cellview and maps the specified requirements to the specified implementations. It also illustrates how to delete the mapping of requirements and implementations using `verifMapping`.

Map requirement with ID `"ID_LT8XzV"` to implementation `"l/c/v/h"`:

```
sess = verifOpenCellView("tests" "sample" "verifier")
verifMapping(sess "ID_LT8XzV" "l/c/v/h")
=> t
```

Remove the mapping of requirement ID `"ID_LT8XzV"`:

```
verifMapping(sess "ID_LT8XzV" nil)
=> t
```

Remove the mapping of implementation `"l/c/v/h"`:

```
verifMapping(sess nil list("l" "c" "v" "h"))
=> t
```

Map both requirement IDs `"ID_LT8XzV"` and `"ID_LT6YzV"` to the same output:

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list("l" "c" "v" "h" "test"
"output1"))
=> t
```

Map both requirement IDs `"ID_LT8XzV"` and `"ID_LT6YzV"` to the same two implementations:

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")
list("l" "c" "v" "h2"))
=> t
```

Add the mapping of both requirement IDs `"ID_LT8XzV"` and `"ID_LT6YzV"` to the same two implementations to any existing mapping:

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")
list("l" "c" "v" "h2"))) "merge")
=> t
```

Specify the implementation using the list notation:

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")))
=> t
```


Virtuoso ADE Verifier SKILL Reference

Requirement-to-Implementation Mapping Functions

Specify two implementations using the list notation:

```
verifMapping(sess list("ID_LT8XzV" "ID_LT6YzV") list(list("l" "c" "v" "h")
list("l" "c" "v" "h2"))
=> t
```

verifGetImpMapping

```
verifGetImpMapping(  
    g_sessionId  
    t_libName  
    t_cellName  
    t_viewName  
    t_historyName  
    [ ?runName t_runName ]  
    [ ?testName t_testName ]  
    [ ?outputName t_outputName ]  
    [ ?statName t_statName ]  
)  
=> l_reqIds / nil
```

Description

Returns the list of requirements that are mapped to the specified mappable object. The mappable object could be an implementation cellview, run, test or output.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_libName</i>	Name of the library that contains the implementation.
<i>t_cellName</i>	Name of the cell that contains the implementation.
<i>t_viewName</i>	Name of the view that contains the implementation.
<i>t_historyName</i>	Name of the history in the implementation.
<i>?runName</i> <i>t_runName</i>	Name of the implementation run.
<i>?testName</i> <i>t_testName</i>	Name of the test.
<i>?outputName</i> <i>t_outputName</i>	Name of the output.
<i>?statName</i> <i>t_statName</i>	Name of the statistical test or output.

Virtuoso ADE Verifier SKILL Reference

Requirement-to-Implementation Mapping Functions

Value Returned

<i>l_reqIds</i>	The list of requirements that are mapped to the specified mappable object.
<i>nil</i>	The specified mappable object does not exist or is not mapped.

Example

Open Verifier cellview and get a list of requirement IDs mapped to statistical test "AC::Overall_Yield":

```
sessionId = verifOpenCellView("test" "setup" "verifier")
=> 0
verifGetImpMapping(0 "test" "setup" "verifier" "Active" ?testName "AC" ?statName
"AC::Overall_Yield")
("ID_1" "ID_5.1")
```

verifGetReqMapping

```
verifGetReqMapping(  
    g_sessionId  
    t_reqId  
    [ ?types l_types ]  
    => l_implementations / nil
```

Description

Returns a list of mappable objects mapped to the specified requirement. The mappable object could be an implementation cellview, run, test or output.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	The ID of the requirement.
<i>?types l_types</i>	The mappable object types to be returned. Valid values are "implementation", "test", "output" and "run". If a value is not specified, all types are returned.

Value Returned

<i>l_implementations</i>	Successful operation.
<i>nil</i>	Unsuccessful operation.

Example

Open Verifier cellview and get a list of all implementations mapped to requirement "ID_1" then get a list of outputs mapped to the requirement:

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 0  
verifGetReqMapping(sessionId "ID_1")  
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" "TRAN"  
    "PhaseMargin"  
)  
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active" "TRAN")  
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active")  
)  
verifGetReqMapping(sessionId "ID_1" ?types '("output"))(("Two_Stage_Opamp" "OpAmp"  
"maestro_MC" "Active" "TRAN"  
    "PhaseMargin"  
)  
)
```

verifExportMapping

```
verifExportMapping(  
    g_sessionId  
    t_fileName  
    [ S_type ]  
    [ g_confirmOverwrite ]  
)  
=> t / nil
```

Description

Export the mapping to either a CSV or Excel file.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	The name of the file to export the mapping to.
<i>S_type</i>	Symbol or string to identify the export type. The supported types are "Excel" or "CSV". The default value is "CSV". The string name is case insensitive.
<i>g_confirmOverwrite</i>	If <i>t</i> and the file already exists, will ask for confirmation to overwrite it, otherwise will just go ahead and overwrite it.

Value Returned

<i>t</i>	Successfully exported the mapping.
<i>nil</i>	Failed to export the mapping.

Example

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 1  
verifExportMapping(sess "mapping.csv")  
=> t  
verifExportMapping(sess "mapping.xlsx" "Excel" t)  
=> t
```

verifImportMapping

```
verifImportMapping(  
    g_sessionId  
    t_fileName  
    S_type  
)  
=> t / nil
```

Description

Import mapping from an Excel or CSV file. The first row of the file will be ignored as it should be a header row. The rest of the rows should have at least two columns: `RequirementId` and `MappingName`. Any other columns will be ignored. Each row is a single Requirement to Implementation mapping, so if a requirement is mapped to multiple implementations, then there will be multiple rows with the same requirement id in the first column. For example:

```
Requirement,Mapping  
ID6,opamp090/full_diff_opamp_AC/maestro/Active/opamp090:full_diff_opamp_TRAN:1/  
SlewRate  
ID6,opamp090/full_diff_opamp_AC bad/maestro/Active/  
opamp090:full_diff_opamp_TRAN:1/SlewRate  
ID5,opamp090/full_diff_opamp_AC/maestro/Active/opamp090:full_diff_opamp_TRAN:1/  
SettlingTime
```

Note: Shell-style wildcard of "*" can be used in the names, but it is recommended to avoid numerous matches when importing the file. For example:

```
ID*,op*/full_diff_opamp_AC/maestro*/Active
```

is valid, but would map every requirement beginning with ID to all implementations that match. Any existing mapping on the requirement is retained, so any new mapping is added on top of that.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	File to import the mapping from.
<i>S_type</i>	String or symbol to specify the type of file. It can be either "CSV" (the default) or "Excel".

Virtuoso ADE Verifier SKILL Reference

Requirement-to-Implementation Mapping Functions

Value Returned

t	The import was successful.
nil	The import was unsuccessful.

Example

```
sess = verifOpenCellView("test" "sample" "verifier")
=> 0
verifExportMapping(sess "map.csv")
=> t
sess2 = verifOpenCellView("test" "sample2" "verifier")
=> 1
verifImportMapping(sess2 "map.csv")
=> t
```

Virtuoso ADE Verifier SKILL Reference

Requirement-to-Implementation Mapping Functions

Simulation and Results Extraction Functions

This chapter describes the following SKILL functions:

- Implementation Set Functions
- Results Extraction Functions
- Simulation Functions

Implementation Set Functions

Use the following functions to retrieve results on implementation sets:

- verifAddImpSet
- verifAddImpToImpSet
- verifGetImpSets
- verifGetImpsInImpSet
- verifSetImpSetName
- verifRemoveImpSet
- verifRemoveImpFromImpSet

verifAddImpSet

```
verifAddImpSet(  
    g_sessionId  
    t_impSetName  
)  
=> t / nil
```

Description

Adds a new implementation set in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impSetName</i>	Name of implementation set.

Value Returned

<i>t</i>	The implementation set is added to the specified Verifier session.
<i>nil</i>	The implementation set is not added to the specified Verifier session because of errors.

Example

The following example opens a Verifier cellview and add a new implementation set.

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 1  
verifAddImpSet(sessionId "Weekly")  
=> t
```

verifAddImpToImpSet

```
verifAddImpToImpSet (  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
    t_impSetName  
)  
=> t / nil
```

Description

Adds an implementation to the specified implementation set in a Verifier session.

Note: You can add an implementation to multiple implementation sets.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impLib</i>	The library name of the implementation.
<i>t_impCell</i>	The cell name of the implementation.
<i>t_impView</i>	The view name of the implementation.
<i>t_impHistory</i>	The history name of the implementation.
<i>t_impSetName</i>	The name of the implementation set.

Value Returned

<i>t</i>	The implementation is added to the implementation set.
<i>nilf</i>	The implementation or implementation set does not exist, or the command is not successful.

Example

The following example opens a Verifier cellview that has an implementation 'opamp090/full_diff_opamp_AC/maestro/Active'. It adds an implementation set 'Weekly' and then add this implementation to that set.

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
uid=verifOpenCellView("test" "sample" "verifier")
=> "0"
verifAddImpSet(uid "Weekly")
=> t
verifAddImpToImpSet(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active"
"Weekly")
=> t
```

verifGetImpSets

```
verifGetImpSets(  
    g_sessionId  
    [ ?parentName t_parentName ]  
    [ ?runnable g_runnable ]  
    [ ?isGroup g_isGroup ]  
)  
=> l_impSets / nil
```

Description

Retrieves the list of all the implementation sets in a Verifier session.

Argument

g_sessionId Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).

?parentName *t_parentName*
Name of the parent implementation set.

?runnable *g_runnable*
A check that when enabled, retrieves the implementation set that can be run from Verifier. Only the top-level implementation set is runnable.

?isGroup *g_isGroup*
A check that when enabled, retrieves the implementation set which is a group of implementations.

Value Returned

l_impSets The list of implementation sets in the Verifier session.
nil There is no implementation set in the Verifier session, or the command is not successful.

Example

The following example opens a Verifier cellview and retrieves all the implementation sets.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
verifGetImpSets(uid)
=> ("Weekly" "Weekly->Two_Stage_Opamp/OpAmp/maestro_MC/Active" "Weekly->PlanB"
    "Daily")
```

;; Retrieves the list of dependent implementation sets from a specified implementation or implementation set.

```
verifGetImpSets(sess ?parentName "Weekly")
=> ("Weekly->Two_Stage_Opamp/OpAmp/maestro_MC/Active" "Weekly->PlanB")
```

;; Retrieves the list of top-level implementation sets. Only a top-level implementation set is runnable.

```
verifGetImpSets(sess ?runnable t) nil
=> ("Weekly" "Daily")
```

;; Retrieves the list of implementation set that include a group

```
verifGetImpSets(sess ?isGroup t)
=> ("Weekly" "Weekly->PlanB" "Daily")
```

;; Retrieves the list of implementation sets that include an implementation

```
verifGetImpSets(sess ?isGroup nil)
=> ("Weekly->Two_Stage_Opamp/OpAmp/maestro_MC/Active")
```

verifGetImpsInImpSet

```
verifGetImpsInImpSet(  
    g_sessionId  
    t_impSetName  
)  
=> l_impSets / nil
```

Description

Retrieves the list of implementations from the specified implementation set in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impSetName</i>	The name of the implementation set.

Value Returned

<i>l_impSets</i>	The list of implementations from the specified implementation set in a Verifier session.
<i>nil</i>	The implementation set does not exist or there is no implementation in the specified set.

Example

The following example opens a Verifier cellview and retrieves the list of implementations from the specified implementation set in a Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifGetImpSets(uid)  
("Weekly" "Daily")  
verifGetImpsInImpSet(uid "Weekly")  
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "MonteCarlo.3.R0")  
 ("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Interactive.0")  
)  
verifGetImpsInImpSet(uid "Daily")  
=> nil
```


verifSetImpSetName

```
verifSetImpSetName(  
    g_sessionId  
    t_impSetName  
    t_newImpSetName  
)  
=> t / nil
```

Description

Renames the specified implementation set.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>t_impSetName</i>	The name of implementation set.
<i>t_newImpSetName</i>	The new name of implementation set.

Value Returned

<i>t</i>	The specified implementation set is renamed successfully.
<i>nil</i>	The specified implementation set does not exist, or the command is not successful.

Example

The following example opens a Verifier cellview that has an implementation set 'Weekly'. It renames this implementation set in the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifSetImpSetName(uid "Weekly" "Daily")  
=> t
```

verifRemoveImpFromImpSet

```
verifRemoveImpFromImpSet(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
    t_impSetName  
)  
=> t / nil
```

Description

Deletes an implementation from the specified implementation set in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impLib</i>	The library name of the implementation.
<i>t_impCell</i>	The cell name of the implementation.
<i>t_impView</i>	The view name of the implementation.
<i>t_impHistory</i>	The history name of the implementation.
<i>t_impSetName</i>	The name of the implementation set.

Value Returned

<i>t</i>	The specified implementation is deleted from the implementation set.
<i>nil</i>	The specified implementation set does not exist, the specified implementation cellview is not available in the implementation set, or the command is not successful.

Example

The following example opens a Verifier cellview that has an implementation 'opamp090/full_diff_opamp_AC/maestro/Active' and an implementation set 'Weekly'. It deletes this implementation from that set.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
verifRemoveImpFromImpSet(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"  
"Weekly")  
=> t
```

verifRemoveImpSet

```
verifRemoveImpSet(  
    g_sessionId  
    t_impSetName  
)  
=> t/nil
```

Description

Deletes the implementation set from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impSetName</i>	The name of the implementation set.

Value Returned

<i>t</i>	The specified implementation set is deleted from the Verifier session.
<i>nil</i>	The specified implementation set does not exist, or the command is not successful.

Example

The following example opens a Verifier cellview that has an implementation set 'Weekly'. It deletes this implementation set from the Verifier session.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifRemoveImpSet(uid "Weekly")  
=> t
```

Results Extraction Functions

Use the following functions to retrieve simulation results:

- verifGetResultDataForImp
- verifGetResultDataForReq
- verifReloadAllRes

verifGetResultDataForImp

```
verifGetResultDataForImp(  
    g_sessionId  
    t_lib  
    t_cell  
    t_view  
    t_history  
    [ ?runName t_runName ]  
    [ ?testName t_testName ]  
    [ ?outputName t_outputName ]  
    [ ?statName t_statName ]  
)  
=> o_resultDataTable / nil
```

Description

Returns all the result data for the specified implementation in a Verifier session. For each sub tables, there is a table that contains the multiple child items result data.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_lib</i>	The library name of the implementation.
<i>t_cell</i>	The cell name of the implementation.
<i>t_view</i>	The view name of the implementation.
<i>t_history</i>	The history name of the implementation.
<i>?runName</i> <i>t_runName</i>	The name for the implementation in 'Run Plan' mode. This argument is optional.
<i>?testName</i> <i>t_testName</i>	The name for the implementation test. This argument is optional.
<i>?outputName</i> <i>t_outputName</i>	The name for the implementation output. This argument is optional.
<i>?statName</i> <i>t_statName</i>	The name of the statistical value for implementation test or output. This argument is optional.

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

Value Returned

o_resultDataTable The table containing the result data for all the child items of the specified implementation.

nil The specified implementation does not exist, the results data is not available, or the command is not successful.

Example

The following example opens a Verifier cellview, and retrieves the result data of the specified implementation.

```
uid = verifOpenCellView("test" "results" "verifier")
=>0
tableToList(verifGetResultDataForImp(uid "Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active"))
(("Gain" table:info)
 ("TRAN" table:info)
 ("SettlingTime" table:info)
 ("Current" table:info)
 ("Voffset" table:info)
 ("RelativeSwingPercent" table:info)
 ("PhaseMargin" table:info)
 ("UGF" table:info)
 ("Implementation" list("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active"))
 ("Swing" table:info)
 ("AC" table:info)
 ("Op_Region" table:info)
)
```

To use the result data sub table:

```
tableToList(verifGetResultDataForImp(0 "Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active") ["Current"])
(("points" "1")
 ("history" "verifier_run.102.R0")
 ("spec" "tol 7m 6%")
 ("disabledPoints" "0")
 ("min" "7.13069521418212e-3")
 ("max" "7.13069521418212e-3")
 ("canceledPoints" "0")
 ("Implementation" list("opamp090" "full_diff_opamp_AC" "maestro" "Active"))
 ("failedPoints" "0")
 ("minParams" "Model=(gpd090.scs,MC_models),corner=C1")
 ("Test" "opamp090:full_diff_opamp_AC:1")
 ("Output" "Current")
 ("status" "Pass")
 ("maxParams" "Model=(gpd090.scs,MC_models),corner=C1")
 ("passedPoints" "1"))
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

)

verifGetResultDataForReq

```
verifGetResultDataForReq(  
    g_sessionId  
    t_reqId  
)  
=> o_resultDataTable / nil
```

Description

Return the simulation result data for the specified requirement in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_reqId</i>	The ID of the requirement.

Value Returned

<i>o_resultDataTable</i>	The table containing the result data.
<i>nil</i>	The specified requirement does not exist, the result data is not available, or the command is not successful.

Example

The following example opens a Verifier cellview and retrieves the result data of the implementation mapped to the requirement in that cellview.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0
```

Requirement 'ID1' of 'Note' type

```
tableToList(verifGetResultDataForReq(uid "ID1"))  
(("reqStatus" "0%")  
  ("passed" "0")  
  ("reqTitle" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active (Note)"))  
  ("reqType" "Note")  
  ("noResults" "77")  
  ("reqId" "ID1")  
  ("reqHier" "1"))
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
    ("failed" "0")
    ("unmapped" "0")
)
```

Mapped requirement 'ID1.1'

```
tableToList(verifGetResultDataForReq(uid "ID1.1"))
(("reqStatus" "Not Run")
  ("passed" "0")
  ("reqTitle" list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active"))
  ("reqType" "Ran Ok")
  ("noResults" "1")
  ("reqId" "ID1.1")
  ("reqHier" "1->1.1")
  ("failed" "0")
  (list("Two_Stage_Opamp" "OpAmp" "maestro_MC" "Active") table:info)
)
```

verifReloadAllRes

```
verifReloadAllRes(  
    g_sessionId  
)  
=> t / nil
```

Description

Reloads the simulation results for all implementations that have the Run check box unchecked. Equivalent to clicking the 'Reload Simulation Results' button on the toolbar of UI.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	Successful reloading all the implementation simulation results.
nil	Failed to reload all the implementation simulation results.

Example

This example shows how to reload the simulation results for all implementations within the Verifier session.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=> 0  
verifReloadAllRes(uid)  
=> t
```

verifEvaluateResults

```
verifEvaluateResults(  
    g_sessionId  
)  
=> t / nil
```

Description

Evaluates all requirements and does a force update of the results in special cases.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	Re-evaluates the results for all requirements.
nil	The command is not successful.

Example

The following example shows how to use this function:

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifEvaluateResults(uid)  
=> t
```

verifCopyAndUpdateResultsFromUserDefinedDirectory

```
verifCopyAndUpdateResultsFromUserDefinedDirectory(  
    g_sessionId  
)  
=> t / nil
```

Description

Copies the run summary data files from the user-defined directory to the `results` directory of the current cellview. Additionally, updates the current session setup to restore the run summary data file into the `results` directory of the current cellview.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
--------------------	--

Value Returned

<i>t</i>	Copies the run summary data files from the user-defined directory to the <code>results</code> directory of the current cellview and updates the current setup successfully.
<i>nil</i>	The command is not successful.

Example

The following example shows how to use this function:

```
uid = verifOpenCellView("test" "results" "verifier" ?openWindow nil)
```

```
INFO (VERIFIER-6603): You have opened a cellview that uses the user-defined  
directory './Verifier Results', earlier available from Edit -> Preference-> Run -  
> Location of the Implementation Run Summary Data.  
ADE Verifier no longer supports this feature and will automatically modify this  
setup to store the run summary data in the 'results' directory of the current  
cellview.
```

```
To create the run summary data, run simulations or load the implementation history  
results.
```

```
To copy the previous results from './Verifier Results' and update the current  
setup, make the session editable and use the SKILL function  
'verifCopyAndUpdateResultsFromUserDefinedDirectory(sessionId)'.  
To continue using the user-defined location for the run summary data, contact  
Cadence Customer Support for assistance.
```

```
==> 0
```

```
verifGetOptionVal(uid "palswhere")  
=>"userdir"
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
verifCopyAndUpdateResultsFromUserDefinedDirectory(uid)
```

```
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_basic_Active.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.Nominal.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.MC.xml.
INFO (VERIFIER-7109): Result data loaded: test/ro/verifier_user/results/
Two_Stage_Opamp_OpAmp_maestro_run_plan_Plan.0_RunPlan.Run.0.xml.
=> t
```

```
verifGetOptionVal(uid "palswhere")
=> "currentcellview"
```

```
verifCloseSession(uid ?saveIfModified t) INFO (VERIFIER-1507): Saved cellview
test.ro:verifier_user
=> t
```

Simulation Functions

Use the following functions to retrieve simulation results:

- verifImplsRun
- verifRun
- verifSetImpRun
- verifCheck
- verifCheckImp
- verifStop

verifImplsRun

```
verifImplsRun(  
    g_sessionId  
    t_libName  
    t_cellName  
    t_viewName  
    t_historyName  
)  
=>t / nil
```

Description

Checks if the specified implementation cellview can be run from a Verifier session.

Note: You can load the simulation results from the selected history of an implementation cellview that cannot be run from Verifier.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_libName</i>	The library name of the implementation.
<i>t_cellName</i>	The cell name of the implementation.
<i>t_viewName</i>	The view name of the implementation.
<i>t_historyName</i>	The history name of the implementation.

Value Returned

t	The implementation has the Run check box checked.
nil	The specified implementation does not exist or cannot be run from Verifier.

Example

The following example shows how to use this function:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 0  
verifImplsRun(sess "test" "sample" "maestro" "Active")  
=> t
```


verifRun

```
verifRun(  
    g_sessionId  
    [ ?implementation g_implementation ]  
    [ ?impSet g_impset ]  
    [ ?runPlanRun g_runPlanRun ]  
    [ ?mode g_mode ]  
    [ ?waitUntilDone g_waitUntilDone  
    ]  
)  
=> t/nil
```

Description

Runs or loads the result of the specified implementation(s) in a Verifier session.

Note: If the *Run* check box is enabled then it runs the implementation. If the check box is disabled, it loads any existing simulation results.

Arguments

g_sessionId Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).

?implementation g_implementation
The name of single implementation or list of implementation names where each implementation name can be either a string like "Lib/Cell/View/History" or a list ' ("Lib" "Cell" "View" "History") '. If no implementation is specified, all implementations in the session will be run.

?impSet g_impSet
The name of single implementation set or a list of implementation set names where each implementation set name must be a string. For example, "Weekly".

?runPlanRun g_runPlanRun
A string that represents the name of a single *RunPlan* run name or a list of *RunPlan* run names. For example, "Two_Stage_Opamp/OpAmp/maestro_run_plan/Run.0".

?mode g_mode
A string that represents the mode. The available values are: 'Local Run', 'Batch Run', 'Local with SPACE' or 'Batch with SPACE'. The default is 'Local Run'.

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

`?waitUntilDone` *g_waitUntilDone*

Waits for the simulations to complete the runs before executing any other command. If this is set to `nil`, the wait is disabled. The default is `nil`.

Value Returned

<code>t</code>	The simulations were successfully run or launched.
<code>nil</code>	Failed to run simulations.

Example

The following example shows how to use this function:

```
sess = verifOpenCellView("test" "sample" "verifier")
verifRun(sess)
```

Run all implementations

```
verifRun(sess ?mode "External Run" ?waitUntilDone t)
```

Run all implementations externally sessions and wait until they have finished

```
verifRun(sess ?implementations '("test" "sample" "maestro" "Active"))
```

Run a single implementation

```
verifRun(sess ?implementations '("test" "sample" "maestro" "Active") ("test"
"sample2" "maestro" "Interactive.0"))
```

Run two implementations

```
verifRun(sess ?implementations verifGetImps(sess ?isRun nil))
```

Reload the results of all implementations that do not have the Run check box enabled

```
verifRun(sessionId ?runPlanRun "Two_Stage_Opamp/OpAmp/maestro_run_plan/Active/
Nominal")
```

Runs the specified RunPlan run

```
verifRun(sessionId ?runPlanRun list("Two_Stage_Opamp/OpAmp/maestro_run_plan/
Active/Nominal", "Two_Stage_Opamp/OpAmp/maestro_run_plan/Active/MC"))
```

Runs the list of specified RunPlan runs

verifSetImpRun

```
verifSetImpRun(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
    g_runState  
)  
=> t / nil
```

Description

Sets the check state of the Run check box for the specified implementation in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impLib</i>	The library name of the implementation.
<i>t_impCell</i>	The cell name of the implementation.
<i>t_impView</i>	The view name of the implementation.
<i>t_impHistory</i>	The history name of the implementation.
<i>g_runState</i>	The run state of the implementation.

Value Returned

<i>t</i>	Sets the run state of implementation successfully.
<i>nil</i>	The specified implementation does not exist or the command is not successful.

Example

The following example opens a Verifier cellview that has an implementation 'opamp090/full_diff_opamp_AC/maestro/Active' and enables the run state for this cellview.

```
uid = verifOpenCellView("test" "results" "verifier")  
=> 0  
verifSetImpRun(uid "opamp090" "full_diff_opamp_AC" "maestro" "Active" t)  
=> t
```

verifCheck

```
verifCheck(  
    g_sessionId  
    [ ?implementation g_implementation ]  
    [ ?impSet g_impSet ]  
)  
=> t / nil
```

Description

Checks whether implementations changed since last run or reload by Verifier. In addition, it checks whether the lifetime of implementation simulation results is within the *Expiration Period of Implementation Results* since the last simulation of implementation cellview by Verifier or the last external creation of results. If no implementation or implementation set is specified, then all implementations in the session will be checked.

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

Arguments

<code>g_sessionId</code>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<code>?implementation g_implementation</code>	Name of single implementation or list of implementation names where each implementation name can be either a string like "Lib/Cell/View/History" or a list '("Lib" "Cell" "View" "History")'. If no implementation is specified, all implementations in the session will be run.
<code>?impSet g_impSet</code>	Name of a single implementation set or a list of implementation set names.

Value Returned

<code>t</code>	Check for changes in implementations and simulation results is done.
<code>nil</code>	Check for changes in implementation and simulation results is not done.

Example

The following example starts a Verifier session with a cellview and performs the check:

```
uid = verifOpenCellView("test" "results" "verifier")
=>0

;; Get all implementations
verifGetImps(uid)
(("Two_Stage_Opamp" "OpAmp" "maestro_MC" "MonteCarlo.3.RO")
 ("Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Interactive.0"))

;; Do the check
verifCheck(uid)
INFO (VERIFIER-1329): The implementation 'Two_Stage_Opamp/OpAmp/maestro_MC/
MonteCarlo.3.RO' has not yet been run by Verifier.
Therefore, no checks can be performed.

INFO (VERIFIER-1324): The implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Interactive.0' has changed as below:
some elements in the following test state have changed:
'AC: environmentOptions, faultFiles, faultRules, faults, outputs'
'TRAN: analyses, environmentOptions, faultFiles, faultRules, faults, outputs'
*WARNING* (VERIFIER-1326): Found outdated simulation results for implementation
'Two_Stage_Opamp/OpAmp/maestro_nominal/Interactive.0' because the implementation
cellview has changed since the last simulation run.
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

To ensure that Verifier has the latest simulation results, enable the preference option 'Report identical history before run' and rerun the implementation from Verifier.

=> t

verifCheckImp

```
verifCheckImp(  
    g_sessionId  
    t_impLib  
    t_impCell  
    t_impView  
    t_impHistory  
)  
=> t / nil
```

Description

Checks for changes of the specified implementation and loaded results.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_impLib</i>	Library name of the implementation.
<i>t_impCell</i>	Cell name of the implementation.
<i>t_impView</i>	View name of the implementation.
<i>t_impHistory</i>	History name of the implementation.

Value Returned

<i>t</i>	Check for changes is successful.
<i>nil</i>	Implementation does not exist, or the command is not successful.

Example

The following example starts a Verifier session with a cellview that has implementations 'opamp090/full_diff_opamp_AC/maestro/Active' and 'Two_Stage_Opamp/OpAmp/maestro_basic/Active'. It successfully checks for changes in implementations and loaded results.

```
uid = verifOpenCellView("test" "sample" "verifier")  
=>0  
  
verifCheckImp(uid "Two_Stage_Opamp" "OpAmp" "maestro_nominal" "Active")
```

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
INFO (VERIFIER-1329): The implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/Active' has not yet been run by Verifier.
Therefore, no checks can be performed.
```

```
=>t
```

```
verifCheckImp(uid "Two_Stage_Opamp" "OpAmp" "maestro_basic" "Active")
```

```
INFO (VERIFIER-1324): The implementation 'Two_Stage_Opamp/OpAmp/maestro_basic/Active' has changed as below:
following elements in the setup have changed:
specifications
```

```
*WARNING* (VERIFIER-1326): Found outdated simulation results for implementation
'Two_Stage_Opamp/OpAmp/maestro_basic/Active' because the implementation cellview
has changed since the last simulation run.
```

```
To ensure that Verifier has the latest simulation results, enable the preference
option 'Report identical history before run' and rerun the implementation from
Verifier.
```

```
=> t
```


verifStop

```
verifStop(  
    g_sessionId  
    [ ?implementation g_implementation ]  
    [ ?impSet g_impSet ]  
)  
=> t / nil
```

Description

Stops simulations for one or more implementations. If no implementation or implementation set is specified, then all running implementations in the session will be stopped.

Arguments

g_sessionId Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).

?implementation *g_implementation* Name of single implementation or list of implementation names where each implementation name can be either a string like "Lib/Cell/View/History" or a list ("Lib" "Cell" "View" "History"). If no implementation is specified, all implementations in the session will be run.

?impSet *g_impSet* Name of a single implementation set or a list of implementation set names.

Value Returned

t Successfully stops the simulations.

nil Fails to stop simulations.

Example

The following example starts a Verifier session with a cellview that has an implementation 'opamp090/full_diff_opamp_AC/maestro/Active' and an implementation set 'Weekly'.

Virtuoso ADE Verifier SKILL Reference

Simulation and Results Extraction Functions

```
uid = verifOpenCellView("test" "sample" "verifier")
=>0

verifRun(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp" "maestro_nominal"
"Active")))
Verifier: Started running implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Active' at 'Apr 15 10:00:50 2019'
verifStop(uid ?implementation list(list("Two_Stage_Opamp" "OpAmp"
"maestro_nominal" "Active")))
Verifier: Stopped running implementation 'Two_Stage_Opamp/OpAmp/maestro_nominal/
Active' at 'Apr 15 10:00:18 2019'
=>t

verifRun(uid ?impSet "Weekly")
=>t
verifStop(uid ?impSet "Weekly")
=>t
```

Verification Status and Reports Functions

This chapter describes the following SKILL function:

- verifPublishHTML

To know how this function can be used in custom callback functions, see Appendix A, “Additional Information.”

verifPublishHTML

```
verifPublishHTML(  
    g_sessionID  
    g_openBrowser  
)  
=> t / nil
```

Description

Generates and optionally displays an HTML report in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or <code>window(2)</code> .
<i>g_openBrowser</i>	An optional boolean argument. If this is not nil, the generated HTML report is displayed in the default web browser. Default value is <code>nil</code> .

Value Returned

<code>t</code>	The HTML file was generated.
<code>nil</code>	The HTML file was not generated.

Example

The following example opens a Verifier cellview and generates an HTML report.

```
sessionId = verifOpenCellView("test" "setup" "verifier")  
=> 0  
verifPublishHTML(sessionId t)  
=> t
```

Setup Library Assistant Functions

You can use several SKILL functions to work with the Setup Library Assistant (SLA) in the `nograph` or the non-GUI mode. These functions help you edit, delete, and read SLA components. The components can be corner setups, corner variables, corner model files, sweep setups, sweep variables, and verification spaces that contain sweep setups and corner setups. You can use these functions without opening a `maestro` or `verifier` view because the functions edit or read the `*sdb` files directly from the disk.

This chapter describes the following SKILL functions:

- Functions for opening and saving setup library views
 - ❑ [`slaOpenOrCreateView`](#)
 - ❑ [`slaSaveAndCloseView`](#)
- Functions for creating top-level components (sweep setups, corner setups, or spaces)
 - ❑ [`slaCreateCornerSetup`](#)
 - ❑ [`slaCreateSweepSetup`](#)
 - ❑ [`slaCreateVerificationSpace`](#)
- Functions for adding corners, variables, or model files in a corner setup or sweep setup
 - ❑ [`slaImportCorners`](#)
 - ❑ [`slaImportSweeps`](#)
 - ❑ [`slaAddSweepVariable`](#)
 - ❑ [`slaAddCornerVariable`](#)
 - ❑ [`slaAddCornerModelFile`](#)
 - ❑ [`slaAddDocument`](#)

Virtuoso ADE Verifier SKILL Reference

Setup Library Assistant Functions

■ Functions for reading setups from the setup library view

These functions allow reading the specified setups when you open the setup library view in either edit mode or read-only mode.

- ☐ [slaGetAllDocuments](#)
- ☐ [slaGetCornerModels](#)
- ☐ [slaGetCornerSetupCorners](#)
- ☐ [slaGetCornerSetups](#)
- ☐ [slaGetCornerVars](#)
- ☐ [slaGetDocumentAbsolutePath](#)
- ☐ [slaGetSweepSetupVars](#)
- ☐ [slaGetSweepSetups](#)
- ☐ [slaGetVerificationSpaces](#)

■ Functions for deleting components from the setup library view

- ☐ [slaRemoveCorner](#)
- ☐ [slaRemoveCornerModel](#)
- ☐ [slaRemoveCornerSetup](#)
- ☐ [slaRemoveCornerVariable](#)
- ☐ [slaRemoveSweepSetup](#)
- ☐ [slaRemoveSweepVariable](#)
- ☐ [slaRemoveVerificationSpace](#)

slaOpenOrCreateView

```
slaOpenOrCreateView(  
    t_libName  
    t_cellName  
    t_viewName  
    [ g_readOnly ]  
)  
=> t / nil
```

Description

When this function is run in edit mode, it opens the specified view in edit mode if the view exists. Otherwise, it creates a new view. In read-only mode, the function opens the provided view in read-only mode. In both cases, invoking this function checks out the ADE Verifier license. You can initialize the view from the Virtuoso CIW or by using the `*il` script.

Arguments

<i>t_libName</i>	String value specifying the library name.
<i>t_cellName</i>	String value specifying the cell name.
<i>t_viewName</i>	String value specifying the view name.
<i>g_readOnly</i>	Boolean value that specifies if the view must be opened in read-only mode or edit mode. Setting the value to <code>t</code> opens the cellview in read-only mode. The default is <code>nil</code> . This argument is optional.

Value Returned

<code>t</code>	The specified view has been opened.
<code>nil</code>	The specified view could not be opened.

Examples

Opens a setup library view in the edit mode in edit mode and checks out the Verifier license. It creates the setup library view if the given view does not exist.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t
```

Virtuoso ADE Verifier SKILL Reference

Setup Library Assistant Functions

Opens the setup library view in read-only mode and checks out the Verifier license.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib" t)  
=> t
```


slaAddCornerModelFile

```
slaAddCornerModelFile(  
    t_cornerSetupName  
    t_cornerName  
    l_list  
  
)  
=> t / nil
```

Description

Adds a model file to a corner within a corner setup in the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode. The function creates the specified corner if it does not exist in the corner setup.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup.
<i>t_cornerName</i>	String value specifying the name of the corner in which the model file must be added.
<i>l_list</i>	List containing the name of the model file and its sections.

Value Returned

<i>t</i>	The specified model file has been added to the specified corner.
<i>nil</i>	The specified model file could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified model file to the view. The function creates a corner, C0, if it does not exist in cornerSetup1.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaAddCornerModelFile("cornerSetup1" "C0" list("./model.scs" "ff"))  
=> t
```

slaAddCornerVariable

```
slaAddCornerVariable(  
    t_cornerSetupName  
    t_cornerName  
    t_varName  
    t_varValue  
)  
=> t / nil
```

Description

Adds a corner variable to a corner of a corner setup in the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode. The function creates the specified corner if it does not exist in the corner setup.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup.
<i>t_cornerName</i>	String value specifying the name of the corner in which the corner variable must be added.
<i>t_varName</i>	String value specifying the name of the variable to be added to the specified corner.
<i>t_varValue</i>	String value specifying the value of the variable to be added to the specified corner.

Value Returned

<i>t</i>	The specified corner variable has been added to the corner.
<i>nil</i>	The specified corner variable could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified corner variable to the view. The function creates a corner, C0, if it does not exist in *cornerSetup1*.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaAddCornerVariable("cornerSetup1" "C0" "vdd" "2.0:1.0:5.0")  
=> t
```

slaAddDocument

```
slaAddDocument(  
    t_docAbsolutePath  
)  
=> t / nil
```

Description

Adds a document to a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

t_docAbsolutePath

String value specifying the name and path of the document to be added.

Value Returned

t	The specified document has been added.
nil	The specified document could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified document to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaAddDocument("/home/project/xyz/coverage45.xls")  
=> t
```

slaAddSweepVariable

```
slaAddSweepVariable(  
    t_sweepSetupName  
    t_varName  
    t_varValue  
)  
=> t / nil
```

Description

Adds a sweep variable to a sweep setup in the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_sweepSetupName</i>	String value specifying the name of the sweep setup.
<i>t_varName</i>	String value specifying the name of the variable to be added to the sweep setup.
<i>t_varValue</i>	String value specifying the value of the variable to be added to the sweep setup.

Value Returned

<i>t</i>	The specified sweep variable was added to the specified setup.
<i>nil</i>	The specified sweep variable could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified sweep variable to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaAddSweepVariable("sweepSetup1" "vdd" "2.0:1.0:5.0")  
=> t
```

slaCreateCornerSetup

```
slaCreateCornerSetup(  
    t_setupName  
)  
=> t / nil
```

Description

Creates a corner setup in the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_setupName</i>	String value specifying the name of the corner setup to be created and added to the view.
--------------------	---

Value Returned

<i>t</i>	The specified corner setup has been created.
<i>nil</i>	The specified corner setup could not be created.

Examples

Opens a setup library view in the edit mode and adds the specified corner setup to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaCreateCornerSetup("cornerSetup1")  
=> t
```

slaCreateSweepSetup

```
slaCreateSweepSetup(  
    t_setupName  
)  
=> t / nil
```

Description

Creates a sweep setup in a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_setupName</i>	String value specifying the name of the corner setup to be added to the view.
--------------------	---

Value Returned

<i>t</i>	The specified sweep setup has been added.
<i>nil</i>	The specified sweep setup could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified sweep setup to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaCreateSweepSetup("sweepSetup1")  
=> t
```

slaCreateVerificationSpace

```
slaCreateVerificationSpace(  
    t_spaceName  
    t_sweepSetupName  
    t_cornerSetupName  
)  
=> t / nil
```

Description

Creates a verification space in the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_spaceName</i>	String value specifying the name of the verification space to be added.
<i>t_sweepSetupName</i>	String value specifying the name of the sweep setup to which the verification space is to be added.
<i>t_cornerSetupName</i>	String value specifying the name of the corner setup to which the verification space is to be added.

Value Returned

t	The specified verification space has been added.
nil	The specified verification space could not be added.

Examples

Opens a setup library view in the edit mode and adds the specified verification space to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaCreateVerificationSpace("space1" "sweepSetup1" "cornerSetup1")  
=> t
```

slaGetAllDocuments

```
slaGetAllDocuments(  
    )  
=> l_listOfDocuments
```

Description

Retrieves a list of documents from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

None.

Value Returned

l_listOfDocuments

A list of documents saved in the setup library view.

Examples

Opens a setup library view in the edit mode and adds the specified list of documents to the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetAllDocuments()  
=> ("coverage45nm.xls")
```


slaGetCornerModels

```
slaGetCornerModels(  
    t_cornerSetupName  
    t_cornerName  
)  
=> l_listOfCornerModels
```

Description

Retrieves a list of corner models of corner setup from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup.
<i>t_cornerName</i>	String value specifying the name of the corner that contains the corner models.

Value Returned

l_listOfCornerModels

A list of all corner models saved in the specified corner.

Examples

Opens a setup library view in the edit mode and retrieves the specified list of corner models from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetCornerModels("cornerSetup1" "C0")  
=> (("gpdk090.scs" "path/to/file/gpdk090.scs" "\"SF\" \"FS\""))
```

slaGetCornerSetupCorners

```
slaGetCornerSetupCorners(  
    t_cornerSetupName  
)  
=> l_listOfCorners
```

Description

Retrieves a list of corners for a specified corner setup from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

t_cornerSetupName String value specifying the name of the corner setup.

Value Returned

l_listOfCorners A list of corners saved in the setup library view.

Examples

Opens a setup library view in the edit mode and retrieves the specified list of corners from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetCornerSetupCorners("cornerSetup1")  
=> ("C0" "C1")
```

slaGetCornerSetups

```
slaGetCornerSetups (  
    )  
=> l_listOfCornerSetups
```

Description

Retrieves a list of corner setups from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

None.

Value Returned

l_listOfCornerSetups

A list of corner setups saved in the setup library view.

Examples

Opens a setup library view in the edit mode and retrieves the specified corners from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetCornerSetups()  
=> ("cornerSetup1" "cornerSetup2")
```

slaGetCornerVars

```
slaGetCornerVars(  
    t_cornerSetupName  
    t_cornerName  
)  
=> l_listOfCorners
```

Description

Retrieves a list of corner variables of the specified corner setup from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup within the setup library view.
<i>t_cornerName</i>	String value specifying the name of the corner that contains the corner variables.

Value Returned

<i>l_listOfCornerVariables</i>	A list of all corner variables saved in the specified corner of the corner setup.
--------------------------------	---

Examples

Opens a setup library view in the edit mode and retrieves the specified corner variables from the view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetCornerVars("cornerSetup1" "C0")  
=> (("gain" "11:1:12"))
```

slaGetDocumentAbsolutePath

```
slaGetDocumentAbsolutePath(  
    t_docName  
)  
=> t_docAbsolutePath
```

Description

Retrieves the absolute path of a document in the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_docName</i>	String value specifying the name of the document.
------------------	---

Value Returned

<i>t_docAbsolutePath</i>	Absolute path of the document in the setup library view.
--------------------------	--

Examples

Opens a setup library view in the edit mode and retrieves the absolute path of a specified document.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetDocumentAbsolutePath("coverage45.xls")  
=> "<setupLibrary_sdb_directory_path>/documents/coverage45.xls"
```

slaGetSweepSetupVars

```
slaGetSweepSetupVars(  
    t_sweepSetupName  
)  
=> l_listOfSweepVars
```

Description

Retrieves a list of sweep setup variables from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

t_sweepSetupName String value specifying the name of the sweep setup.

Value Returned

l_listOfSweepVars A list of all sweep variables saved in the specified sweep setup.

Examples

Opens a setup library view in the edit mode and retrieves the specified sweep variables from the specified view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetSweepSetupVars("sweepSetup1")  
=> (("gain" "15")  
    ("var1" "455")  
    )
```

slaGetSweepSetups

```
slaGetSweepSetups (  
    )  
=> l_listOfSweepSetups
```

Description

Retrieves a list of sweep setups from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

None.

Value Returned

l_listOfSweepSetups

A list of sweep setups saved in the setup library view.

Examples

The following example retrieves the specified sweep setups from the setup library view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetSweepSetups()  
=> ("sweepSetup1"  
    "sweepSetup2"  
    )
```

slaGetVerificationSpaces

```
slaGetVerificationSpaces (  
    )  
=> l_listOfLists
```

Description

Retrieves a list of verification spaces from the setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

None.

Value Returned

l_listOfLists A list of verification spaces saved in the setup library view.

Examples

The following example retrieves the specified sweep setups from the setup library view.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaGetVerificationSpaces()  
=> (("space1"  
    ("sweepSetup1"  
     "cornerSetup1")  
    )  
)
```


slaImportCorners

```
slaImportCorners(  
    t_cornerSetupName  
    t_absoluteFilePath  
)  
=> t / nil
```

Description

Imports corners into the setup library corner setup opened using the [slaOpenOrCreateView](#) function in edit mode. The file must be in CSV or SDB format.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup within the setup library view.
<i>t_absoluteFilePath</i>	String value specifying the absolute path of the file that contains the corners.

Value Returned

t	The specified corners were imported into the specified setup.
nil	The specified corners could not be imported.

Examples

Opens a setup library view in the edit mode and imports corners from the specified file.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaImportCorners("cornerSetup1" "path/to/file/corner.csv")  
=> t
```

slaImportSweeps

```
slaImportSweeps(  
    t_sweepSetupName  
    t_absoluteFilePath  
)  
=> t / nil
```

Description

Imports sweeps into the corner setup in a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode. The supported file extensions are `csv` and `sdb`.

Arguments

<i>t_sweepSetupName</i>	String value specifying the name of the sweep setup within the setup library view.
<i>t_absoluteFilePath</i>	String value specifying the absolute path of the file that contains the sweeps.

Value Returned

<i>t</i>	The specified sweeps were imported into the specified setup.
<i>nil</i>	The specified sweeps could not be imported.

Examples

Opens a setup library view in the edit mode and imports sweeps from the specified file.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaImportSweeps("sweepSetup1" "path/to/file/sweep.csv")  
=> t
```

slaRemoveCorner

```
slaRemoveCorner(  
    t_cornerSetupName  
    t_cornerName  
)  
=> t / nil
```

Description

Removes a corner from the corner setup in a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup within the setup library view.
<i>t_cornerName</i>	String value specifying the name of the corner to be removed.

Value Returned

<i>t</i>	The specified corner has been removed.
<i>nil</i>	The specified corner could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveCorner("cornerSetup1" "C0")  
=> t
```

slaRemoveCornerModel

```
slaRemoveCornerModel(  
    t_cornerSetupName  
    t_cornerName  
    t_modelName  
)  
=> t / nil
```

Description

Removes a corner model file from the corner setup in a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup within the setup library view.
<i>t_cornerName</i>	String value specifying the name of the corner that contains the model file to be removed.
<i>t_modelName</i>	String value specifying the name of the model file to be removed.

Value Returned

<i>t</i>	The specified corner model file has been removed.
<i>nil</i>	The specified corner model file could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner model file.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveCornerModel("cornerSetup1" "C0" "model.scs")  
=> t
```

slaRemoveCornerSetup

```
slaRemoveCornerSetup(  
    t_cornerSetupName  
)  
=> t / nil
```

Description

Removes a corner setup from a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

t_cornerSetupName String value specifying the name of the corner setup.

Value Returned

<i>t</i>	The specified corner setup has been removed.
<i>nil</i>	The specified corner setup could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveCornerSetup("cornerSetup1")  
=> t
```

slaRemoveCornerVariable

```
slaRemoveCornerVariable(  
    t_cornerSetupName  
    t_cornerName  
    t_varName  
)  
=> t / nil
```

Description

Removes a corner variable from corner setup of a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_cornerSetupName</i>	String value specifying the name of the corner setup.
<i>t_cornerName</i>	String value specifying the name of the corner from which the corner variable must be removed.
<i>t_varName</i>	String value specifying the name of the corner variable to be removed from the specified corner.

Value Returned

t	The specified corner variable was removed.
nil	The specified corner variable could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified corner variable.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveCornerVariable("cornerSetup1" "C0" "vdd")  
=> t
```

slaRemoveSweepSetup

```
slaRemoveSweepSetup(  
    t_sweepSetupName  
)  
=> t / nil
```

Description

Removes a sweep setup from a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_sweepSetupName</i>	String value specifying the name of the sweep setup within the setup library view.
-------------------------	--

Value Returned

<i>t</i>	The specified sweep setup was removed from the specified setup.
<i>nil</i>	The specified sweep setup could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified sweep setup.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveSweepSetup("sweepSetup1")  
=> t
```

slaRemoveSweepVariable

```
slaRemoveSweepVariable(  
    t_sweepSetupName  
    t_varName  
)  
=> t / nil
```

Description

Removes a specified sweep variable from the corner setup of a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_sweepSetupName</i>	String value specifying the name of the sweep setup.
<i>t_varName</i>	String value specifying the name of the sweep variable to be removed from the sweep setup.

Value Returned

t	The specified sweep variable has been removed.
nil	The specified sweep variable could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified sweep variable.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveSweepVariable("sweepSetup1" "vdd")  
=> t
```


slaRemoveVerificationSpace

```
slaRemoveVerificationSpace(  
    t_spaceName  
)  
=> t / nil
```

Description

Removes a verification space from a setup library view that is opened using the [slaOpenOrCreateView](#) function in edit mode.

Arguments

<i>t_spaceName</i>	String value specifying the name of the verification space.
--------------------	---

Value Returned

<i>t</i>	The specified verification space was removed.
<i>nil</i>	The specified verification space could not be removed.

Examples

Opens a setup library view in the edit mode and removes the specified sweep variable.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")  
=> t  
slaRemoveVerificationSpace("space1")  
=> t
```

slaSaveAndCloseView

```
slaSaveAndCloseView (  
    [ ?overrideDM g_overrideDM ]  
    [ ?autoCheckInNewView g_autoCheckInNewView ]  
    [ ?checkInComment g_checkInComment ]  
)  
=> t / nil
```

Description

Saves the specified view after editing. If a setup library view has been opened with the [slaOpenOrCreateView](#) function, the `slaSaveAndCloseView` function must be invoked after adding or removing components in the setup library assistant to save all the changes in the setupdb. Invoking the `slaSaveAndCloseView()` function releases or checks in the ADE Verifier license.

Argument

`?overrideDM` *g_overrideDM*

Boolean value that checks if the view can be checked in automatically in a DM setup. When set to the default value of `nil`, it checks in the view if the DM allows it. In such a case, the view is checked in automatically without a prompt. Otherwise, the view is not checked in and remains unmanaged. The DM settings are overridden when set to `t`.

This argument is optional.

`?autoCheckInNewView` *g_autoCheckInNewView*

Boolean value that when set to `t`, ignores the DM settings and checks in the view with or without comments. The default is `nil`.

This argument is optional.

`?checkInComment` *g_checkInComment*

String value that specifies any comments.

This argument is optional.

Virtuoso ADE Verifier SKILL Reference

Setup Library Assistant Functions

Value Returned

t	The specified view was saved.
nil	The specified view could not be saved.

Examples

The following example saves and closes the open setup library view and checks in or releases the Verifier license.

```
slaOpenOrCreateView("bertlink" "osc13" "setupLib")
=> t
slaSaveAndCloseView()
=> t
```

Virtuoso ADE Verifier SKILL Reference

Setup Library Assistant Functions

Snapshot Functions

Use the following SKILL functions to manage snapshots:

- [verifAreSnapshotsEnabled](#)
- [verifCreateSnapshot](#)
- [verifCreateSnapshotConfiguration](#)
- [verifDeleteSnapshot](#)
- [verifDeleteSnapshotConfiguration](#)
- [verifExportSnapshotsToExcel](#)
- [verifGetReferenceSnapshot](#)
- [verifGetSnapshot](#)
- [verifGetSnapshotAbsoluteTolerance](#)
- [verifGetSnapshotComment](#)
- [verifGetSnapshotRelativeTolerance](#)
- [verifGetSnapshots](#)
- [verifGetSnapshotsData](#)
- [verifIsSnapshotLocked](#)
- [verifIsSnapshotVisible](#)
- [verifRenameSnapshot](#)
- [verifRestoreFromSnapshot](#)
- [verifSetReferenceSnapshot](#)
- [verifSetSnapshotAbsoluteTolerance](#)
- [verifSetSnapshotComment](#)

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

- verifSetSnapshotConfiguration
- verifSetSnapshotLocked
- verifSetSnapshotRelativeTolerance
- verifSetSnapshotVisible
- verifSetSnapshotsEnabled

verifAreSnapshotsEnabled

```
verifAreSnapshotsEnabled(  
    g_sessionId  
)  
=> t / nil
```

Description

Checks if snapshots are enabled in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

t	Snapshots are enabled in the specified Verifier session.
nil	Snapshots are disabled in the specified Verifier session, or the command is not successful.

Examples

The following example starts a Verifier session with a new cellview:

```
winId = deNewCellView("test" "snapshots" "verifier" "verifier" nil)  
INFO (VERIFIER-8215): Started Verifier session '0'.  
window:2  
uid = winId->verifSession  
0  
;; By default, the snapshots are disabled  
verifAreSnapshotsEnabled(uid)  
nil  
;; Enable the snapshots  
verifSetSnapshotsEnabled(uid t)  
t  
verifAreSnapshotsEnabled(uid)  
t
```

verifCreateSnapshot

```
verifCreateSnapshot(  
    g_sessionId  
    [ t_snapshotName ]  
    [ ?name t_name ]  
    [ ?prefix t_prefix ]  
    [ ?comment t_comment ]  
    [ ?visible g_visible ]  
    [ ?locked g_locked ]  
)  
=> t / nil
```

Description

Creates a new snapshot '*.snapz' that contains the following:

- An archive, stored in .json format, of the current requirements, mappings, and results as shown in *Results* tab
- A copy of the settings.v3 setup
- A copy of the results directory

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot. By default, the name is in the 'snap_<date and time>' format. Note: This argument cannot be used with the ?name argument.
?name <i>t_name</i>	Name of the snapshot. By default, the name is in the 'snap_<date and time>' format. Note: This argument cannot be used with the <i>tSnapshotName</i> argument.
?prefix <i>t_prefix</i>	Prefix of the snapshot. By default, the prefix is 'snap_'.
?comment <i>t_comment</i>	Comment added to the snapshot.
?visible <i>g_visible</i>	Visibility status of the snapshot.
?locked <i>g_locked</i>	Locked status of the snapshot.

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

Value Returned

<code>t</code>	Snapshot is created successfully in the specified Verifier session.
<code>nil</code>	Snapshot does not exist or Verifier does not have the permission to create a new snapshot, or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and create a new snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
```

;; Create a snapshot with the default naming format

```
verifCreateSnapshot(uid)
=> t
```

;; Create a snapshot with specified name

```
verifCreateSnapshot(uid "mySnap")
=> t
```

;; Create a snapshot automatically

```
verifRegisterCallback(
  lambda((sess sig args)
    when(sig == 'simulationsAdded
      verifSetSnapshotsEnabled(sess t)
      verifCreateSnapshot(sess ?prefix "sim" ?comment "Pre-simulation
snapshot")
    )
  )
)
```

verifCreateSnapshotConfiguration

```
verifCreateSnapshotConfiguration(  
    g_sessionId  
    t_configName  
    l_properties  
    [ ?absoluteTolerance x_absoluteTolerance ]  
    [ ?relativeTolerance x_relativeTolerance ]  
    [ ?compareVisible g_differences ]  
    [ ?hideEmpty g_hideEmpty ]  
  
    )  
=> t / nil
```

Description

Creates snapshot configuration that contains the setup for Abs Tolerance, Rel Tolerance and Show filters on the Snapshots tab.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_configName</i>	Name of the configuration.
<i>l_properties</i>	List of requirement properties that will be shown on the snapshots tree.
?absoluteTolerance <i>x_absoluteTolerance</i>	Integer or floating-point number specifying absolute tolerance.
?relativeTolerance <i>x_relativeTolerance</i>	Integer or floating-point number specifying relative tolerance.
?differences <i>g_differences</i>	Shows the difference in property values, if enabled. The default is nil.
?compareVisible <i>g_compareVisible</i>	Compares the visible property values, if enabled. The default is nil.
?hideEmpty <i>g_hideEmpty</i>	Hides the empty property values, if enabled. The default is nil.

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

Value Returned

t	Snapshot configuration is created successfully.
nil	Command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and creates a new snapshot configuration:

```
uid = verifOpenCellView("test" "snapshots" "verifier")
=> 0
;; Create a snapshot configuration and show the properties
verifCreateSnapshotConfiguration(uid "config" list("Title" "Type"
"OverallStatus"))
;; Create a snapshot configuration and only show different properties
verifCreateSnapshotConfiguration(uid "diff" list("Title" "Type" "OverallStatus")
?differences t)
```

verifDeleteSnapshot

```
verifDeleteSnapshot(  
    g_sessionId  
    t_snapshotName  
)  
=> t / nil
```

Description

Deletes the snapshot from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.

Value Returned

<i>t</i>	Snapshot is deleted successfully in the specified Verifier session.
<i>nil</i>	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and deletes the snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
;; Delete the snapshot with specified name  
verifGetSnapshots(uid)  
("active" "snap_2019_07_08_11_25_26" "mySnap")  
verifDeleteSnapshot(uid "snap_2019_07_08_11_25_26")  
=> t  
verifGetSnapshots(uid)  
("active" "mySnap")  
;; The 'active' snapshot cannot be deleted  
verifDeleteSnapshot(uid "active")  
=> nil
```

verifDeleteSnapshotConfiguration

```
verifDeleteSnapshotConfiguration(  
    g_sessionId  
    t_configName  
)  
=> t / nil
```

Description

Deletes the snapshot configuration with the specified name.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_configName</i>	Name of the configuration.

Value Returned

<i>t</i>	Snapshot configuration is deleted successfully in the specified Verifier session.
<i>nil</i>	Snapshot configuration does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and deletes the snapshot configuration:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0
```

```
;; Delete the snapshot configuration 'diff'  
verifDeleteSnapshotConfiguration(uid "diff")  
=> t
```

```
;; Delete an nonexistent snapshot configuration  
verifDeleteSnapshotConfiguration(uid "config")
```

```
*WARNING* (VERIFIER-5006): verifDeleteSnapshotConfiguration : Cannot delete  
snapshots configuration 'config' as it does not exist in the current session.
```

```
=> nil
```

verifExportSnapshotsToExcel

```
verifExportSnapshotsToExcel(  
    g_sessionId  
    t_fileName  
    g_exportAll  
)  
=> t / nil
```

Description

Deletes the snapshot configuration with the specified name.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_fileName</i>	Name of the Excel file.
<i>g_exportAll</i>	Boolean to specify whether all the snapshot data or only visible properties need to be exported to the Excel file.

Value Returned

t	Export of snapshots to Excel file is successful.
nil	The command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and exports snapshots to an Excel file:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
;; Export the visible snapshots data with format to Excel file  
verifExportSnapshotsToExcel(uid "mySnapshots")  
INFO (VERIFIER-10019): Beginning export of snapshots to Excel file  
'mySnapshots.xlsx'...  
=> t  
INFO (VERIFIER-10020): Successfully exported snapshots to Excel file  
'mySnapshots.xlsx'.  
;; Export all the snapshots data with format to Excel file  
verifExportSnapshotsToExcel(uid "mySnapshots_all" t)  
INFO (VERIFIER-10019): Beginning export of snapshots to Excel file  
'mySnapshots.xlsx'...
```

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

```
=> t  
INFO (VERIFIER-10020): Successfully exported snapshots to Excel file  
'mySnapshots.xlsx'.
```

verifGetReferenceSnapshot

```
verifGetReferenceSnapshot(  
    g_sessionId  
)  
=> t_refSnapshotName / nil
```

Description

Retreives the snapshot from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>t_refSnapshotName</i>	Name of the reference snapshot.
nil	Command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and retrieves the reference snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
;; Retrieves the snapshot with specified name  
verifGetReferenceSnapshot(uid)  
"active"
```


Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

verifGetSnapshot

```
verifGetSnapshot(  
    g_sessionId  
    t_snapshotName  
)  
=> l_data / nil
```

Description

Retrieves the disembodied property list for snapshot from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.

Value Returned

<i>l_data</i>	Disembodied property list for snapshot.
<i>nil</i>	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and retrieves the disembodied property list for snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifGetSnapshot(uid "snap_2019_07_08_11_25_26")  
(nil comment "" customProperties (nil)  
date "Jul 8 14:57:32 2019" locked nil name  
"snap_2019_07_08_11_25_26" requirements nil user "tom"  
visible t  
)
```

verifGetSnapshotAbsoluteTolerance

```
verifGetSnapshotAbsoluteTolerance(  
    g_sessionId  
)  
=> f_value / nil
```

Description

Returns the absolute tolerance value used when comparing snapshot values.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>f_value</i>	Floating point value of the tolerance.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session and returns the tolerance:

```
sess = verifOpenCellView("mylib" "mycell" "verifier")  
=> 0  
verifGetSnapshotAbsoluteTolerance(sess)  
=> 1.34
```

verifGetSnapshotComment

```
verifGetSnapshotComment(  
    g_sessionId  
    t_snapshotName  
)  
=> t_comment / nil
```

Description

Returns the comment text for a snapshot.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.

Value Returned

<i>t_comment</i>	Comment text.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session and returns the comments of a snapshot:

```
sess = verifOpenCellView("mylib" "mycell" "verifier")  
=> 0  
verifGetSnapshotComment(sess "snap1")  
=> "The first snapshot"
```

verifGetSnapshotRelativeTolerance

```
verifGetSnapshotRelativeTolerance(  
    g_sessionId  
)  
=> f_value / nil
```

Description

Returns the relative tolerance value used when comparing snapshot values.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>f_value</i>	Floating point value of the tolerance.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session and returns the tolerance:

```
sess = verifOpenCellView("mylib" "mycell" "verifier")  
=> 0  
verifGetSnapshotRelativeTolerance(sess)  
=> 2.54
```

verifGetSnapshots

```
verifGetSnapshots(  
    g_sessionId  
)  
=> l_snapshots / nil
```

Description

Retrieves the list of snapshots from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
--------------------	--

Value Returned

<i>l_snapshots</i>	List of snapshots.
nil	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and retrieves the list of snapshots:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifGetSnapshots(uid)  
("active" "snap_2019_07_08_11_25_26" "mySnap")
```

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

verifGetSnapshotsData

```
verifGetSnapshotsData(  
    g_sessionId  
    [ g_showAll ]  
)  
=> l_snapshotsData / nil
```

Description

Retrieves the disembodied property list for snapshots from the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_showAll</i>	Boolean that controls whether to show all details, that is, all contents of every snapshot.

Value Returned

<i>l_snapshotsData</i>	Disembodied property list for snapshots.
<i>nil</i>	Snapshot does not exist or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and retrieves disembodied property list for snapshots:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifGetSnapshotsData(uid)  
(nil activeConfig "" configs nil  
enabled nil reference "mySnap" snaps  
((nil comment "Active requirements snapshot data" date "Jul 8 15:57:46 2019"  
locked t name "active" user "tom" visible t  
)  
(nil comment "" date "Jul 8 13:59:58 2019" location "snapshots/  
snap_2019_07_08_11_25_26/snapshot.json" locked nil name "mySnap" user "tom"  
visible t)  
(nil comment "" date "Jul 8 14:57:32 2019" location "snapshots/  
snap_2019_07_08_11_25_26/snapshot.json" locked nil name "snap_2019_07_08_11_25_26"  
user "tom" visible t)  
)
```

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

)

verifIsSnapshotLocked

```
verifIsSnapshotLocked(  
    g_sessionId  
    t_snapshotName  
)  
=> t / nil
```

Description

Returns the locked status of the snapshot.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot to query.

Value Returned

t	Snapshot is locked.
nil	Snapshot is not locked or the command is not successful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and queries if a snapshot is locked:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifIsSnapshotLocked(sess "snap1")  
=> t  
verifIsSnapshotLocked(sess "snap2")  
=> nil  
verifIsSnapshotLocked(sess "invalid-snap")  
=> t  
*WARNING* (VERIFIER-10003): Cannot find snapshot 'invalid-snap' as there is no  
snapshot with this name.  
Use a valid name and try again.  
=> nil
```


verifIsSnapshotVisible

```
verifIsSnapshotLocked(  
    g_sessionId  
    t_snapshotName  
)  
=> t / nil
```

Description

Returns the visibility status for the specified snapshot.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.

Value Returned

t	Snapshot is visible.
nil	Snapshot is not visible or the command is unsuccessful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and returns the visibility of a snapshot:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifIsSnapshotVisible(sess "snap1")  
=> t  
verifIsSnapshotVisible(sess "snap2")  
=> nil  
verifIsSnapshotVisible(sess "invalid-snap")  
*WARNING* (VERIFIER-10003): Cannot find snapshot 'invalid-snap' as there is no  
snapshot with this name.  
Use a valid name and try again.  
=> nil
```

verifRenameSnapshot

```
verifRenameSnapshot(  
    g_sessionId  
    t_snapshotName  
    t_newSnapshotName  
)  
=> t / nil
```

Description

Renames a snapshot with the specified name.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.
<i>t_newSnapshotName</i>	New name of the snapshot.

Value Returned

t	Snapshot is renamed successfully.
nil	Snapshot does not exist or the command is unsuccessful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and renames the specified snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifRenameSnapshot(uid "snap_2019_07_08_11_25_26" "mySnap_1")  
=> t
```

verifRestoreFromSnapshot

```
verifRestoreFromSnapshot(  
    g_sessionId  
    t_snapshotName  
)  
=> t / nil
```

Description

Restores a Verifier session from a snapshot along with any stored results.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.

Value Returned

t	Snapshot is restored successfully.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and restores a snapshot:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifRestoreFromSnapshot(sess "snap1")  
=> t
```

verifSetReferenceSnapshot

```
verifSetReferenceSnapshot(  
    g_sessionId  
    t_refSnapshotName  
)  
=> t / nil
```

Description

Sets the reference snapshot in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_refSnapshotName</i>	Name of the reference snapshot.

Value Returned

t	Snapshot is set successfully.
nil	Command is unsuccessful.

Examples

The following example starts a Verifier session with a cellview 'test/snapshots/verifier' and sets the reference snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifSetReferenceSnapshot(uid "mySnap")  
=> t
```

verifSetSnapshotAbsoluteTolerance

```
verifSetSnapshotAbsoluteTolerance(  
    g_sessionId  
    f_value  
)  
=> t / nil
```

Description

Sets the absolute tolerance for snapshot comparison in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>f_value</i>	Floating point number for the tolerance.

Value Returned

t	Tolerance is set successfully.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and sets the reference snapshot:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifSetSnapshotAbsoluteTolerance(sess 1.2)  
=> t
```

verifSetSnapshotComment

```
verifSetSnapshotComment(  
    g_sessionId  
    t_snapshotName  
    t_comment  
)  
=> t / nil
```

Description

Sets the comment for a snapshot in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.
<i>t_comment</i>	Comment for the snapshot.

Value Returned

t	Comment is set successfully.
nil	Snapshot does not exist or command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and sets the comment for the specified snapshot:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifSetSnapshotComment(uid "snap_2019_07_08_11_25_26" "This is a comment")  
=> t
```

verifSetSnapshotConfiguration

```
verifSetSnapshotConfiguration(  
    g_sessionId  
    t_configName  
)  
=> t / nil
```

Description

Saves the current *Show* list configuration with the specified name and filters the *Snapshots* tab items with the specified configuration.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_configName</i>	Name of the configuration.

Value Returned

t	Current snapshot configuration is set successfully.
nil	Snapshot configuration does not exist or command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and sets the current snapshot configuration:

```
uid = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
;; Set the current snapshot configuration to 'diff'  
verifSetSnapshotConfiguration(uid "diff")  
=> t  
;; Set the current snapshot configuration to the nonexistent one  
verifSetSnapshotConfiguration(uid "config")  
*WARNING* (VERIFIER-5006): verifSetSnapshotConfiguration : Cannot set current  
snapshot configuration to 'config' as it does not exist in the current session or  
it is already the current configuration.  
=> nil
```

verifSetSnapshotLocked

```
verifSetSnapshotLocked(  
    g_sessionId  
    t_snapshotName  
    g_isLocked  
)  
=> t / nil
```

Description

Locks or unlocks the specified snapshots in the given Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.
<i>g_isLocked</i>	Boolean to specify whether the snapshot should be locked or not.

Value Returned

t	Snapshot is locked or unlocked.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and then locks and unlocks a snapshot:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifSetSnapshotLocked(sess "snap1" t)  
=> t  
verifSetSnapshotLocked(sess "snap1" nil)  
=> t
```


verifSetSnapshotRelativeTolerance

```
verifSetSnapshotRelativeTolerance(  
    g_sessionId  
    f_value  
)  
=> t / nil
```

Description

Sets the relative tolerance for snapshot comparison in a Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>f_value</i>	Floating point number for the tolerance.

Value Returned

t	Tolerance is set successfully.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and sets its tolerance:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifSetSnapshotRelativeTolerance(sess 1.2)  
=> t
```

verifSetSnapshotVisible

```
verifSetSnapshotVisible(  
    g_sessionId  
    t_snapshotName  
    g_isVisible  
)  
=> t / nil
```

Description

Sets the visibility of a snapshot in a Verifier session. When made visible, it appears as a column in the *Snapshots* Tab.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>t_snapshotName</i>	Name of the snapshot.
<i>g_isVisible</i>	Boolean to specify whether the snapshot should be made visible or not.

Value Returned

t	Snapshot is visible or hidden.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and toggles the visibility of a snapshot:

```
sess = verifOpenCellView("test" "snapshots" "verifier")  
=> 0  
verifSetSnapshotVisible(sess "snap1" t)  
=> t  
verifSetSnapshotVisible(sess "snap1" nil)  
=> t
```

verifSetSnapshotsEnabled

```
verifSetSnapshotsEnabled(  
    g_sessionId  
    g_enabled  
)  
=> t / nil
```

Description

Enables or disables snapshots in the specified Verifier session.

Arguments

<i>g_sessionId</i>	Integer, string number, or window specifying the Verifier session ID. For example, 0, "0", or window(2).
<i>g_enabled</i>	State of 'Enabled'.

Value Returned

t	Enables snapshots in the specified Verifier session.
nil	Command is unsuccessful.

Examples

The following example opens a Verifier session with a cellview 'test/snapshots/verifier' and change the 'Enabled' check state:

```
winId = deNewCellView("test" "snapshots" "verifier" "verifier" nil)  
INFO (VERIFIER-8215): Started Verifier session '0'.  
window:2  
uid = winId->verifSession  
=> 0  
;; By default, snapshots are disabled  
verifAreSnapshotsEnabled(uid)  
=> nil  
;; Enable the snapshots  
verifSetSnapshotsEnabled(uid t)  
=> t  
verifAreSnapshotsEnabled(uid)  
=> t
```

Virtuoso ADE Verifier SKILL Reference

Snapshot Functions

Debugging Functions

Use the following SKILL functions to debug commands:

- verifEnableDebug
- verifDisableDebug
- verifGetDebug

verifEnableDebug

```
verifEnableDebug(  
    [ t_string ]  
)  
=> t / nil
```

Description

Enables additional debug logging of various categories.

Arguments

<i>t_string</i>	A string of space-delimited categories that can be enabled., Possible values are: all, db, default, detailedperformance, gui, importexport, performance, simulation, and vmanager. Note: When enabling categories related to performance, the recommendation is to enable the performance category instead of detailedperformance. Enabling the detailedperformance category shows large amount of information, and must be used in rare cases.
-----------------	---

Value Returned

t	Debugging is successfully enabled.
nil	Operation is unsuccessful.

Examples

The following examples describe the possible scenarios:

Enables default debugging category (for example, db, gui, and simulation)

```
verifEnableDebug()  
=> t
```

Enables the db and gui categories

```
verifEnableDebug("db gui")  
=> t
```

Errors out to indicate that the debugging operation has failed because an invalid argument was passed.

```
verifEnableDebug("foo")
```

```
*WARNING* (VERIFIER-5022): Cannot enable category 'foo' because it is not a valid  
category name. Select a category name from 'all db default detailedperformance gui  
importexport performance simulation vmanager' and try again.
```

Virtuoso ADE Verifier SKILL Reference

Debugging Functions

=> nil

verifDisableDebug

```
verifDisableDebug(  
    [ t_string ]  
)  
=> t / nil
```

Description

Disables one or more categories for debug logging.

Arguments

<i>t_string</i>	A string of space-delimited categories that can be disabled. Possible values are: all, db, default, detailedperformance, gui, importexport, performance, simulation, and vmanager.
-----------------	--

Value Returned

t	Debugging is successfully disabled.
nil	Operation is unsuccessful.

Examples

The following examples describe the possible scenarios:

Enables default debugging with the `importexport` category (for example, `db`, `gui`, and `simulation`)

```
verifEnableDebug("default importexport")  
=> t
```

Disables the `importexport` category

```
verifDisableDebug("importexport")  
=> t
```

Disables the `importexport` category again

```
verifDisableDebug("importexport")  
=> nil
```

Disables debugging by giving no arguments

```
verifDisableDebug()  
verifEnableDebug("foo")
```

```
*WARNING* (VERIFIER-5022): Cannot disable category 'foo' because it is not a valid  
category name. Select a category name from 'all db default detailedperformance gui
```


Virtuoso ADE Verifier SKILL Reference

Debugging Functions

```
importexport performance simulation vmanager' and try again.  
=> nil
```

verifGetDebug

```
verifGetDebug(  
    [ g_all ]  
)  
=> t_categories / nil
```

Description

Gets the names of enabled debug categories or all possible categories.

Arguments

<i>g_all</i>	Returns all possible categories if non-zero, else returns only the enabled categories. Possible values are: all, db, default, detailedperformance, gui, importexport, performance, simulation, and vmanager.
--------------	--

Value Returned

<i>t_categories</i>	Returns a space-separated string consisting of all or enabled categories.
nil	No categories are enabled.

Examples

The following examples illustrate the possible scenarios:

Gets the currently enabled categories (by default none)

```
verifGetDebug()  
=> nil
```

Gets all of the available categories

```
verifGetDebug(t)  
=> "all db default detailedperformance gui importexport performance simulation  
vmanager"
```

Enables and then gets those categories

```
verifEnableDebug("db gui simulation")  
=> t  
verifGetDebug()  
=> "db gui simulation"
```

Disables the 'gui' category

Virtuoso ADE Verifier SKILL Reference

Debugging Functions

```
verifDisableDebug("gui")  
verifGetDebug()  
=> "db simulation"
```

Virtuoso ADE Verifier SKILL Reference

Debugging Functions

Additional Information

This section illustrates the following information:

- Examples:
 - ❑ [Custom Function to Copy Implementation Units to Requirements](#)
 - ❑ [Callback Function to Save Verifier Cellviews Automatically](#)
 - ❑ [Callback Function to Automatically Create Snapshots for Each Simulation Run](#)
 - ❑ [Callback Function to Automatically Create Snapshots for Backup](#)
 - ❑ [Function to Customize Menu Banners](#)
 - ❑ [Function to Customize Context Menus](#)
- [Removed SKILL functions](#)

Custom Function to Copy Implementation Units to Requirements

The following code illustrates a custom function to get the unit and mapping information from each output of an implementation cellview in a Verifier session. It then sets the unit in the requirements mapped to the output.

Note: This custom function does not copy the specification values set in the implementations to the requirements. To copy the specifications and units of the implementations to the mapped requirements, use the *Override Verifier Specification* feature.

```
;; Overwrite the unit of requirements that are mapped to the specified
implementation
;; NOTE: This is not support for implementation with 'Run Plan' mode
procedure(copyUnitToReq(sessionId impLib impCell impView impHistory "gt")
  let((reqIds list()) mappedReqs unit)
    when(verifIsSessionReadOnly(sessionId)
      error("Cannot overwrite the unit of requirement because of read-only
        verifier session")
    )
    ;; Iterate tests in the implementation
    foreach(testName verifGetImpTests(sessionId impLib impCell impView
      impHistory)
      ;; Iterate outputs in the test
      foreach(outputName verifGetImpTestOutputs(sessionId impLib impCell
        impView impHistory testName)
        ;; Get the list of requirements that are mapped to the output
        mappedReqs = verifGetImpMapping(sessionId impLib impCell impView
          impHistory ?testName testName ?outputName outputName)
        when(listp(mappedReqs) && mappedReqs
          ;; Get unit of output
          unit = verifGetImpData(sessionId impLib impCell impView
            impHistory ?testName testName ?outputName outputName
            ?dataName "unit")
          when(stringp(unit) && unit != ""
            ;; Set unit to all the mapped requirements
            foreach(reqId mappedReqs
              when(verifSetReqProp(sessionId reqId "Unit" unit)
                reqIds = append1(reqIds reqId))))))
    reqIds)
```

Copy the given above code and paste it in Virtuoso CIW to register the procedure. Then, run the following function in Virtuoso CIW:

Virtuoso ADE Verifier SKILL Reference

Additional Information

```
copyUnitToReq(sesssionId impLib impCell impView impHistory)
```

For example:

```
sess = verifOpenCellView("test" "sample" "verifier")  
=> 0  
copyUnitToReq(sess "")
```

Callback Function to Save Verifier Cellviews Automatically

The following code illustrates a custom callback function that saves the active Verifier cellviews periodically. You can add such a function in your `.cdsinit` file to save your cellviews automatically.

```
;; Callback function to save all the active Verifier cellviews automatically.
;; Specify the save interval in seconds as argument, like:
;;     autoSaveVerifierCellView(300)
;; This will save all opened Verifier cellviews every 5 minutes.
;; To interrupt the auto saving, set the global variable
;;     _stopAutoSaveVerifier = t
procedure(autoSaveVerifierCellView(saveInterval "x")
    ;; Get all opened Verifier sessions
    foreach(sessionId verifGetAllSessions()
        ;; Don't save for the read-only session
        unless(verifIsSessionReadOnly(sessionId)
            verifSaveSession(sessionId)))

    if((boundp('_stopAutoSaveVerifier) && _stopAutoSaveVerifier) then
        printf("Auto save interrupted. Set _stopAutoSaveVerifier = nil and start
        autoSaveVerifierCellView %d again.\n" saveInterval)
    else
        hiRegTimer(sprintf(nil "autoSaveVerifierCellView(%d)" saveInterval)
            saveInterval*10)
        printf("...done. Next auto save scheduled in %d seconds.\n"
            saveInterval))
```


Callback Function to Automatically Create Snapshots for Each Simulation Run

The following code illustrates a custom callback function that automatically creates a snapshot for every simulation. You can add such a function in your `.cdsinit` file to automatically create a snapshot of the session before the simulation starts.

```
;; Callback function to save create snapshots automatically for each simulation
run.
verifRegisterCallback(
    lambda((sess sig args)
        when(sig == 'simulationsAdded
            verifSetSnapshotsEnabled(sess t)
            verifCreateSnapshot(sess ?prefix "sim" ?comment "Pre-simulation
snapshot")
        )
    )
)
```

Callback Function to Automatically Create Snapshots for Backup

The following code illustrates a custom callback function that automatically creates a snapshot for backup purposes. You can add such a function in your `.cdsinit` file to automatically create a snapshot of the session.

```
;; Callback function to save create snapshots automatically for each simulation
run.

;; Callback function to auto save all active Verifier cellviews.
;; Specify the save interval in seconds as argument, like:
;;   AutoCreateSnapshotForCellView("test" "run" "verifier")
;;   AutoCreateSnapshotForCellView("test" "run" "verifier")
;;   AutoCreateSnapshotForSession(sessionId 120)
;;   AutoCreateSnapshotForSession(sessionId)
;;   AutoCreateSnapshot(120 hiGetCurrentWindow())
;;   AutoCreateSnapshot(120)
;;   AutoCreateSnapshot()
;; This will auto create the snapshot for that cellview every 5 minutes.
;; To interrupt the auto creation, set the global variable
;;   StopCreateSnapshot = t
;;

procedure (AutoCreateSnapshotForCellView(lib cell view @optional (interval 300)
"tttx")
  let((sess)
    when(_verifIsValidSession(sess = verifGetCellViewSession(lib cell view))
      printf("Starting to create snapshot automatically...\n")
      when(verifCreateSnapshot(sess)
        printf("Created new snapshot '%s' for Verifier cellview '%s'.\n"
cadr(reverse(verifGetSnapshots(sess))) buildString(verifGetSessionCellView(sess)
"/"))
      )
      if(!(boundp('StopAutoCreateSnapshot) && StopAutoCreateSnapshot) then
        hiRegTimer(sprintf(nil "AutoCreateSnapshotForCellView(\"%s\" \"%s\"
\"%s\" %d)" lib cell view interval) interval*10)
        printf("Next auto creation scheduled in %d seconds.\n" interval)
      else
        printf("Auto creation interrupted. Call
unbindVar(StopAutoCreateSnapshot) and start AutoCreateSnapshotForCellView(<lib>
<cell> <view> 300) again.\n")
      )
    )
  )
)
```

Virtuoso ADE Verifier SKILL Reference

Additional Information

```
procedure (AutoCreateSnapshotForSession(sess @optional (interval 300) "gx")
  when(_verifIsValidSession(sess)
    printf("Starting to create snapshot automatically...\n")
    when(verifCreateSnapshot(sess)
      printf("Created new snapshot '%s' for Verifier cellview '%s'.\n"
cadr(reverse(verifGetSnapshots(sess))) buildString(verifGetSessionCellView(sess)
"/"))
    )
    if(!(boundp('StopAutoCreateSnapshot) && StopAutoCreateSnapshot) then
      hiRegTimer(sprintf(nil "AutoCreateSnapshot(%d)" interval) interval*10)
      printf("Next auto creation scheduled in %d seconds.\n" interval)
    else
      printf("Auto creation interrupted. Call
unbindVar(StopAutoCreateSnapshot) and start AutoCreateSnapshotForSession(<sess>
300) again.\n")
    )
  )
)

procedure (AutoCreateSnapshot(@optional (interval 300) (win hiGetCurrentWindow())
"zg")
  let((sess)
    when(_verifIsValidSession(sess = win->verifSession)
      printf("Starting to create snapshot automatically...\n")
      when(verifCreateSnapshot(sess)
        printf("Created new snapshot '%s' for Verifier cellview '%s'.\n"
cadr(reverse(verifGetSnapshots(sess))) buildString(verifGetSessionCellView(sess)
"/"))
      )
      if(!(boundp('StopAutoCreateSnapshot) && StopAutoCreateSnapshot) then
        hiRegTimer(sprintf(nil "AutoCreateSnapshot(%d)" interval)
interval*10)
        printf("Next auto creation scheduled in %d seconds.\n" interval)
      else
        printf("Auto creation interrupted. Call
unbindVar(StopAutoCreateSnapshot) and start AutoCreateSnapshot(300) again.\n")
      )
    )
  )
)
```

Function to Customize Menu Banners

The following code illustrates menu customization in Verifier. The customization can be placed in different locations. See [Customizing the Menu Banner](#).

Menu files are read in the following order:

```
your_install_dir/etc/tools/menus/appName.menus
your_install_dir/local/menus/appName.menus
workOrProjectArea/menus/appName.menus
~/menus/appName.menus
```

For example, you can place the attached file in `workOrProjectArea/menus/verifier.menus` and start a new Virtuoso session and launch Verifier. You find a new custom menu entry with two actions.

```
;; Verifier menu customization example
verifier.menus:

;; Load the existing Verifier menus in order to add to it instead of replacing it
loadi(prependInstallPath("etc/tools/menus/verifier.menus"))
;; Define functions to be called by new menu entries
procedure(VerifPrintSess(sess)
    printf("%L\n" sess)
)
procedure(VerifPrintReq(sess)
    let((reqIDs)
        reqIDs = verifGetReqs(sess)
        foreach(reqID reqIDs
            printf("%L -- \t%L \t%L--%L\n"
                reqID
                verifGetReqProp(sess reqID "Title")
                verifGetReqProp(sess reqID "MinSpec")
                verifGetReqProp(sess reqID "MaxSpec")
            )
        )
    )
)
;; Define the new menu entries
myMenu = ' (MyMenu "CustomMenu"
    (
        (PrintSess "Print Session Number" "VerifPrintSess(hiGetCurrentWindow()->verifSession)")
        (PrintReq "Print Requirements" "VerifPrintReq(hiGetCurrentWindow()->verifSession)")
    )
)
;; Add new menus to the banner menus
verifMenus = append1(verifMenus 'myMenu)
```

Function to Customize Context Menus

You can define customized context menu entries in ADE Verifier by including any of the variables —`verifImpContextMenu`—`verifReqContextMenu`—`verifRunContextMenu`—`verifResultsMenu`— in the `verifier.menus` file. By using any of these environment variables, you can create custom context menus in the *Implementations* or *Requirements* panes of the *Setup* tab or in the *Run* and *Results* tabs. Any definitions are added at the end of the respective context menu. For more information, see [Function to Customize Menu Banners](#). You define these variables as a list of items that can be another list, a symbol, or a string in the following format:

```
(id text tool-tip entry [enabled-fn] [show-fn])
```

Here,

- `id` is the symbol identifier,
- `text` denotes `<menu text>` with its corresponding `<tool-tip>`,
- `entry` is either a string with a SKILL callback, a symbol with the name of a function, or a list defining a sub-menu.
- `enabled-fn` and `shown-fn` are optional callbacks that specify if the items should be enabled and shown.

For example, you can place the following code in `work Or ProjectArea/menus/verifier.menus` and launch Verifier in a new Virtuoso session. You then find the new menu actions added to the context menus.

Virtuoso ADE Verifier SKILL Reference

Additional Information

```
; The menu entry is a list of items that is a list, a string, or a symbol.
;
; If it is a string, a separator is placed at the menu entry location, and the
; contents of the string are ignored.
;
; If it is a symbol, then the menu entry is a reference to an existing menu entry/
; definition that can be defined elsewhere(for example, as myAbc and mySlider).
;
; If it is a list, then a menu action is created using the contained definition,
; which needs to be in the following format:
;
; (id menu-name menu-tooltip menu-entry menu-tooltip [menu-enabled-fn]
; [menu-item-shown-fn]).
;
; menu-name is the text that is displayed in the menu, and menu-tooltip is its
; corresponding tooltip.
;
; The menu-entry can be one of the following:
;   a) a string: It is evaluated as a callback and no arguments are passed on
;       to the callback.
;   b) a symbol: It is the name of a callable object (i.e. function-name or
;       lambda function) which takes two arguments - session, and selected.
;   c) a list: It defines the contents of a pull-right menu.
;
; The menu-enabled-fn is either nil, a string or a callable function which if
; non-nil, is evaluated to determine if the menu is enabled or not.
;
; The menu-item-shown-fn is either nil, a string or a callable function. If it is
; non-nil, it is evaluated to determine if the menu items are shown in the context
; menu or not. For example, you can use it to only show an item if there is at
; least one item selected, or only show an entry if a single implementation is
; selected .
;
; If the menu or the enabled callback function are not defined, or if the enabled
; function returns an error, then the menu is disabled.
;
; The menu-callback, enable-callback, and show-callback functions should take the
; following two arguments:
;     a session-number + list-of-selected-items.
;
;
info("Loading custom menus...\n")

custom1 = '(custom1 "My Impl 1" "Custom 1 Tooltip" doImpl1CB1)

;; Defines the custom context menu for the Implementations pane in the Setup tab.
verifImpContextMenu =
'(
  custom1
  (custom2 "My Impl 2" "Do more custom stuff" "doImplCB2()" doImplCB2Enabled
isCustom2Shown)
  "----" ; separator
  (slider1 "Slider" "Slide right for extra stuff"
    (
      (item1 "Item1" "Item1 Tooltip" doImplCB1 item1Enabled)
    )
    isSlider1Enabled
  )
  slider2
)
```

Virtuoso ADE Verifier SKILL Reference

Additional Information

```
slider2 = '(slider2 "Slider 2" "Yet another Slider"
                ((item3 "Item3" "Item3 Tooltip" doImplCB1))
                nil showSlider2)

;; Defines the custom context menu for the Requirements pane in the Setup tab.
verifReqContextMenu =
'(
    (req1 "My Req Item" "Do req stuff" doReqCB doReqEnabled isReqShown)
)

;; Defines the custom context menu for the run table in the Run tab.
verifRunContextMenu =
'(
    (run1 "My Run Item" "Do run stuff" doRunCB doRunEnabled isRunShown)
)

;; Defines the custom context menu for the results table in the Results tab.
verifResultsContextMenu =
'(
    (res1 "My Res Item 1" "Do res stuff" doResCB doResEnabled isResShown)
)

info("...finished loading custom menus.\n")
```

Any callback takes two arguments: (session-number list-of-selected-names).

The following section describes the various callback functions:

Virtuoso ADE Verifier SKILL Reference

Additional Information

```
; Defines all the various callbacks for the custom menus in the Requirement, Run
; and Results hierarchies
defun( doImplCB1 (sess sel)
  info("doImplCB1 CB session: %L selected set: %L\n" sess sel)
)
defun( doImplCB2 ()
  info("doImplCB2 CB called with no arguments")
)
defun( doImplCB2Enabled (sess sel)
  ;; custom2 is enabled only if more than 2 items are selected.
  length(sel)>2
)
defun( isImpl2Shown (sess sel)
  ;; customer2 is shown only if more than 1 item is selected.
  length(sel)>1
)
defun( item1Enabled (sess sel)
  length(sel)>2
)
defun( isSlider1Enabled (sess sel)
  ;; slider1 is always enabled.
  t
)
defun( showSlider2 (sess sel)
  ;; slider2 is visible only if more than 3 items are selected.
  length(sel)>3
)

defun( doReqCB (sess sel)
  info("Req CB session: %L selected set: %L\n" sess sel)
)
defun( doRunCB (sess sel)
  info("Run CB session: %L selected set: %L\n" sess sel)
)
defun( doResCB (sess sel)
  info("Res CB session: %L selected set: %L\n" sess sel)
)
defun( doReqEnabled (sess sel)
  info("Req Enabled session: %L selected set: %L\n" sess sel)
  t
)
defun( doRunEnabled (sess sel)
  info("Run Enabled session: %L selected set: %L\n" sess sel)
  t
)
defun( doResEnabled (sess sel)
  info("Res Enabled session: %L selected set: %L\n" sess sel)
  t
)
defun( isReqShown (sess sel)
  info("Is Req shown session: %L selected set: %L\n" sess sel)
  t
)
defun( isRunShown (sess sel)
  info("Is Run shown session: %L selected set: %L\n" sess sel)
  t
)
defun( isResShown (sess sel)
  info("Is Res shown session: %L selected set: %L\n" sess sel)
  t
)
```


Removed SKILL functions

This section lists the SKILL functions that have been replaced by new or enhanced SKILL functions, or are no longer supported in Verifier:

Function	Replaced by
Implementation Functions	
Data Extraction Functions	
verifGetAllImps	verifGetImps
verifGetImpSimRunData	verifGetResultDataForImp
verifGetImpSimRunDataByName	verifGetResultDataForImp
verifGetImpTestOutputData	verifGetImpData
verifGetImpTestOutputDataByName	verifGetImpData
verifGetImpTestOutputs	verifGetImpTestOutputs
verifGetImpTests	verifGetImpTests
verifGetImpsByCell	verifGetImps
verifGetImpsByHistory	verifGetImps
verifGetImpsByLib	verifGetImps
verifGetImpsByView	verifGetImps
Implementation List Management Functions	
verifAddImp	verifAddImp
verifCheckAllImps	verifCheckForChanges
verifDeleteImps	verifRemoveImp
verifMoveImp	verifMoveImp
verifPullSpecFromImplementation	verifOverwriteSpec
verifPushSpecToImplementation	verifOverwriteSpec
verifReloadAllRes	verifReloadAllRes

Virtuoso ADE Verifier SKILL Reference

Additional Information

Requirement-Implementation Mapping Functions

Mapping Functions

verifGetAllMappingForReq	verifGetReqMapping
verifGetMappableReqs	verifGetReqs
verifGetMappedImpRunsForReq	verifGetReqMapping
verifGetMappedImpTestOutputsForReq	verifGetReqMapping
verifGetMappedImpTestsForReq	verifGetReqMapping
verifGetMappedImpsForReq	verifGetReqMapping
verifGetMappedReqs	verifGetReqs
verifGetMappedReqsForImp	verifGetImpMapping
verifGetMappedReqsForImpTest	verifGetImpMapping
verifGetMappedReqsForImpTestOutput	verifGetImpMapping
verifGetUnmappableReqs	verifGetReqs
verifGetUnmappedReqs	verifGetReqs
verifMapping	verifMapping

Requirements Functions

Addition and Deletion Functions

verifAddReq	verifAddReq
verifAddSubReq	verifAddReq
verifDeleteReq	verifRemoveReq

Custom Fields Functions

verifGetCustomFieldNames	verifGetCustomFieldNames
verifGetCustomFieldValue	verifGetCustomFieldValue
verifGetReqCustomFieldNames	verifGetReqCustomFieldNames
verifGetReqCustomFieldValue	verifGetReqCustomFieldValue
verifSetCustomFieldValue	verifSetCustomFieldValue
verifSetReqCustomFieldValue	verifSetReqCustomFieldValue
verifGetAllReqProps	verifGetReqProps

Virtuoso ADE Verifier SKILL Reference

Additional Information

Data Extraction Functions

verifGetReqProps	verifGetReqProps
verifGetAllReqs	verifGetReqs
verifGetReqHier	verifGetReqParent
verifGetReqPropByName	verifGetReqProp
verifGetReqStatus	verifGetReqStatus
verifGetReqsByGoalType	verifGetReqs
verifGetReqsByGroup	verifGetReqs
verifGetReqsByOwner	verifGetReqs

Export and Import Functions

verifCompareImportedFiles	verifCompareImportedFiles
verifGetImportedFiles	verifGetImportedFiles
verifMappingAndResExportToImportedFiles	
	verifExportReqsToFile
verifMergeImportedFiles	verifMergeImportedFiles
verifReqsExportToFile	verifExportReqsToFile
verifReqsImportFromCSVFile	verifImportFile
verifReqsImportFromExcelFile	verifImportFilefileType

Manual Sign-off Functions

verifDeleteSignoff	verifDeleteReqSignoff
verifGetSignoffInfo	verifGetReqSignoff
verifSignoff	verifSignOffReq

Requirement Setup Functions

verifCreateRandomID	verifCreateRandomId
verifGetReferencedCellViews	verifGetReferencedCellViews
verifModifyReqId	verifSetReqId
verifModifyReqTitle	verifSetReqTitle
verifSetReqGoalType	verifSetReqGoalType
verifSetupReqData	verifSetReqProp

Virtuoso ADE Verifier SKILL Reference

Additional Information

verifUpdate

verifUpdate

Simulation and Results Extraction Functions

Implementation Set Functions

verifAddImpToRunGroup

verifAddImpToImpSet

verifAddRunGroup

verifAddImpSet

verifDeleteImpFromRunGroup

verifRemoveImpSet

verifDeleteRunGroup

verifRemoveImpSet

verifGetAllRunGroups

verifGetImpSets

verifGetImpsInRunGroup

verifGetImpsInImpSets

Results Extraction Functions

verifGetResultDataForImp

verifGetResultDataForImp

verifGetResultDataForImpTest

verifGetResultDataForImp

verifGetResultDataForImpTestOutput

verifGetResultDataForImp

verifGetResultDataForReq

verifGetResultDataForReq

Simulation Functions

verifGetLoadImps

verifGetImps

verifGetRunImps

verifGetImps

verifIsImpRun

verifImpIsRun

verifRunAll

verifRun

verifRunGroup

verifRunImpSet

verifRunImps

verifRun

verifRunOrReloadAll

verifRun

verifRunOrReloadGroup

verifRunImpSet

verifRunOrReloadImps

verifRun

verifWaitUntilDone

verifRun

Verification Status and Reports Functions

HTML Report Functions

verifCreateHTMLReport

verifPublishHTML

Virtuoso ADE Verifier SKILL Reference

Additional Information

verifInitialHTMLReport	Removed
verifInsertHTMLEmptylines	Removed
verifInsertHTMLFormattedText	Removed
verifInsertHTMLImageFile	Removed
verifInsertHTMLResultTable	Removed
verifInsertHTMLRunTable	Removed
verifInsertHierarchyHTMLResultTable	Removed
verifInsertSignoffInfoTable	Removed
verifOpenHTMLReport	Removed
verifWriteHTMLToFile	Removed
Image Creation Functions	
verifCreateSchematicImage	Removed
verifCreateWaveformImage	Removed
Text Report Function	
verifCreateTextReport	Removed
Verification Status Functions	
verifDefineReturnStatusSet	verifGetReqs() combined with verifGetReqStatus()
verifGetOverallInformation	verifGetReqs() combined with verifGetReqStatus()
verifGetOverallStatus	verifGetReqs() combined with verifGetReqStatus()
XML Report Functions	
verifCreateXMLReport	Removed
verifFinishXMLReport	Removed
verifInitialXMLReport	Removed
verifInsertXMLResultInfo	Removed
verifInsertXMLRunInfo	Removed
verifInsertXMLUserInformation	Removed

Virtuoso ADE Verifier SKILL Reference

Additional Information

Verifier Session and Setup Functions

Batch Mode Functions

verifIsBatchMode	Removed
verifResetBatchMode	Removed
verifSetBatchMode	Removed

Menu Configuration Functions

verifDeregisterMenuItem	Removed
verifGetRegisteredMenus	Removed
verifRegisterMenuItem	Removed

Other Functions

verifGetErrors	Removed
verifRegisterEventTrigger	verifRegisterCallback

Preferences Functions

verifGetOptionVal	verifGetOptionVal
verifGetOptions	verifGetOptions
verifSetPreference	verifSetOptionVal

Startup Exit and Session Information Functions

verifCloseSession	verifCloseSession
verifGetActiveSession	Removed
verifGetAllActiveSessions	verifGetSessionIds
verifGetSessionCell	verifGetSessionCellView
verifGetSessionLib	verifGetSessionCellView
verifGetSessionView	verifGetSessionCellView
verifIsWritableMode	verifIsSessionReadOnly
verifStart	verifOpenCellView

Verifier Setup File and Cellview Functions

verifOpenCellview	verifOpenCell
verifSaveAsCellview	verifSaveSessionAs

Virtuoso ADE Verifier SKILL Reference

Additional Information

verifSaveAsFile	_verifSaveAsFile
verifSaveCellview	verifSaveSession
verifSaveFile	Removed

Deprecated Functions

verifAddVariantToRunGroup	verifCreateVariant
verifCreateVariantFromVariant	verifDeleteVariantFromRunGroup
verifDeleteVariants	verifEditVariant
verifGetAllVariants	verifGetVariantSetup
verifMoveVariant	