OCEAN XL Reference

Product Version ICADVM20.1
October 2020

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

Contents

Preface . . . 9
Scope . . . 10
Licensing Requirements . . . 10
Related Documentation . . . 10
What's New and KPNS . . . 10
Ocean Commands . . . 10
Installation, Environment, and Infrastructure . . . 11
Virtuoso Tools . . . 11
Additional Learning Resources . . . 11
Video Library . . . 11
Virtuoso Videos Book . . . 12
Rapid Adoption Kits . . . 12
Help and Support Facilities . . . 12
Customer Support . . . 13
Feedback about Documentation . . . 13
Understanding Cadence SKILL . . . 14
Using SKILL Code Examples . . . 14
Sample SKILL Code . . . 14
Accessing API Help . . . 15
Typographic and Syntax Conventions . . . 16
Identifiers Used to Denote Data Types . . . 17
Introduction to OCEAN XL Commands . . . 19
Creating and Running an OCEAN XL Script . . . 20
Creating an OCEAN XL Script . . . 20
Modifying an OCEAN XL Script . . . 21
Running an OCEAN Script . . . 23
Running Parallel OCEAN XL Simulation Runs for an ADE XL View . . . 24
Viewing Results of Simulations Run using OCEAN XL Scripts . . . 26

OCEAN Commands in XL Mode . . . 27
Commands to Set Up OCEAN XL Mode . . . 28
ocnxlEndXLMode . . . 29
ocnSetXLMode . . . 31
Commands for Tests . . . 33
ocnxlBeginTest . . . 34
ocnxlDisableTest . . . 35
ocnxlEndTest . . . 36
ocnxlGetTests . . . 37
ocnxlMTSBlock . . . 38
ocnxlSetMTSMode . . . 40
ocnxlMTSEnable . . . 42
ocnxlSelectTest . . . 43
Commands for Active Setup . . . 44
ocnxlDeleteNote . . . 45
ocnxlDisableSweepParam . . . 46
ocnxlDisableSweepVar . . . 47
ocnxlEnableSweepParam . . . 48
ocnxlEnableSweepVar . . . 49
ocnxlEnableTest . . . 50
ocnxlGetJobId . . . 51
ocnxlGetSession . . . 52
ocnxlJobSetup . . . 53
ocnxlLoadSetupState . . . 56
ocnxlMaxJobFail . . . 59
ocnxlMainSimSession . . . 60
ocnxlProjectDir . . . 61
ocnxlPutNote . . . 62
ocnxlResultsLocation . . . 63
ocnxlSaveSetupAs . . . 64
ocnxlSensitivityVars . . . 65
ocnxlSetDesignVariablePerTest . . . 66
ocnxlSetAllVarsDisabled . . . 67
ocnxlSetupLocation . . . 68
ocnxlSimResultsLocation . . . 69

ocnxlStimuliData . . . 70
ocnxlSweepParam . . . 71
ocnxlSweepVar . . . 72
ocnxlTargetCellView . . . 73
Commands for Corners . . . 75
ocnxlCorner . . . 76
ocnxlDisableCorner . . . 77
ocnxlDisableCornerForTest . . . 78
ocnxlEnableCorner . . . 79
ocnxlEnableCornerForTest . . . 80
ocnxlGetCorners . . . 81
ocnxlModelGroup . . . 82
Commands for Outputs . . . 83
ocnxlAddOrUpdateOutput . . . 84
ocnxlExportOutputView . . . 85
ocnxlOutputAreaGoal . . . 87
ocnxlOutputExpr . . . 88
ocnxlOutputMatExpr . . . 90
ocnxlOutputMatlabScript . . . 92
ocnxlOutputOceanScript . . . 93
ocnxlOutputOpRegion . . . 94
ocnxlOutputSignal . . . 95
ocnxlOutputSpiceScript . . . 96
ocnxlOutputTerminal . . . 97
ocnxlPutChecksAsserts . . . 98
ocnxlOutputViolations . . . 100
ocnxlPutChecksAssertsTest . . . 102
ocnxlPutEnabledChecksAssertsCellViews . . . 105
Commands for Specifications . . . 107
ocnxlGetSpecs . . . 108
ocnxlPutGreaterthanSpec . . . 109
ocnxlPutInfoSpec . . . 110
ocnxlPutLessthanSpec . . . 111
ocnxlPutMaxSpec . . . 112
ocnxlPutMinSpec . . . 113
ocnxlPutToleranceSpec . . . 114

ocnxlPutRangeSpec . . . 116
ocnxlRemoveSpec . . . 118
Commands to Set Up Run Options . . . 119
ocnxlConjugateGradientOptions . . . 120
ocnxlCornerVars . . . 121
ocnxlFeasibilityAnalysisOptions . . . 122
ocnxlGetBestPointParams . . . 124
ocnxlGlobalOptimizationOptions . . . 125
ocnxlLocalOptimizationOptions . . . 128
ocnxlMonteCarloOptions . . . 131
ocnxlSamplingOptions . . . 135
ocnxlSensitivityOptions . . . 136
ocnxlSizeOverCornersOptions . . . 139
ocnxlStartingPoint . . . 141
ocnxlSweepsAndCornersOptions . . . 142
ocnxlWorstCaseCornersOptions . . . 143
ocnxlYieldEstimationOptions . . . 144
ocnxlYieldImprovementOptions . . . 148
Commands for OCEAN XL Runs . . . 152
ocnxlGetRunDistributeOptions . . . 153
ocnxlOpenResults . . . 154
ocnxlOutputSummary . . . 155
ocnxlRun . . . 161
ocnxlRunSetupSummary . . . 165
ocnxlSetRunDistributeOptions . . . 166
ocnxlWaitUntilDone . . . 168
Commands for History . . . 171
ocnxlGetCurrentHistory . . . 172
ocnxlGetCurrentHistoryId . . . 174
ocnxlGetHistory . . . 175
ocnxlGetOverwriteHistory . . . 177
ocnxlGetOverwriteHistoryName . . . 178
ocnxlGetReferenceHistory . . . 179
ocnxlHistoryPrefix . . . 180
ocnxlRenameCurrentHistory . . . 181
ocnxlSetOverwriteHistory . . . 182

ocnxlSetOverwriteHistoryName . . . 183
ocnxlSetReferenceHistory . . . 184
ocnxlWriteDatasheet . . . 187
Commands for Parametric Sets . . . 190
ocnxlParametricSet . . . 191
ocnxlLocalParametricSet . . . 192
ocnxlSetAllParametersDisabled . . . 193
ocnxlSetAllParameterPSetsDisabled . . . 194
ocnxlSetAllVariablePSetsDisabled . . . 195
Commands for Pre-run Scripts . . . 196
ocnxlGetPointId . . . 197
ocnxlLoadCurrentEnvironment . . . 198
ocnxlMCIterNum . . . 200
ocnxlPreRunScript . . . 201
ocnxlRunCalibration . . . 202
ocnxlSetCalibration . . . 203
ocnxlSetMCdut . . . 204
ocnxlSetMCignore . . . 205
ocnxlSetPreRunScriptEnabled . . . 207
ocnxlUpdatePointVariable . . . 208
Commands for Reliability Analysis . . . 210
ocnxlAddRelxSetup . . . 211
ocnxlAddRelxScenariosSetup . . . 213
ocnxlDisableRelxScenariosSetup . . . 215
ocnxlDisableRelxSetup . . . 216
ocnxlSetRelxAnalysisEnabled . . . 217
ocnxlSetRelxScenariosEnabled . . . 218
ocnxlSetRelxEnabledForPreRun . . . 219
Commands for EAD . . . 220
ocnxlEADAddMeasurement . . . 221
ocnxlEADCreateDataSet . . . 222
ocnxlEADEnableLiveProcessing . . . 224
ocnxlEADSelectAllSignals . . . 226
ocnxlEADSetDutMaster . . . 227
ocnxlEADSetHierarchyLevel . . . 228
ocnxlEADSetWaveFormClipping . . . 229

Preface

Open Command Environment for Analysis (OCEAN) XL commands described in this manual let you set up, simulate, and analyze circuit data without starting Virtuoso Analog Design Environment XL. This manual assumes that you are familiar with analog design and simulation using Virtuoso Analog Design Environment XL.

The preface discusses the following:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions
- Identifiers Used to Denote Data Types

Scope

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

Label	Meaning
(ICADVM20.1 Only)	Features supported only in ICADVM20.1 advanced nodes and advanced methodologies releases.
(IC6.1.8 Only)	Features supported only in mature node releases.

Licensing Requirements

The OCEAN XL commands can be run for Virtuoso® Analog Design Environment XL, Virtuoso® ADE Explorer, or Virtuoso® ADE Assembler. Depending on the mode set in the ocnxlSetXLMode command, any one of the following licenses is checked out to run an OCEAN XL script:

- Analog_Design_Environment_XL (95210)
- Virtuoso_ADE_Explorer (95250)
- Virtuoso_ADE_Assembler (95260)

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

Related Documentation

What's New and KPNS

- *Virtuoso Analog Design Environment XL What's New*
- *Virtuoso Analog Design Environment XL Known Problems and Solutions*

Ocean Commands

OCEAN Reference.

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*.
- *Virtuoso Design Environment User Guide*.
- *Cadence SKILL Language User Guide*
- *Cadence SKILL Language Reference*
- *Cadence SKILL++ Object System Reference*
- *Virtuoso Design Environment SKILL Reference*
- *Virtuoso Design Environment SKILL Reference*
- *Virtuoso Analog Design Environment L SKILL Language Reference*
- *Virtuoso Analog Design Environment XL SKILL Language Reference*

Virtuoso Tools

- *Virtuoso Analog Design Environment L User Guide*
- *Virtuoso Analog Design Environment XL User Guide*
- *Virtuoso Analog Design Environment GXL User Guide*
- *Virtuoso Analog Distributed Processing Option User Guide*

Additional Learning Resources

Video Library

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

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

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

Virtuoso Videos Book

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

Rapid Adoption Kits

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

In addition, Cadence offers the following training courses on the SKILL programming language, which you can use to customize, extend, and automate your design environment:

- SKILL Language Programming Introduction
- SKILL Language Programming
- Advanced SKILL Language Programming

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

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

Help and Support Facilities

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

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

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the

Home button on the Cadence Help toolbar (provided you have not set a custom home page).

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

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

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

- Log on to Cadence Online Support

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

Feedback about Documentation

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

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

You can also submit feedback by using the following methods:

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

Understanding Cadence SKILL

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

For more information about the SKILL language, see Getting Started in the *SKILL Language User Guide*.

Using SKILL Code Examples

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

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

Sample SKILL Code

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

axlGetRunStatus

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

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

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.

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type exactly as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
`|`	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.
`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument. Used with brackets to indicate that you can specify zero or more arguments. Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence® SKILL® language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

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

Identifiers Used to Denote Data Types

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

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

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

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

1

Introduction to OCEAN XL Commands

OCEAN XL commands let you simulate and analyze your design in Virtuoso® Analog Design Environment XL. In this chapter, you can find information about creating and running OCEAN XL scripts.

Creating and Running an OCEAN XL Script

You can create and run an OCEAN XL script to run one or more tests over zero or more corner conditions. When you save an OCEAN XL script, the program saves setup information for all your tests, sweeps, corner conditions and run options.

For more details, refer to the following sections:

- Creating an OCEAN XL Script
- Modifying an OCEAN XL Script
- Running an OCEAN Script
- Running Parallel OCEAN XL Simulation Runs for an ADE XL View
- Viewing Results of Simulations Run using OCEAN XL Scripts

Creating an OCEAN XL Script

You can create an OCEAN XL script in any of the following three ways:

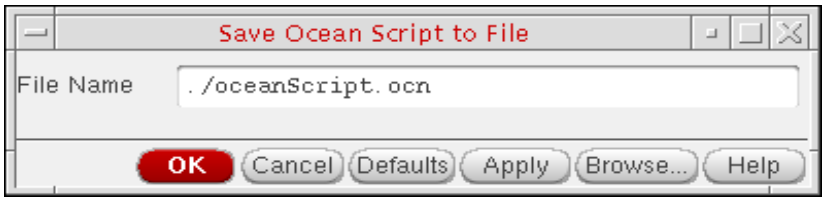
- ➡ Saving the ADE XL Setup to an OCEAN XL Script
- ➡ Scripting an OCEAN XL Script
- ➡ Creating OCEAN XL Scripts with Netlist Specified as a Design

Saving the ADE XL Setup to an OCEAN XL Script

You can create a simulation setup in ADE XL user interface and save it as an OCEAN XL script. For this, after the simulation setup is complete, do the following:

1. In the main session window, choose *File – Save Script*.

   The Save OCEAN Script form appears.



2. In the *File Name* field, type a name and location for your OCEAN XL script file.

Note: The default name and location is `./oceanScript.ocn`.

3. Click *OK*.

   The program saves simulation setup and conditions to the specified OCEAN XL script file.

Note: The setup information for non-ADE XL tests will not be saved in the OCEAN XL script.

Scripting an OCEAN XL Script

You can use a text editor to create an OCEAN XL script by using OCEAN XL commands and save it in a `.ocn` file.

Creating OCEAN XL Scripts with Netlist Specified as a Design

If you have an already generated netlist of a design, you can provide the path to that netlist file to the `design` OCEAN XL command. By doing this, you can run a simulation without loading/ requiring the complete design hierarchy. Later, you can view the simulation results in ADE XL user interface. For more details, refer to Viewing Results of Simulations Run using OCEAN XL Scripts.

Modifying an OCEAN XL Script

If you run an OCEAN XL script from an ADE XL interface, if required, you can load its setup in ADE XL user interface and modify it after the run. To modify the OCEAN XL setup, do the following:

1. In the History tab of the Data View assistant pane, right-click the history item for the OCEAN XL run and choose *Load Setup to Active*.

Caution

Load Setup to Active will overwrite the current setup with the setup in the history item for the OCEAN XL run.

2. Modify the setup as required.

   You can modify the setup using the ADE XL user interface (such as adding or modifying global variables or corners using the Data View assistant pane), or use the OCEAN Test Editor form to modify the setup.

   To use the OCEAN Test Editor form, do the following:

a. Right-click the test name in the *Tests* tree in the Data tab of Data View assistant pane and choose *Open Test Editor*.

The OCEAN Test Editor form appears in which the OCEAN commands saved in the script are displayed.

The OCEAN Test Editor form is a text editor specified by using the `editor` SKILL variable, or the `$VISUAL` or `$EDITOR` variables in the `.cshrc` file, in the given order of preference.

b. Change the simulation setup by modifying the commands.

c. Click *OK* to save the changes in the script.

Important

When you modify the setup in the history item for the OCEAN XL run in ADE XL, the original OCEAN XL script that you used to run the simulation from a UNIX shell will not be modified. Choose *File – Save Script* if you want to save the changes in the setup to a new OCEAN XL script file.

Running an OCEAN Script

To run simulations using OCEAN scripts, perform any one of the following steps:

➡ In CIW, run the following command:

```
load("<name-of-OCEAN-script-file>")
```

➡ In ADE XL user interface, load the set up from history of a previous OCEAN run and click *Run Simulation* on the Run toolbar to start the simulation.

➡ In a UNIX shell, type the following commands:

```
ocean
load("<name-of-OCEAN-script-file>")
```

For information about running an OCEAN script from a UNIX shell, see the *OCEAN Reference*.

When you run a simulation using an OCEAN script, a history item named `Ocean.n` is created for the run. You can then start ADE XL, and use the ADE XL user interface to view the results for the OCEAN run.

Important

You must not load and run your OCEAN script in the Command Interpreter Window (CIW) while ADE XL is still running for the same cellview.

When you run the script, the program reports the following information:

- Sweep parameters and their values
- Number of tests, sweep points, and corners
- Points completed and job status information
- Results location to the output area of the CIW

For example:

```
1/1 completed.
*Info*   The result of this OCEAN XL run are saved in "Interactive.3" in library
"rfExamples", cell "ne600", view "adexl".
```

The results location corresponds to the lib/cell/view specified in the ocnxlTargetCellView call, such as

```
ocnxlTargetCellView( "rfExamples" "ne600" "adexl")
```

Tip

You can run an OCEAN script using LSF, for example, by submitting the job remotely using the `bsub` command (for the LSF example) as follows:

```
bsub ocean -nograph OCEANscriptFileName
```

Also see: includeSimLogInJobLog, an environment that can be used to control the inclusion of the simulator output log in the OCEAN job log.

Running Parallel OCEAN XL Simulation Runs for an ADE XL View

In OCEAN XL, you can simultaneously run multiple simulation runs for an ADE XL view. You can do this by executing multiple ocnxlRun functions in parallel from the same OCEAN XL script. The results of all the runs are saved in the results database of that ADE XL view.

This section describes how to prepare setup and run parallel simulation runs in OCEAN XL.

Preparing Setup for Parallel Simulation Runs

You can enable the parallel run option for your OCEAN XL scripts in any of the following ways:

- To enable the parallel run, before saving an OCEAN XL script from an ADE XL view, select the *Parallel* option in the **Run Options** form.
- Use the ocnxlSetRunDistributeOptions command in the OCEAN XL script to set the parallel run option.

Running Parallel Simulation Runs

In your OCEAN script, ensure that the *`waitUntilDone`* argument of the ocnxlRun function is set to `nil`.

For example, if you have saved an ADE XL state, `Ac_State1`, and want to run a simulation for it parallel to other runs in your OCEAN XL script, use the following commands:

```
ocnxlLoadSetupState( "AC_State1" 'retain ?tests t ?vars t ?parameters t
?currentMode t ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid1 = ocnxlRun(?waitUntilDone nil)

ocnxlLoadSetupState( "Tran_State2" 'retain ?tests t ?vars t ?parameters t
?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid2 = ocnxlRun(?waitUntilDone nil)
```

When the *`waitUntilDone`* argument is set to `nil`, OCEAN XL does not wait for the run to complete. Instead, while the run is in progress, it executes the next line in the script. In the example given above, OCEAN XL loads another state, `Tran_State2`, and starts a simulation for that without waiting for the first simulation run to finish.

Note: You need to set the *`waitUntilDone`* argument for each OCEAN XL run that you want to run in parallel. By default, this argument is set to `t`.

When you set the *`waitUntilDone`* argument to `nil`, the ocnxlRun function also returns a run ID for each run. You can use this run ID later to either wait for that run to complete at a later stage or to access the history results of that run.

For example, before printing results, if you want to wait for a particular OCEAN XL run to complete simulations, you can use the following statement in your script:

```
(ocnxlWaitUntilDone rinid2)

(ocnxlOutputSummary ?forRun runid1 ?detailed nil)

; The previous command displays run summary for the simulation run with run ID as
runid1
```

After a run is complete, you can get access to its results by using its run ID, as shown in the following statement:

```
h1 = ocnxlGetHistory(runid1)
```

You can use this handle to get results for the given OCEAN XL run.

For details on the related OCEAN XL functions and their examples, refer to the following sections in the OCEAN Reference Guide:

- ocnxlRun
- ocnxlOutputSummary
- ocnxlWaitUntilDone
- ocnxlGetHistory
- ocnxlSetRunDistributeOptions

- ocnxlGetRunDistributeOptions

Viewing Results of Simulations Run using OCEAN XL Scripts

The results of simulations run from OCEAN XL scripts are saved in the results database of the ADE XL view. While the OCEAN XL simulations are in progress or after they are complete, you can view these results in the ADE XL user interface.

To view the results for an OCEAN XL simulation run in the ADE XL user interface, do the following:

➡ In the History tab of the Data View assistant pane, right-click the history item for the OCEAN run and choose *View Results*.

Note: While the simulations are in progress, you need to close and reopen the results to view the updated data.

When you view the results of simulations performed by specifying netlist as design, you cannot use the schematic-based post processing options. For example, you cannot use the `VT` calculator function to select a net on schematic or you cannot use the direct plotting feature. Therefore, when you right-click data in the *Results* tab, some of the commands that require use of the schematic view are not enabled.

2

OCEAN Commands in XL Mode

This chapter provides details of the OCEAN XL commands related to the following areas:

- Commands to Set Up OCEAN XL Mode
- Commands for Tests
- Commands for Active Setup
- Commands for Corners
- Commands for Outputs
- Commands for Specifications
- Commands to Set Up Run Options
- Commands for OCEAN XL Runs
- Commands for History
- Commands for Parametric Sets
- Commands for Pre-run Scripts
- Commands for Reliability Analysis
- Commands for EAD

Commands to Set Up OCEAN XL Mode

- ocnxlEndXLMode
- ocnSetXLMode

ocnxlEndXLMode

```
ocnxlEndXLMode(
     [ t_maestroMode ]
     )
     => t / nil
```

Description

This command indicates the end of the current XL mode. This command releases the license that was checked out for the XL mode.

Arguments

t_maestroMode

Specifies the application for which the XL mode is to be ended.

Possible values:

- `"explorer"`: Ends the OCEAN XL mode for ADE Explorer.
- `"assembler"`: Ends the OCEAN XL mode for ADE Assembler.

Any other value ends the OCEAN XL mode for ADE XL.

Default value: `nil`

Important

Ensure that the value given for this argument is same as the mode in the ocnSetXLMode command. If the two values do not match, the checked out license will not be released.

Value Returned

`t`	Returns `t` if the test setup completes.
`nil`	Returns `nil` otherwise.

Example

```
ocnSetXLMode()
ocnxlBeginTest("test_one")
```

```
design("solutions" "ampTest" "schematic")
simulator('spectre)
...
ocnxlEndTest()
```

ocnSetXLMode

```
ocnSetXLMode(
     [ t_maestroMode ]
     )
     => t / nil
```

Description

Sets the OCEAN mode to XL. In this mode, the simulation setup can contain multiple tests, corners, or advanced run modes, such as MonteCarlo. The OCEAN XL mode can be used to run simulations for ADE XL, ADE Explorer, or ADE Assembler. Accordingly, it checks out the required license. Once the mode is set to XL, it cannot be reverted.

Arguments

t_maestroMode

Specifies the application for which simulations are to be run.

Possible values:

- `"explorer"`: Sets the OCEAN XL mode for ADE Explorer and checks out the Virtuoso_ADE_Explorer (95250) license. If 95250 is not available, Virtuoso_ADE_Assembler (95260) can also be checked out.
- `"assembler"`: Sets the OCEAN XL mode for ADE Assembler and checks out the Virtuoso_ADE_Assembler (95260) license.

Any other value sets the OCEAN XL mode for ADE XL and checks out the Analog_Design_Environment_XL (95210) license.

Default value: `nil`

Value Returned

`t`	Returns `t` if the mode is set to XL.
`nil`	Returns `nil` otherwise

.Example

```
ocnSetXLMode()
ocnxlBeginTest("test_one")
```

```
design("solutions" "ampTest" "schematic")
simulator('spectre)
...
ocnxlEndTest()
```

Commands for Tests

- ocnxlBeginTest
- ocnxlDisableTest
- ocnxlEndTest
- ocnxlGetTests
- ocnxlMTSBlock
- ocnxlSetMTSMode
- ocnxlMTSEnable
- ocnxlSelectTest

ocnxlBeginTest

```
ocnxlBeginTest(
     t_testName
     )
     => t / nil
```

Description

This command indicates the beginning of the test specified by `testName`. Subsequent commands populate this test. The test specification ends when `ocnxlEndTest()` command is given.

Arguments

`t_testName`	The name of the test.

Value Returned

`t`	Returns `t` if its able to begin the test.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlBeginTest("test_one")
```

ocnxlDisableTest

```
ocnxlDisableTest(
     t_testName
     )
     => t / nil
```

Description

Lets you disable a test. A disabled test will not be run when `ocnxlRun()` command is fired. See help on `ocnxlRun()`.

Arguments

t_testName	Name of the test.

Value Returned

`t`	Returns `t` if the test is disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlDisableTest("test_two") => t
```

ocnxlEndTest

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

Description

This command indicates the end of the current test's specification.

Arguments

None.

Value Returned

`t`	Returns `t` if the test setup completes.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlBeginTest("test_one")
design("solutions" "ampTest" "schematic")
simulator('spectre)
ocnxlEndTest()
```

ocnxlGetTests

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

Description

Returns a list of test names.

Arguments

None

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGetTests()
```

ocnxlMTSBlock

```
ocnxlMTSBlock(
     s_blockName
     [ ?isMtsBlock  g_isMtsBlock ]
     [ ?includeFile l_includeFile ]
     [ ?modelFiles  l_modelFiles ]
     [ ?simOptions t_simOptions ]
     )
     => t / nil
```

Description

Enables a block for multi-technology simulation (MTS) and specifies the include files and model files associated with the block.

Arguments

s_blockName	Specifies the name of the block that needs to be enabled for multi-technology simulation. Valid values: a string
?isMtsBlock *g_isMtsBlock*	Specifies whether the block is enabled or disabled for multi-technology simulation. Valid values: `t`, `nil` Default Value: `nil`
`?includeFile` *l_includeFile*	Specifies the include files associated with the block. Valid values: a list of strings or `nil` Default value: `nil`
`?modelFiles` *l_modelFiles*	Specifies the model files associated with the block. Valid values: a list of strings or `nil` Default value: `nil`
`?simOptions` *t_simOptions*	Specifies simulator options.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlMTSBlock(‘digLib\ inv_usim
?isMtsBlock t
?modelFiles ’(("Models/myModels.scs" "ss")
("Models/spectre_cl013lv.scs" "aa"))
```

Enables the `inv_usim` cell in the `digLib` library for multi-technology simulation and specifies the model files (and the sections of the model files) associated with the block.

ocnxlSetMTSMode

```
ocnxlSetMTSMode(
     s_blockName
     [ ?enableMTS g_enableMTSBlock ]
     [ ?modelFiles  l_modelFiles ]
     [ ?simOptions t_simOptions ]
     )
     => t / nil
```

Description

Enables a block for multi-technology simulation (MTS) and specifies the model files associated with the block.

Arguments

s_blockName	Specifies the name of the block that needs to be enabled for multi-technology simulation. Valid values: a string
?enableMTS *g_enableMTSBlock*	Specifies whether the block is enabled or disabled for multi-technology simulation. Valid values: `t`, `nil` Default Value: `nil`
`?modelFiles` *l_modelFiles*	Specifies the model files associated with the block. Valid values: a list of strings or `nil` Default value: `nil`
`?simOptions` *t_simOptions*	Specifies simulator options.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlMTSEnable(t "assembler") ocnxlSetMTSMode("design_45 inv" ?enableMTS t
?modelFiles '(("gpdk045/gpdk045_v_3_0/models/spectre/gpdk045.scs" "tt"))
?simOptions '(("scale" "2") ("temp" "25")))
```

Enables the `inv` cell in the `design_45` library for multi-technology simulation and specifies the model files associated with the block.

ocnxlMTSEnable

```
ocnxlMTSEnable(
     g_enable
     )
     => t / nil
```

Description

Enables or disables multi-technology simulation (MTS) mode for the current test. The current test's specification begins with `ocnxlBeginTest()`. See help on `ocnxlBeginTest()`.

Arguments

g_enable	Enables or disables MTS mode for the current test. Valid values: `t`, `nil` Default Value: `nil`

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlMTSEnable(t)
;Enables MTS mode for the current test.
```

ocnxlSelectTest

```
ocnxlSelectTest(
     t_testName
     )
     => t / nil
```

Description

Lets you select a test. List of test names can be obtained by `ocnxlGetTests()` command. See help on `ocnxlGetTests()`.

Arguments

t_testName	The name of the test.

Value Returned

`t`	Returns `t` if the test is selected.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSelectTest("test_two") => t
Sets "test_two" as the currently selected test.
```

Commands for Active Setup

- ocnxlDeleteNote
- ocnxlDisableSweepParam
- ocnxlDisableSweepVar
- ocnxlEnableSweepParam
- ocnxlEnableSweepVar
- ocnxlEnableTest
- ocnxlGetJobId
- ocnxlGetSession
- ocnxlJobSetup
- ocnxlLoadSetupState
- ocnxlMaxJobFail
- ocnxlMainSimSession
- ocnxlProjectDir
- ocnxlPutNote
- ocnxlResultsLocation
- ocnxlSaveSetupAs
- ocnxlSensitivityVars
- ocnxlSetDesignVariablePerTest
- ocnxlSetAllVarsDisabled
- ocnxlSetupLocation
- ocnxlSimResultsLocation
- ocnxlStimuliData
- ocnxlSweepVar
- ocnxlSweepParam
- ocnxlTargetCellView

ocnxlDeleteNote

```
ocnxlDeleteNote(
     t_item
     t_name
     )
     => t / nil
```

Description

Deletes a note from the given test, history, corner, or variable.

For more information about notes, see Adding Notes to a Test.

Arguments

t_item	Type of the element from which you need to delete a note. Valid values: `"test"`, `"history"`, `"corner"`, or `"global-var"`
t_name	Name of the element for which you are deleting a note.

Value Returned

`t`	Returns `t`, when successful.
`nil`	Unsuccessful operation.

Example

The following example code shows how to delete a note added to a corner:

```
ocnSetXLMode()
ocnxlTargetCellView("opamp090" "full_diff_opamp" "adexl_1")
ocnxlCorner( "C0" '(
              ("model" "gpdk090.scs" ?section "\"SF\" \"FS\"") ("modelGroup" "")
              )
           )
ocnxlPutNote( "corner" "C0" "note_content")
ocnxlDeleteNote( "corner" "C0" )
```

ocnxlDisableSweepParam

```
ocnxlDisableSweepParam(
     t_paramName
     )
     => t / nil
```

Description

Lets you disable a sweep parameter. A disabled parameter is not run when `ocnxlRun()` command is fired. See help on `ocnxlRun()`.

Arguments

t_paramName	Name of the parameter.

Value Returned

`t`	Returns `t` if the sweep parameter is disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlDisableSweepParam("solutions/ampTest/schematic/R1/r")
=> t
```

ocnxlDisableSweepVar

```
ocnxlDisableSweepVar(
     t_varName
     )
     => t / nil
```

Description

Lets you disable a sweep variable. A disabled sweep is not run when `ocnxlRun()` command is fired. See help on `ocnxlRun()`.

Arguments

t_varName	Name of the variable.

Value Returned

`t`	Returns `t` if the sweep variable is disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlDisableSweepVar("CAP")
=> t
```

ocnxlEnableSweepParam

```
ocnxlEnableSweepParam(
     t_paramPath
     )
     => t / nil
```

Description

Enables a sweep parameter. A disabled sweep parameter is not run when the `ocnxlRun()` command is run.

Arguments

t_paramPath	Name of the sweep parameter.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlEnableSweepParam("solutions/ampTest/schematic/R1/r")
=> t
```

ocnxlEnableSweepVar

```
ocnxlEnableSweepVar(
     t_varName
     )
     => t / nil
```

Description

Enables a sweep variable. A disabled sweep variable is not run when the `ocnxlRun()` command is run.

Arguments

t_varName	Name of the sweep variable.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlEnableSweepVar("CAP")
=> t
```

ocnxlEnableTest

```
ocnxlEnableTest(
     t_testName
     )
     => t / nil
```

Description

Enables a test. A disabled test will not be run when the `ocnxlRun()` command is run.

Arguments

t_testName	Name of the test.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlEnableTest("test_two")
=> t
```

ocnxlGetJobId

```
ocnxlGetJobId(
     )
     => x_jobID / nil
```

Description

Returns the ID of the current simulation job. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

None

Value Returned

x_jobId	Returns the job ID of the current simulation job
`nil`	Returns `nil` otherwise

Example

```
id=ocnxlGetJobId()
```

ocnxlGetSession

```
ocnxlGetSession(
     )
     => t_sessionName / nil
```

Description

Returns the session name.

Arguments

None

Value Returned

t_sessionName	Returns the name of the session.
`nil`	Returns `nil` and prints an error message if there is no OCEAN XL session.

Example

The following example shows that this function returns the default session name assigned by OCEAN.

```
ocnSetXLMode()
t
ocnxlTargetCellView("myLib" "ampTest" "adexl")
t
ocnxlGetSession()
"ocnXLSession_Apr_18_10_11_38_2013"
```

ocnxlJobSetup

```
ocnxlJobSetup(
     l_setupOptions
     [ ?name t_name ]
     )
     => t / nil
```

Description

Specifies various job setup details for a simulation job.

Arguments

l_setupOptions	The list of setup options.
?name *t_name*	Specifies a name to be used to save the job setup.

The following table describes all the job setup options for `l_setupOptions`.

`-distributionmethod`	Specifies the location where the job has to run. The possible values for distribution method are: `Local`, `Remote-Host`, `Command`, `LBS`. Default value: `Local` When you choose `LBS` distribution method, by default OCEAN uses the LBS Distributed Resource Management Systems (DRMS). You can choose to use LSF or SGE DRMS by setting the `LBS_CLUSTER_MASTER` and `LBS_BASE_SYSTEM` environment variables in your `.cshrc` file. For more details, refer to Setting Up to Use Distributed Processing Option in *Virtuoso Analog Distributed Processing Options User Guide*.
`-maxjobs`	Specifies the maximum number of jobs that can run at any time during the given session. Default value: `1`
`-starttimeout`	Specifies the time (number of seconds) to wait for the icrp process (a process that runs the specific job) to report back that it has started the job. The wait time starts as the job is submitted. Default value: `300`
`-startmaxjobsimmed`	Immediately submits all the specified maximum number of jobs. Default value: `1`

`-configuretimeout`	Specifies the time (number of seconds) to wait for the icrp process to report back that it has configured the job. The wait time starts as soon as a job configure command is sent. Default value: `300`
`-lingertimeout`	Specifies the time (number of seconds) after which you want the program to kill the icrp process after the simulations finish. Default value: `300`
`-runtimeout`	Specifies the time (number of seconds) to wait for the icrp process to report back that it has run the job. The wait time starts as soon as the run command for the job is sent. Default value: `-1`, which means ADE XL keeps waiting for infinite time for the icrp process to report back.
`-showoutputlogerror`	Displays the output log files of all error points in the test. Default value: `0`
`-showerrorwhenretrying`	Displays the output log file on the occurrence of an error for a test, even if the ADE XL distribution system is retrying the test. Default value: `1`
`-reconfigureimmediately`	When running multiple runs in the same ADE XL session, specifies that a completed job be reassigned from the current run to a new run. Default value: `1`
`-jobqueue`	Specifies the name of the queue. If no queue is defined, the job is placed in the default queue. This option is used only when LBS or LSF DRMS is used.
`-jobhostname`	Specifies the name of the host on which the job will run. If no host is specified, the system assigns the job to any available host. This option is used when `distributedmethod` is set as `Remote-Host` or when `distributedmethod` is set as LBS and either LBS or LSF DRMS is used.
`-parallelnumprocs`	Specifies the number of parallel processors to be used. This option is used only when LSF DRMS is used.
`-jobresourcerequirements`	Specifies a string describing the resources required to run the job when LSF DRMS is used. To know more about the format of the resource requirements string, refer to LSF Resource Requirement String Format in *Virtuoso Analog Distributed Processing Options User Guide*.

`-sgehardresource`	Specifies requirements for hardware resources for the job to be run when SGE DRMS is used.
`-sgesoftresource`	Specifies requirements for software resources for the job to be run when SGE DRMS is used.
`-sgepriority`	Specifies priority for the job being submitted when SGE DRMS is used.
`-sgeparallelenv`	Specifies the name of a parallel environment for the job to be run when SGE DRMS is used.
`-jobsubmitcommand`	Specifies the command you want to use to start jobs. This option is used when `distributionmethod` is set as `Command`.

Value Returned

`t`	Returns `t` if the job setup information is set successfully.
`nil`	Returns `nil` otherwise.

Example 1

The following command sets the job policy for a local job:

```
ocnxlJobSetup( '("configuretimeout" "300" "distributionmethod" "Local"
"lingertimeout" "300" "maxjobs" "1" "preemptivestart" "1" "reconfigureimmediately"
"1" "runtimeout" "-1" "showerrorwhenretrying" "1" "starttimeout" "300" ) )
```

Example 2

The following command sets the job policy for a bsub command for LSF (command):

```
ocnxlJobSetup ( '( "distributionmethod" "command" "bsub -I -q queue1") )
```

Example 3

The following command sets the job policy for LSF (LBS mode):

```
ocnxlJobSetup ( '( "distributionmethod" "LBS" configuretimeout" "300"
"lingertimeout" "300" "maxjobs" "1" "jobqueue" "fast" "jobhostname" "sun15") )
```

ocnxlLoadSetupState

```
ocnxlLoadSetupState(
     t_state
     t_mode
     [ ?tests t_tests ]
     [ ?vars t_vars ]
     [ ?parameters t_parameters ]
     [ ?currentMode t_currentMode ]
     [ ?runOptions t_runOptions ]
     [ ?specs t_specs ]
     [ ?corners t_corners ]
     [ ?modelGroups t_modelGroups ]
     [ ?extensions t_extensions]
     [ ?relxanalysis t_relxanalysis ]
     )
     => t / nil
```

Description

Restores the settings in the specified setup state to the active setup.

Arguments

t_state	The name of the setup state to be restored.
t_mode	Specifies the mode for restoring the settings in the setup state to the active setup. Valid values: `'retain`, `'merge`, `'overwrite`
?test *t_tests*	Specifies whether the tests in the setup state should be restored to the active setup. Valid values: `t`, `nil` Default Value: `t`
?vars *t_vars*	Specifies whether the global variables in the setup state should be restored to the active setup. Valid values: `t`, `nil` Default Value: `t`
?parameters *t_parameters*	Specifies whether the parameters in the setup state should be restored to the active setup. Valid values: `t`, `nil` Default Value: `t`
?currentMode *t_currentMode*	Specifies whether the run mode in the setup state should be restored to the active setup Valid values: `t`, `nil` Default Value: `t`
?runOptions *t_runOptions*	Specifies whether the run options in the setup state should be restored to the active setup Valid values: `t`, `nil` Default Value: `t`
?specs *t_specs*	Specifies whether the parameter specifications in the setup state should be restored to the active setup Valid values: `t`, `nil` Default Value: `t`

`?corners` *`t_corners`*	Specifies whether the corners in the setup state should be restored to the active setup Valid values: `t`, `nil` Default Value: `t`
`?modelGroups` *`t_modelGroups`*	Specifies whether the model groups in the setup state should be restored to the active setup. Valid values: `t`, `nil` Default Value: `t`
`?extensions` *`t_extensions`*	Specifies whether the extensions in the setup state should be restored to the active setup. Valid values: `t`, `nil` Default Value: `t`
`?relxanalysis` *`t_relxanalysis`*	Specifies whether the details of reliability analysis should be restored to the active setup. Valid values: `t`, `nil` Default Value: `t`

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlLoadSetupState("optimize")
```

Loads the setup state named `optimize`.

ocnxlMaxJobFail

```
ocnxlMaxJobFail(
     n_int
     )
```

Description

Specifies the maximum number of times the application should restart an ICRP job if it fails to start. This variable is used in batch mode to ensure that the job retries do not go into an infinite loop.

Arguments

n_int	An integer ranging between 1 and 100000. Default value: `20`

Value Returned

None

Example

```
ocnxlMaxJobFail(20)
```

ocnxlMainSimSession

```
ocnxlMainSimSession(
     )
     => g_session / nil
```

Description

Returns the session object for the main simulation session. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

None

Value Returned

g_session	Returns the session object for the main simulation session
`nil`	Returns `nil` otherwise

Example

```
ocnxlMainSimSession()
```

ocnxlProjectDir

```
ocnxlProjectDir(
     t_projectDir
     )
     => t / nil
```

Description

Sets the project directory to the specified location. All simulation data goes into this location by default, if the simulation results or results directories are not set. By default, the project directory is set as `$HOME/simulation`.

Arguments

`t_projectDir`	Sets the location of the project directory.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlProjectDir("/tmp/simulation")
```

Related Commands

ocnxlResultsLocation, ocnxlSetupLocation

ocnxlPutNote

```
ocnxlPutNote(
     t_item
     t_name
     t_note
     )
     => t / nil
```

Description

Adds a note to the given test, history, corner, or variable.

For more information about notes, see Adding Notes to a Test.

Arguments

t_item	Type of the element to which you need to add a note. Valid values: `"test"`, `"history"`, `"corner"`, or `"globalvar"`
t_name	Name of the element to which you are adding a note.
t_note	Text to be added to the note. A note can contain a maximum of 512 characters.

Value Returned

`t`	Returns `t`, when successful.
`nil`	Unsuccessful operation.

Example

The following example code shows how to add a note for the gain global variable:

```
ocnSetXLMode()
ocnxlTargetCellView("opamp090" "full_diff_opamp" "adexl_1")
ocnxlSweepVar("gain" "10")
ocnxlPutNote( "globalvar" "gain" "note_content")
```

ocnxlResultsLocation

```
ocnxlResultsLocation(
     t_resultsDir
     )
     => t / nil
```

Description

Sets the results directory to the specified location. All results database and log files are saved in the /`libraryName/cellName/<target-view>/results/data/` directory at this location.

By default, data is saved at the `<target-view>/results/data` directory. See help on ocnxlTargetCellView.

Note: If you do not specify the results directory and you open the ADE XL view in read-only mode or do not have write permissions in the ADE XL view, the program writes results databaseinformation and run log files to `libraryName/cellName/adexl/results/data/<history_item>` in the project directory, set by ocnxlProjectDir.

Arguments

t_resultsDir	Location of the results directory.

Value Returned

`t`	Returns `t` if the results location is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlResultsLocation("/home/ocnuser")
```

ocnxlSaveSetupAs

```
ocnxlSaveSetupAs(
     t_lib
     t_cell
     t_view
     )
     => t / nil
```

Description

Saves the current setup to a different adexl view.

Arguments

t_lib	The name of the library in which the new adexl view is to be saved.
t_cell	The name of the cell in which the new adexl view is to be saved.
t_view	The name of the new adexl view.

Value Returned

`t`	Returns `t` if the save is successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSaveSetupAs("solution" "ampTest" "newView")
```

ocnxlSensitivityVars

```
ocnxlSensitivityVars(
     l_varValueList
     )
     => t / nil
```

Description

Specifies a list of sensitivity variables for the given setup.

Arguments

l_varValueList	List of variable and values combination. Each list item contains the name of sensitivity variable, a list of sweep values for that variable, and a reference value.

Value Returned

`t`	Returns `t` if the list of sensitivity variables is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSensitivityVars(list '("CAP" "100f 200f 300f" "200f") '("RES" "1K 1.5K 2K"
"1.5K"))
=> t
```

ocnxlSetDesignVariablePerTest

```
ocnxlSetDesignVariablePerTest(
     t_varName
     t_testName
     )
     => t / nil
```

Description

Sets a value for a design variable to be overridden for the given test. In this case, the test will not use the global value set for the variable.

Arguments

t_varName	Name of the variable for which value is to be set.
t_testName	Name of the test for which the variable value is to be set.

Value Returned

`t`	If the value is set successfully.
`nil`	Returns nil otherwise

Example

```
ocnxlSetDesignVariablePerTest("CAP" "test1")
=> t
; this implies that for test test1, the CAP variable will use the local value
; instead of the global value.
```

ocnxlSetAllVarsDisabled

```
ocnxlSetAllVarsDisabled(
     g_disabled
     )
     => t / nil
```

Description

Lets you enable or disable all global variables.

Arguments

g_disabled	Specify `t` to disable all variables, and `nil` to enable all variables.

Value Returned

`t`	Returns `t` if all variables are enabled or disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSetAllVarsDisabled(t) => t
```

ocnxlSetupLocation

```
ocnxlSetupLocation(
     t_setupDir
     )
     => t / nil
```

Description

Sets the setup directory to the specified location. All setup data goes into this location. By default data goes into the target cell view. See help on `ocnxlTargetCellView`.

Arguments

t_setupDir	Location of the setup directory.

Value Returned

`t`	Returns `t` if the location of the setup directory is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlTargetCellView( "solutions" "ampTest" "maestro" )
; All the setup data goes into the solutions/ampTest/maestro directory.

ocnxlTargetCellView( "solutions" "ampTest" "maestro" )
ocnxlSetupLocation("/home/ocnuser")
;Creates a new directory structure solutions/ampTest/maestro in the location "/
home/ocnuser". All the setup data, such as data.sdb, documents etc. will be saved
in this location.
```

ocnxlSimResultsLocation

```
ocnxlSimResultsLocation(
     t_simResultsDir
     )
     => t / nil
```

Description

Sets the simulation results directory to the specified location. All simulation data goes into this location. If the simulation results directory is not set using this function, the simulation results are saved at any one of the following locations:

- In the /`libraryName/cellName/<target-view>/results/data/<history_item>` directory at the location set by ocnxlResultsLocation, if set.
- Otherwise, in the `libraryName/cellName/<target-view>/results/data/<history_item>` directory at the location set by ocnxlProjectDir, if set.
- Otherwise, in the `$HOME/simulation/libraryName/cellName/<target-view>/results/data` directory.

Arguments

`t_simResultsDir`	Sets the location of the simulation results directory.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSimResultsLocation("/home/ocnuser")
```

ocnxlStimuliData

```
ocnxlStimuliData(
     [ ?inputs t_inputs ]
     [ ?globals t_globals ]
     )
     => t / nil
```

Description

Sets the specified analog stimuli (input stimulus and global sources) for a test.

Arguments

`?inputs` *`t_inputs`*	Specifies the input stimuli.
`?globals` *`t_globals`*	Specifies the global sources.

Value Returned

`t`	Returns `t` if the analog stimuli is set successfully.
`nil`	Returns `nil` otherwise.

Example

The following command sets the specified input stimuli:

```
ocnxlStimuliData(
?inputs '(((name "iopin") (nodes ("iopin" "/gnd!")) (enabled t) (tranType "dc")
(srcType "Voltage") (instParameters (dc("3n") type("dc"))) (otherParameters
((FNpairs "0")))))
)
```

ocnxlSweepParam

```
ocnxlSweepParam(
     t_paramName
     t_paramValue
     )
     => t / nil
```

Description

Lets you define a sweep parameter along with its value.

Arguments

t_paramName	Name of the parameter.
t_paramValue	Value of the parameter.

Value Returned

`t`	Returns `t` if sweep for the parameter is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSweepParam("solutions/ampTest/schematic/R1/r" "10K") =>t
```

ocnxlSweepVar

```
ocnxlSweepVar(
     t_varName
     t_varValue
     )
     => t / nil
```

Description

Lets you define a sweep variable along with its value.

Arguments

t_varName	Name of the variable.
t_varValue	Value of the variable and the specification for the sweep.

Value Returned

`t`	Returns `t` if the sweep is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSweepVar("CAP" "5p")
=>t
```

ocnxlTargetCellView

```
ocnxlTargetCellView(
     t_lib
     t_cell
     t_view
     [ ?mode t_mode ]
     )
     => t / nil
```

Description

Specifies target cellview where data will be created.

Arguments

t_lib	Name of the library.
t_cell	Name of the cell.
t_view	Name of the view.
?mode *t_mode*	Specifies the mode of the target cellview. Valid values: ■ `r` - Opens the target cellview in read mode ■ `a` - Opens the target cellview in edit mode. Default value: `a`

Value Returned

`t`	Returns `t` if it is able to use the given library, cell, and view as the target.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlTargetCellView("opamplib" "ampTest" "adexl" ?mode "r")
=> t
; ...
;; to access the library, cell, and view name set by this function later in the
script, you can use the following code:
```

```
session = ocnxlGetSession()
=> "ocnXLSession_Nov__9_11_27_10_2015"
axlGetSessionLibName(session)
=> "opamplib"
axlGetSessionCellName(session)
=> "ampTest"
axlGetSessionViewName(session)
=> "adexl"
```

Commands for Corners

- ocnxlCorner
- ocnxlDisableCorner
- ocnxlDisableCornerForTest
- ocnxlEnableCorner
- ocnxlEnableCornerForTest
- ocnxlGetCorners
- ocnxlModelGroup

ocnxlCorner

```
ocnxlCorner(
     t_cornerName
     l_cornerDetails
     )
     => t / nil
```

Description

Lets you define a corner.`cornerDetails` is a list of elements where each element is (*`t_type t_varName t_value`*). Available types are `variable`, `parameter`, and `model`.

Arguments

`t_cornerName`	Name of the corner.
`l_cornerDetails`	Details of the corner. Details is a list of items where each item has a tag, name, and a value. The tag can be of 3 types: ■ `variable` ■ `parameter` ■ `model`

Value Returned

`t`	Returns `t` if the corner is defined.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlCorner("C0" '( ("variable" "CAP" "2p") ("variable" "T" "78"))) => t
```

ocnxlDisableCorner

```
ocnxlDisableCorner(
     t_cornerName
     )
     => t / nil
```

Description

Lets you disable a corner. A disabled corner will not be run when the ocnxlRun() command is run. This command works only in XL mode. See help on `ocnSetXLMode()`

Arguments

t_cornerName	The name of the corner to be disabled.

Value Returned

`t`	Returns `t` if the corner is disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlDisableCorner("C0")  => t
```

ocnxlDisableCornerForTest

```
ocnxlDisableCornerForTest(
     t_cornerName
     t_testName
     )
     => t / nil
```

Description

Lets you disable a corner for a test.

Arguments

t_cornerName	Name of the corner.
t_testName	Name of the test.

Value Returned

`t`	Returns `t` if the corner of the test is disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlDisableCornerForTest("C0" "test_one")
```

ocnxlEnableCorner

```
ocnxlEnableCorner(
     t_cornerName
     )
     => t / nil
```

Description

Lets you enable a corner. An enabled corner will be run when ocnxlRun() command is run.

Arguments

t_cornerName	The name of the corner to be enabled.

Value Returned

`t`	Returns `t` if the corner is disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlEnableCorner("C0") => t
```

ocnxlEnableCornerForTest

```
ocnxlEnableCornerForTest(
     t_cornerName
     t_testName
     )
     => t / nil
```

Description

Enables a corner for a test.

Arguments

t_cornerName	Name of the corner.
t_testName	Name of the test.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlEnableCornerForTest("C0" "test_one") => t
```

ocnxlGetCorners

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

Description

Returns a list of corners names.

Arguments

None

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGetCorners()
```

ocnxlModelGroup

```
ocnxlModelGroup(
     t_modelGroupName
     l_modelFileSetup)
     => t / nil
```

Description

Lets you add and define a new model group.

Arguments

t_modelGroupName	Name of the new model group.
l_modelFileSetup	List of model file spec. Model file spec: ■ *t_modelFilePath* ■ *[*?section *t_section]* ■ *[*?enabled *g_enabled]* ■ *[*?test *t_test]* ■ *[*?block *t_block]*

Value Returned

`t`	Returns `t` if a new model is defined.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlModelGroup( "F2"
'(
( "/myModels/Models/model1.scs" ?enabled nil  ?section "")
( "/myModels/Models/model2.scs" ?section "")
( "/myModels/Models/model3.scs" ?enabled nil  ?section "")
))
```

Commands for Outputs

- ocnxlAddOrUpdateOutput
- ocnxlExportOutputView
- ocnxlOutputAreaGoal
- ocnxlOutputExpr
- ocnxlOutputMatExpr
- ocnxlOutputMatlabScript
- ocnxlOutputOceanScript
- ocnxlOutputOpRegion
- ocnxlOutputSignal
- ocnxlOutputSpiceScript
- ocnxlOutputTerminal
- ocnxlOutputViolations
- ocnxlPutChecksAsserts
- ocnxlPutChecksAssertsTest
- ocnxlPutEnabledChecksAssertsCellViews

ocnxlAddOrUpdateOutput

```
ocnxlAddOrUpdateOutput(
     t_outputName
     t_outputVal
     )
     => t / nil
```

Description

Adds the specified scalar output to the simulation setup so that the results for the output can be viewed on the Results tab in ADE XL. If the specified output name already exists, only its value is updated with the specified value. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

t_outputName	Output name
t_outputVal	Output value

Value Returned

`t`	Returns `t` if output is successfully added or updated
`nil`	Returns `nil` otherwise

Example

```
ocnxlAddOrUpdateOutput("Calibrated_ParamName" CalResult)
```

ocnxlExportOutputView

```
ocnxlExportOutputView(
     t_fileName
     t_viewType
     )
     => t / t_error
```

Description

Exports the results view to the specified .csv or .html file.

Argument

t_fileName	Path and name of file to which the results need to be exported. Append .htm or .html to the filename to write in HTML format and .csv to write in CSV format. If no file extension is specified, a .csv file is created by default. The file is saved in the current working directory.

t_viewType	Name of the view type that needs to be exported.
	
	Possible values:
	
	`""` or `Current`: writes the currently visible view
	
	`Detail`
	
	`Detail-Transpose`
	
	`Optimization`
	
	`Summary`
	
	`Yield`
	
	Default value: `""`

Value Returned

`t`	Successful export of output view.
t_error	If unsuccessful, returns an error message.

Examples

```
ocnxlExportOutputView( "./abc.csv" "Yield")

ocnxlExportOutputView( "./abc.html" "Detail-Transpose")

ocnxlExportOutputView("./abcd.html" "Yield" )
```

ocnxlOutputAreaGoal

```
ocnxlOutputAreaGoal(
     t_expr
     [ ?name t_outputName ]
     [ ?plot plot ]
     [ ?save save ]
     )
     => t / nil
```

Description

Adds an area goal output expression in the current test being specified. A test's specification begins with `ocnxlBeginTest()`. See help on `ocnxlBeginTest()`.

Arguments

t_expr	The expression that you want to add.
?name *t_outputName*	The name of the expression.
?plot *t_plot*	Whether to plot or not.
?save *t_save*	Whether to save or not.

Value Returned

t	Returns t if the output expression is set.
nil	Returns nil otherwise.

Example

```
ocnxlOutputAreaGoal( "(('I8/R4' ('res' 'w*l' 'default' 'enabled'))...)" ?name
"MAX" ?plot t ?save t ) => t
```

Adds an area goal output expression named `MAX` for the current test.

ocnxlOutputExpr

```
ocnxlOutputExpr
     ( t_expr
     [ ?name t_outputName ]
     [ ?plot plot ]
     [ ?save save ]
     [ ?evalType t_evaltype]
     )
     => t / nil
```

Description

This command adds an output expression in the current test being specified. Specification of a test specification begins with `ocnxlBeginTest()`. See help on `ocnxlBeginTest()`.

Arguments

t_expr	The expression that you want to add.
?name *t_outputName*	The name of the expression.
?plot *plot*	Whether to plot or not.
?save *save*	Whether to save or not.
?evaltype *t_evaltype*	Evaluation type of the expression Possible values: ■ "point": Calculates the expression for every design point ■ "corners": Calculates the expression across all the corners ■ "sweeps": Calculates the expression across all the sweep points ■ "maa": Calculates the expression across all the corners and sweep points Default value: "point" **Note:** "sweeps" and "maa" are applicable only for the maestro cellviews created using ADE Assembler.

Value Returned

t	Returns t if the output expression is set.
nil	Returns nil otherwise.

Example

```
ocnxlOutputExpr( "ymax(VT("/out"))" ?name "MAX" ?plot t ?save t )
# Adds "/out" in the outputs.
```

ocnxlOutputMatExpr

```
ocnxlOutputMatExpr
     ( t_expr
     [ ?name t_name ]
     [ ?plot b_flag ]
     [ ?save b_flag ]
     [ ?evalType s_type]
     )
     => t / nil
```

Description

Creates a MATLAB expression output in the test where it is being specified.

Note: This functions is applicable only for ADE Explorer and ADE Assembler.

Arguments

t_expr	The MATLAB expression that you want to add to the output. **Note:** If you need to use a Calculator function in the MATLAB expression, specify it using the matEval function.
`?name` *t_name*	The name of the MATLAB expression.
`?plot` *b_flag*	Specifies whether to plot or not.
`?save` *b_flag*	Specifies whether to save or not.
`?evaltype` *s_type*	Evaluation type of the expression Possible values: ■ `"point"`: Specifies that the MATLAB expression is to be evaluated for every design point ■ `"maa"`: Specifies that the MATLAB expression is to be evaluated across all design points Default value: `"point"`

Value Returned

`t`	Returns `t` if the MATLAB expression is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlOutputMatExpr("matEval(\"fft(VF(\\\"/out\\\"))\" ?name "out" ?plot t
?evalType 'maa)
;Creates a MATLAB expression output for evaluation during the simulation. The
simulation will dispatch this output expression to matlab for evaluation.
```

ocnxlOutputMatlabScript

```
ocnxlOutputMatlabScript(
     t_script
     [ ?name t_outputName ]
     [ ?plot plot ]
     [ ?save save ]
     )
     => t / nil
```

Description

Adds a MATLAB script based output in the current test being specified. A test's specification begins with `ocnxlBeginTest()`. See help on `ocnxlBeginTest()`.

Arguments

t_script	Name and location of the script file.
t_outputName	Name of the output file.
plot	Specifies if the values are to be plotted.
save	Specifies if the output are to be saved.

Value Returned

`t`	Returns `t` if the output is generated.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlOutputMatlabScript( "/tmp/my_measure.m" ?name "MAX" ?plot t ?save t )
=> t
Adds "MAX" in the outputs.
```

ocnxlOutputOceanScript

```
ocnxlOutputOceanScript(
     t_script
     [ ?name t_outputName ]
     [ ?plot plot ]
     [ ?save save ]
     [ ?evalType t_evaltype ]
     )
     => t / nil
```

Description

Adds an OCEAN script based output in the current test being specified. A test's specification begins with `ocnxlBeginTest()`. See help on `ocnxlBeginTest()`.

Arguments

t_script	Name and location of the script file.
?name *t_outputName*	Name of the output file.
?plot *plot*	Specifies if the values are to be plotted.
?save *save*	Specifies if the output are to be saved.
?evalType *t_evaltype*	Whether to evaluate the OCEAN script for a design point or across all corners for a design points. Valid Values: `corners`, `point` Default Value: `point`

Value Returned

`t`	Returns `t` if the output is generated.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlOutputOceanScript( "/tmp/my_measure.ocn" ?name "MAX" ?plot t ?save t )
=> t
Adds "MAX" in the outputs.
```

ocnxlOutputOpRegion

```
ocnxlOutputOpRegion(
     t_expr
     [ ?name name ]
     [ ?plot plot ]
     [ ?save save ]
     )
     => t / nil
```

Description

Adds an operating region specification in the current test being specified. There can be only one operating region for a test. A test specification begins with ocnxlBeginTest.

Arguments

t_expr	Operating region expression
?name *t_name*	Name of the operating region output
?plot *t_plot*	Specifies if the results for the output are to be plotted Specify: ■ `t` to plot the results ■ `nil` to disable plotting of the results
?save *t_save*	Specifies if the results for the output are to be saved Specify: ■ `t` to save the results ■ `nil` to disable saving of the results

Return Values

t	Returns `t` if the output is added
nil	Returns `nil` otherwise

Example

```
ocnxlOutputOpRegion( "((\"enabled\" \"/amp/M1\" \"\" \"Schematic\" \"vgs-vth\"
\">\" \"25m\" \"amp.M1\"))" ?name "Op_Region" ?plot t ?save t)
```

ocnxlOutputSignal

```
ocnxlOutputSignal(
     t_signal
     [ ?plot plot ]
     [ ?save save ]
     )
     => t / nil
```

Description

This command adds an output signal in the current test being specified. A test's specification begins with `ocnxlBeginTest()`. See help on `ocnxlBeginTest()`.

Arguments

t_signal	The name of the signal.
?plot *plot*	Whether to plot or not.
?save *save*	Whether to save or not.

Value Returned

t	Returns t if the output signal is set.
nil	Returns nil otherwise.

Example

```
ocnxlOutputSignal( "/out" ?plot t ?save t )

# Adds "/out" in the outputs.
```

ocnxlOutputSpiceScript

```
ocnxlOutputSpiceScript(
     t_script
     [ ?name t_name ]
     [ ?plot g_plot ]
     [ ?save g_save ]
     )
     => t / nil
```

Description

Adds the spice .measure script as an output

Arguments

t_script	Name of the variable for which value is to be set.
?name *t_Name*	Name of the test for which the variable value is to be set. Default: `nil`.
?plot g_plot	Specifies if the output is to be plotted. Default: `nil`.
?save *g_save*	Specifies if the output is to be saved. Default: `nil`

.Value Returned

`t`	Returns `t` if the output is added successfully.
`nil`	When unsuccessful.

Example

```
ocnxlOutputSpiceScript("./spice.meas")
```

ocnxlOutputTerminal

```
ocnxlOutputTerminal(
     t_term
     [ ?plot plot ]
     [ ?save save ]
     [ ?type type ]
     )
     => t / nil
```

Description

This command adds an output terminal in the current test being specified. Specifications for a test begin with `ocnxlBeginTest()`. For more information, see `ocnxlBeginTest()`.

Arguments

t_term	The name of the terminal.
?plot *plot*	Specifies whether to plot or not.
?save *save*	Specifies whether to save or not.
?type *type*	If `type` is set to 'terminalV', a terminal voltage output is created instead of a normal terminal current output. If `type` is not specified, a current terminal is saved.

Value Returned

`t`	Returns `t` if the output terminal is set.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlOutputTerminal( "/I0/M18/S" ?plot t ?save t)
# Adds "/I0/M18/S" in the current outputs.

ocnxlOutputTerminal( "/I0/M18/G" ?plot t ?save t ?type "terminalV")
# Adds "/I0/M18/G" in the voltage terminal outputs.
```

ocnxlPutChecksAsserts

```
ocnxlPutChecksAsserts(
     [ ?netlist g_netlist ]
     )
     => t / nil
```

Description

Adds the checks and asserts to the active setup for an overall control on the netlisting of checks and asserts defined in the Checks/Asserts assistant. Technically, this command adds the `checksasserts` node in the active setup.

Arguments

`?netlist` *`g_netlist`*	Specifies whether the checks and asserts defined in the Checks/Asserts assistant will be netlisted or not. This provides an overall control over the netlisting of checks and asserts. Valid values: ■ `t`: Enables overall netlisting of checks and asserts. However, netlisting of the checks and asserts defined for the cellviews under a particular test is controlled by the ocnxlPutChecksAssertsTest command. ■ `nil`: Checks and asserts are not netlisted Default value: `nil`

Value Returned

`t`	Returns `t` f the checks and asserts are successfully added in the active setup
`nil`	Returns `nil` if the checks and asserts already exist in the active setup.

Example

The following example controls netlisting of checks and asserts for two tests. Netlisting of checks and asserts is disabled for the first test. All the checks and asserts are netlisted for the second test.

ocnxlPutChecksAsserts(?netlist t)

```
ocnxlPutChecksAssertsTest(?testName "vip:test:ca_disabled" ?netlist nil
?netlistScope "all")
```

```
ocnxlPutChecksAssertsTest(?testName "vip:test:ca_enabled:all" ?netlist t
?netlistScope "all")
```

ocnxlOutputViolations

```
ocnxlOutputViolations(
     t_expr
     [ ?name t_name ]
     [ ?plot g_plot ]
     [ ?save g_save ]
     )
     => t / nil
```

Description

Adds a new output of type `violations` for the current test to show violation filters for checks or asserts.

Arguments

t_expr

Expression to be used for the output. The format for the expression contains the following parts:

- Name of the operation or aggregate function. For example, `Avg`, `Max`, `Min`, or `Sum`.
- Name of the field to be accessed in the function
- Name of the violation filter on which you need to run the aggregate function

Example:

```
"Max 'Duration (s)' on 'Dynamic Excessive Rise, Fall,
Undefined State Time Check'"
```

where, `Max` is the aggregate function name, `Duration (s)` is the field name, and `Dynamic Excessive Rise, Fall, Undefined State Time Check` is the violation filter name.

Note: For the `Count` aggregate function, the format for the expression is:

```
"Count '<field-name>'"
```

Example:

```
"Count 'All Checks/Asserts'"
```

?name *t_name*

Name to be assigned to the output.

?plot *g_plot*

Specifies if the value of this output is to be plotted.

`?save` *`g_save`*	Specifies if the value of this output is to be saved.

Value Returned

`t`	Returns `t` if the output is added successfully.
`nil`	Returns `nil` otherwise.

Example

The following example code adds two new outputs `max_duration` and `allChecks` for the current test:

```
ocnxlOutputViolations( "Max 'Duration (s)' on 'Dynamic Excessive Rise, Fall,
Undefined State Time Check'" ?name "max_duration" ?plot t)
```

```
ocnxlOutputViolations( "Count 'All Checks/Asserts'" ?name "Count 'All Checks/
Asserts'" ?plot t)
```

ocnxlPutChecksAssertsTest

```
ocnxPutChecksAssertsTest(
     [ ?testName t_testName ]
     [ ?netlist g_netlist ]
     [ ?netlistScope g_netlistScope ]

     )
     => t / nil
```

Description

Specifies the name of the test to which the checks and asserts defined in the Checks/Asserts assistant will be applicable and whether to netlist the checks and asserts for that test. In addition, it also specifies the scope for netlisting. By default, the checks and asserts are not applied to any test.

Arguments

`?testName` *`t_testName`*	Name of the test
`?netlist` *`g_netlist`*	Controls whether checks and asserts are to be netlisted. Valid values: `t`, `nil` Default value: `nil`
`?netlistScope` *`g_netlistScope`*	Specifies the scope for netlist. Valid values: ■ `"all"`: Checks and asserts of all the cellviews beneath the top level schematic/config view will be netlisted. ■ `"partial"`: Checks and asserts of selected cellviews will be netlisted. Use ocnxlPutEnabledChecksAssertsCellViews to specify the cellviews. ■ `"none"`: Nothing beneath the test bench top level schematic/config view will be netlisted. **Note:** `?netlistScope` is ignored if `?netlist` is `nil`. In that case, netlisting of checks and asserts is disabled so that `?netlistScope` does not take any effect. Default value: `"none"`

Value Returned

`t`	Returns `t` if the test and its netlisting setting for checks and asserts are successfully applied
`nil`	Returns `nil` in the following cases: ■ The `checksasserts` node does not exist in the active setup ■ The test is already added under the `checksasserts` node in the active setup ■ *`t_testName`* is not a valid string ■ *`t_netlistScope`* is not a valid string

Example

The following example controls netlisting of checks and asserts for two tests. Netlisting of checks and asserts is disabled for the first test. All the checks and asserts are netlisted for the second test.

```
ocnxlPutChecksAsserts(?netlist t)
=> t
ocnxlPutChecksAssertsTest(?testName "vip:test:ca_disabled" ?netlist nil
?netlistScope "all")
=> t
ocnxlPutChecksAssertsTest(?testName "vip:test:ca_enabled:all" ?netlist t
?netlistScope "all")
=> t
```

ocnxlPutEnabledChecksAssertsCellViews

```
ocnxlPutEnabledChecksAssertsCellViews(
     [ ?testName t_testName ]
     [ ?enabledCellViews l_enabledCellViews ]
     )
     => t / nil
```

Description

Specifies a list of library/cell/view lists that define the names of cellviews within a test for which the checks and asserts defined in the Checks/Asserts assistant will be netlisted. The checks and asserts under each of these cellviews will be netlisted provided that the netlist attribute of the `test` is set to `t` and its `netlistScope` attribute is set to `partial` or `all`. The cellviews will not be netlisted if the `netlist` attribute is `nil` or the `netlistScope` attribute is `none`.

Arguments

`?testName` *`t_testName`*	Name of the test for which the cellview list is to be applied.
`?enabledCellViews` *`l_enabledCellViews`*	List containing list of library, cell, and view names for which the checks and asserts defined in the Checks/Asserts assistant will be netlisted. The list must be given in the following format: `list(list("lib1 "cell1" "view1") list("lib2 "cell2" "view2") ...)`

Value Returned

`t`	Returns `t` if the list of cellview names is successfully added under the given testname within the `checksasserts` node.
`nil`	Returns `nil` in the following cases: ■ The test specified using *`t_testName`* does not exist in the setup ■ *`t_testName`* is not a valid string ■ Format of the *`l_enabledCellviews`* list is incorrect ■ *`l_enabledCellviews`* does not contain any list item

Example

The following example enables netlisting of checks and asserts for two cellviews in the test `vip:test:ca_enabled:partial`:

```
ocnxlPutChecksAsserts(?netlist t)
=> t
ocnxlPutChecksAssertsTest(?testName "vip:test:ca_enabled:partial" ?netlist t
?netlistScope "partial")
ocnxlPutEnabledChecksAssertsCellViews(?testName "vip:test:ca_enabled:partial"
?enabledCellViews list(list("vip" "test" "schematic") list("checks" "inv"
"schematic")))
```

Commands for Specifications

- ocnxlGetSpecs
- ocnxlPutGreaterthanSpec
- ocnxlPutInfoSpec
- ocnxlPutLessthanSpec
- ocnxlPutMaxSpec
- ocnxlPutMinSpec
- ocnxlPutToleranceSpec
- ocnxlPutRangeSpec
- ocnxlRemoveSpec

ocnxlGetSpecs

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

Description

Returns a list of parameter specification names.

Arguments

None

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGetSpecs()
```

ocnxlPutGreaterthanSpec

```
ocnxlPutGreaterthanSpec(
     t_testName
     t_outputName
     t_thresholdValue
     [ ?weight g_weight ]
     [ ?corner g_cornerName ]
     )
     => t / nil
```

Description

Lets you specify that value of an output must be greater than the given threshold value.

Arguments

t_testName	Name of the test.
t_outputName	Name of the output.
	If you do not specify any name for an output, the expression itself is used as the name of that output. Therefore, in case of unnamed outputs, you can specify the expression as a value for this argument.
t_thresholdValue	The threshold value.
?weight *g_weight*	The weighting factor for the spec.
?corner *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

t	Returns t if the specifications are specified.
nil	Returns nil otherwise.

Example

```
ocnxlPutGreaterthanSpec("test_one" "VT("/out")" "3.5" 4)
=> t
; Spec is defined that transient voltage for /out signal
; must always be greater than 3.5 volts. The weighting factor for the spec is 4.
```

ocnxlPutInfoSpec

```
ocnxlPutInfoSpec(
     t_testName
     t_outputName
     [ ?corner g_cornerName ]
     )
     => t / nil
```

Description

Specifies an info spec for an output.

Arguments

t_testName	Name of the test.
t_outputName	Name of the output. If you do not specify any name for an output, the expression itself is used as the name of that output. Therefore, in case of unnamed outputs, you can specify the expression as a value for this argument.
?corner *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

t	Returns t if the specifications are specified.
nil	Returns nil otherwise.

Example

```
ocnxlPutInfoSpec("test_one" "VT(\"/out\")")
=> t

# An info spec “test_one” is set for the expression “VT(\"/out\”.
```

ocnxlPutLessthanSpec

```
ocnxlPutLessthanSpec(
     t_testName
     t_outputName
     t_thresholdValue
     [ ?weight g_weight ]
     [ ?corner g_cornerName ]
     )
     => t / nil
```

Description

Lets you specify that value for a output must be less than the given threshold value.

Arguments

t_testName	Name of the test.
t_outputName	Name of the output.
	If you do not specify any name for an output, the expression itself is used as the name of that output. Therefore, in case of unnamed outputs, you can specify the expression as a value for this argument.
t_thresholdValue	The threshold value.
?weight *g_weight*	The weighting factor for the spec.
?corner *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

t	Returns t if the specifications are specified.
nil	Returns nil otherwise.

Example

```
ocnxlPutLessthanSpec("test_one" "VT("/out")" "6.5" 4)
=> t
; Spec is defined that the transient voltage for /out signal
; must always be less than 6.5 volts. The weighting factor for the spec is 4.
```

ocnxlPutMaxSpec

```
ocnxlPutMaxSpec(
     t_testName
     t_resultName
     t_maxValue
     [ ?weight g_weight ]
     [ ?corner g_cornerName ]
     )
     => t / nil
```

Description

Lets you specify a maximize spec for a result.

Arguments

t_testName	Name of the test.
t_resultName	Name of the result. If you do not specify any name for a result, the expression itself is used as the name of that result. Therefore, in case of unnamed results or outputs, you can specify the expression as a value for this argument.
t_maxValue	The maximum target value.
?weight *g_weight*	The weighting factor for the spec.
?corner *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

t	Returns t if the specifications are specified.
nil	Returns nil otherwise.

Example

```
ocnxlPutMaxSpec("test_one" "VT("/out")" "6.5" ?weight 4)
=> t
; Spec is defined that transient voltage for /out signal should be as high as
; possible and must be more than 6.5 volts to pass. The weighting factor for the
; spec is 4.
```

ocnxlPutMinSpec

```
ocnxlPutMinSpec(
     t_testName
     t_outputName
     t_minValue
     [ ?weight g_weight ]
     [ ?corner g_cornerName ]
     )
     => t / nil
```

Description

Lets you specify a minimize spec for an output.

Arguments

t_testName	Name of the test.
t_outputName	Name of the output.
	If you do not specify any name for an output, the expression itself is used as the name of that output. Therefore, in case of unnamed outputs, you can specify the expression as a value for this argument.
t_minValue	The minimum target value.
?weight *g_weight*	The weighting factor for the spec.
?corner *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

t	Returns t if the specifications are specified.
nil	Returns nil otherwise.

Example

```
ocnxlPutMinSpec("test_one" "VT("/out")" "3.5" ?weight 4)
=> t

; Spec is defined that the transient voltage for /out signal should be
; as low as possible and it must be below 3.5 volts to pass. The weighting factor
; for the spec is 4.
```

ocnxlPutToleranceSpec

```
ocnxlPutToleranceSpec(
     t_testName
     t_outputName
     t_value
     s_toleranceType
     t_toleranceValue
     [ ?weight g_weight ]
     [ ?corner g_cornerName ]
     )
     => t / nil
```

Description

Lets you specify a tolerance spec for an output.

Arguments

t_testName	Name of the test.
t_outputName	Name of the output.
	If you do not specify any name for an output, the expression itself is used as the name of that output. Therefore, in case of unnamed outputs, you can specify the expression as a value for this argument.
t_value	The target value.
s_toleranceType	The type of tolerance.
t_toleranceValue	The tolerance value.
`?weight` *g_weight*	The weighting factor for the spec.
`?corner` *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

`t`	Returns `t` if the specifications are specified.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlPutToleranceSpec("test_one" "VT("/out")" "5.0" 'percentage "10" ?weight 4)
=> t

# Spec is defined that transient voltage for /out signal must be 5.0 volts with
tolerance 10%. The weighting factor for the spec is 4.
```

ocnxlPutRangeSpec

```
ocnxlPutRangeSpec(
     t_testName
     t_outputName
     t_maxValue
     t_minValue
     [ ?weight g_weight ]
     [ ?corner g_cornerName ]
     => t / nil
```

Description

Lets you specify a range spec for an output.

Arguments

t_testName	Name of the test.
t_outputName	Name of the output.
	If you do not specify any name for an output, the expression itself is used as the name of that output. Therefore, in case of unnamed outputs, you can specify the expression as a value for this argument.
t_maxValue	The maximum value of the range.
t_minValue	The minimum value of the range.
?weight *g_weight*	The weighting factor for the spec.
?corner *g_cornerName*	Name of the corner for which the spec is to be used. This argument helps in overridding a spec for the given corner.

Value Returned

t	Returns t if the specifications are specified.
nil	Returns nil otherwise.

Example

```
ocnxlPutRangeSpec("test_one" "VT("/out")" "6.5" "3.5" 4)
=> t
; Spec is defined that maximum transient voltage for /out signal
```

```
; must be 6.5 volts and minimum must be 3.5 volts. The weighting factor for the
; spec is 4.
```

ocnxlRemoveSpec

```
ocnxlRemoveSpec(
     t_specName
     )
     => t / nil
```

Description

Removes the specified parameter specification.

Arguments

`t_specName`	Name of the spec.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlRemoveSpec("MAX")
```

Commands to Set Up Run Options

- ocnxlConjugateGradientOptions
- ocnxlCornerVars
- ocnxlFeasibilityAnalysisOptions
- ocnxlGetBestPointParams
- ocnxlGlobalOptimizationOptions
- ocnxlLocalOptimizationOptions
- ocnxlMonteCarloOptions
- ocnxlSamplingOptions
- ocnxlSensitivityOptions
- ocnxlSizeOverCornersOptions
- ocnxlStartingPoint
- ocnxlSweepsAndCornersOptions
- ocnxlWorstCaseCornersOptions
- ocnxlYieldEstimationOptions
- ocnxlYieldImprovementOptions

ocnxlConjugateGradientOptions

```
ocnxlConjugateGradientOptions (
     [ ?runFullEvaluation t_runFullEvaluation ]
     [ ?meetAllGoals t_meetAllGoals ]
     [ ?timeLimit t_timeLimit ]
     [ ?numPoints t_numPoints ]
     )
     => t / nil
```

Description

Sets options for conjugate gradient runs. This command works only in XL mode. See help on `ocnSetXLMode()`.

Arguments

t_runFullEvaluation	Sets to run full evaluation. Possible Values: “1” and “0”. Default Value: “1”
t_meetAllGoals	Sets to run only until all goals are met. Possible Values: “1” and “0”. Default Value: “1”
t_timeLimit	Sets a time limit (in seconds) for the run.
t_numPoints	Sets the limit in the number of points to be run.

Value Returned

`t`	Returns `t` if options are specified for conjugate gradient run.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlConjugateGradientOptions(?numPoints "3000")
Sets to run for 3000 points.
```

ocnxlCornerVars

```
ocnxlCornerVars(
     l_varValueList
     )
     => t / nil
```

Description

Specifies a list of corner variables (along with their minimum, maximum, and reference values) to be used to run the Create Worst Care Corner simulation run mode.

Arguments

l_varValueList	List of corner variable-value combination list. Each list item contains a list of variable name, minimum and maximum values for the variable, and a reference value.

Value Returned

`t`	Returns `t` if the corner variables for Create Worst Care Corner simulation run mode are set successfully.
`nil`	Returns `nil` otherwise.

ocnxlFeasibilityAnalysisOptions

```
ocnxlFeasibilityAnalysisOptions(
     [ ?refPoint t_refPoint ]
     [ ?startingstateorpoint t_startingstateorpoint ]
     [ ?startingstatename t_startingstatename ]
     [ ?meetAllGoals t_meetAllGoals ]
     [ ?effort t_effort ]
     )
     => t / nil
```

Description

Specifies options for the Feasibility Analysis run mode. See help on ocnxlRun for help on run modes. This command works only in XL mode.

Arguments

`?refPoint` *`t_refPoint`*	Specifies whether to use a reference point that you have created as a starting place for sizing. It is optional to set this argument when the algorithm specified with the `t_effort` argument is `neocircuitGlobal`. For other values of `t_effort`, set this as `1`. Default value is `0`.
`?meetAllGoals` *`t_meetAllGoals`*	Specifies the stopping criteria for the analysis. By default, it is set to 1 and all operating region specifications are to be met. **Note:** Currently, you cannot set this argument to any value other then 1.
`?effort` *`t_effort`*	Specifies the name of algorithm for optimizing the design to meet the operating region specifications. Possible values are: `neocircuitGlobal`, `conjugateGradient`, `brentPowell`, `hookeJeeves`. The default algorithm is `neocircuitGlobal`.

Value Returned

`t`	Returns t when successful
`nil`	Otherwise, returns nil

Example

```
ocnxlFeasibilityAnalysisOptions(?effort "conjugateGradient")
t
```

ocnxlGetBestPointParams

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

Description

Returns a list of best design points.

Arguments

None

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGetBestPointParams()
```

ocnxlGlobalOptimizationOptions

```
ocnxlGlobalOptimizationOptions(
     [ ?runFullEvaluation t_runFullEvaluation ]
     [ ?refPoint t_refPoint ]
     [ ?meetAllGoals t_meetAllGoals]
     [ ?timeLimit t_timeLimit ]
     [ ?numPoints t_numPoints ]
     [ ?noImprvPoints t_noImprvPoints ]
     [ ?pointsAfterAllSpecsSatisfied t_pointsAfterAllSpecsSatisfied ]
     [ ?startingstateorpoint t_startingstateorpoint ]
     [ ?startingstatename t_startingstatename ]
     )
     => t / nil
```

Description

Lets you specify options for global optimization run. See help on ocnxlRun() for run modes. This command works only in XL mode. See help on `ocnSetXLMode()`

Arguments

`?runFullEvaluation` *`t_runFullEvaluation`*	Sets the program to run full optimization. Default for `runFullEvaluation` is `0`. Possible values are `0` and `1`.
`?refPoint` *`t_refPoint`*	Sets the current point as a starting place for sizing. Default for `refPoint` is `0`. Possible values are `0` and `1`.
`?meetAllGoals` *`t_meetAllGoals`*	Sets the program to run optimization only until all specifications are met. Default for `meetAllGloals` is `0`. Possible values are `0` and `1`.
`?timelimit` *`t_timeLimit`*	Sets the time limit (in minutes) for the optimization run. Default for `timeLimit` is "".
`?numPoints` *`t_numPoints`*	The maximum number of points for the optimization run. Default for `numPoints` is "".
`?noImprvPoints` *`t_noImprvPoints`*	Default for `noImprvPoints` is "".
`?pointsAfterAllSpecsSatisfied` *`t_pointsAfterAllSpecsSatisfied`*	Sets the number of points to be run after all specifications are satisfied. Default for `t_pointsAfterAllSpecsSatisfied` is "".
`?startingstateorpoint` *`t_startingstateorpoint`*	Sets the starting point for the simulation run as reference point or starting state. Default for *`t_startingstateorpoint`* is "". Possible values are `Starting State` or `Reference Point`.
`?startingstatename` `t_startingstatename`	Sets the starting state if the *`startingstateorpoint`* argument is set to `Starting State`.

Value Returned

`t`	Returns `t` if options are specified for global optimization run.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGlobalOptimizationOptions(?runFullEvaluation "1" )
```

```
# Sets global optimization to be run only until all specifications are met.
ocnxlGlobalOptimizationOptions(?startingstateorpoint "Starting State"
?startingstatename "Optimization.PointID.46")
# Sets the optimization to be run with starting point as “Optimization.PointID.46”
state.
```

ocnxlLocalOptimizationOptions

```
ocnxlLocalOptimizationOptions(
     [ ?effort t_effort ]
     [ ?runFullEvaluation t_runFullEvaluation ]
     [ ?meetAllGoals t_meetAllGoals ]
     [ ?timeLimit t_timeLimit ]
     [ ?numPoints t_numPoints ]
     [ ?startingstateorpoint t_startingstateorpoint]
     [ ?startingstatename t_startingstatename] )
     => t / nil
```

Description

Lets you specify options for local optimization run. See help on `ocnxlRun()` for run modes.

Arguments

`?effort` *`t_effort`*
: Value for effort. Default for effort is `coarse`. Possible values are `fine` and `coarse`.

`?runFullEvaluation` *`t_runFullEvaluation`*
: Sets the program to run full optimization. Default for `runFullEvaluation` is `0`. Possible values are `0` and `1`.

`?meetAllGoals` *`t_meetAllGoals`*
: Sets the program to run optimization only until all specifications are met. Default for `meetAllGloals` is `0`. Possible values are `0` and `1`.

`?timeLimit` *`t_timeLimit`*
: Sets the time limit (in minutes) for the optimization run. Default for `timeLimit` is "".

`?numPoints` *`t_numPoints`*
: Sets the maximum number of points for the optimization run. Default for `numPoints` is "".

`?startingstateorpoint` *`t_startingstateorpoint`*
: Sets the starting point for the simulation run as reference point or starting state. Default for *`t_startingstateorpoint`* is "". Possible values are `Starting State` or `Reference Point`.

`?startingstatename` *`t_startingstatename`*
: Sets the starting state if the *`startingstateorpoint`* argument is set to `Starting State`.

Value Returned

`t`	Returns `t` if options are specified for local optimization run.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlLocalOptimizationOptions(?effort "coarse" )
# Sets coarse as the effort for local optimization run.
```

```
ocnxlLocalOptimizationOptions(?startingstateorpoint "Starting State"
?startingstatename "Optimization.PointID.46")
```

```
# Sets the optimization to be run with starting point as “Optimization.PointID.46”
state.
```

ocnxlMonteCarloOptions

```
ocnxlMonteCarloOptions(
     [ ?mcmethod t_mcmethod ]
     [ ?mcNumPoints t_mcNumPoints ]
     [ ?samplingMode t_samplingMode ]
     [ ?saveAllPlots t_saveAllPlots ]
     [ ?saveProcess t_saveProcess ]
     [ ?saveMismatch t_saveMismatch ]
     [ ?useReference t_useReference ]
     [ ?donominal t_donominal ]
     [ ?monteCarloSeed t_monteCarloSeed ]
     [ ?mcStartingRunNumber t_mcStartingRunNumber ]
     [ ?designUnderTest t_designUnderTest]
     [ ?dutIntances t_dutIntances]
     [ ?dutSummary t_dutSummary ]
     [ ?ignoreFlag t_ignoreFlag ]
     [ ?mcNumBins t_mcNumBins ]
     [ ?mcStopEarly t_mcStopEarly ]
     [ ?mcYieldTarget t_mcYieldTarget ]
     [ ?mcYieldAlphaLimit t_mcYieldAlphaLimit ]
     [ ?mcStopMethod t_mcStopMethod ]
     [ ?mcSigmaScaleValue t_mcSigmaScaleValue ]
     [ ?dumpParamMode t_dumpParamMode ]
     [ ?evaluationmode t_evaluationmode ]
     [ ?limitOnOutstandingPoints t_limitOnOutstandingPoints ]
     )
     => t / nil
```

Description

Lets you specify options for Monte Carlo runs. See help on `ocnxlRun()` for run modes.

Arguments

`?mcmethod` *`t_mcmethod`*	Sets the statistical variation method for Monte Carlo runs. Default for `mcmethod` is `all`. Possible values are `global`, `mismatch` and `all`.
`?mcNumPoints` *`t_mcNumPoints`*	Sets the number of points you want to simulate for Monte Carlo runs. Default for `mcNumPoints` is `100`.
`?samplingMode` *`t_samplingMode`*	Sets the default statistical sampling method for Monte Carlo runs. Default for `samplingMode` is `random`. Possible values are `random`, `orthogonal`, and `lhs` (Latin Hypercube).
`?saveAllPlots` *`t_saveAllPlots`*	Saves raw data (`psf` files) for every Monte Carlo iteration so that you can plot a family of curves. Default for `saveAllPlots` is 0. Possible values are `0` and `1`.
`?saveProcess` *`t_saveProcess`*	Controls whether ‘process’ parameters need to be saved to the results database. Default value is `1`. Possible values are `0` and `1`.
`?saveMismatch` *`t_saveMismatch`*	Controls whether ‘mismatch’ parameters need to be saved to the results database. Default value is `0`. Possible values are `0` and `1`.
`?useReference` *`t_useReference`*	Specifies whether to use a schematic point or a reference point that you have created as a starting place for sizing. Possible values are `0` and `1`. The default value is 0.
`?donominal` *`t_donominal`*	Specifies whether to run a simulation at the reference point prior to beginning the Monte Carlo process. Possible values are `0` and `1`. If set to `1`, Spectre will run a simulation at the reference point, and, if this fails, then the sampling process is not initiated and the simulation stops. The default value is 1.
`?monteCarloSeed` *`t_monteCarloSeed`*	Specifies a different seed for Monte Carlo runs. Default for `monteCarloSeed` is `12345`.
`?mcStartingRunNumber` *`t_mcStartingRunNumber`*	Specifies a starting run number for Monte Carlo runs. Default for `mcStartingRunNumber` is `1`.

`?dutSummary` *`t_dutSummary`*	Specifies a list of design under test (DUT) instances for Monte Carlo runs. In this list, you can specify the instances and devices to which mismatch variations must be applied. The format to specify the list is as given below: `<testname%instances%Libname/Cellname/Viewname%Master#testname%instances%modelname%Subcircuit#testname%instances%Schematic%Schematic>` where two DUT instances in the list are separated by a `#` (hash). For example: *`"opamp090:full_diff_opamp_AC:2:1%/I21%acOpenDiff%Subcircuit#opamp090:full_diff_opamp_AC:2:1%/I0/I1%opamp090/ampn/schematic%Master#opamp090:full_diff_opamp_AC:2:1%/I0/M5A, /I0/M3A%Schematic%Schematic"`* Default for `dutSummary` is "".
`?ignoreFlag` *`t_ignoreFlag`*	Argument to specify if the user wants to apply mismatch variations to instances specified with `dutSummary`. Default value is `0`. Set it to `1` if you do not want to apply mismatch variations to instances.
`?mcNumBins` *`t_mcNumBins`*	Argument to specify the number of bins. Set this value when `samplingMode` is `lhs`. Default for `mcNumBins` is "". If not set, simulator uses its own default number of bins. For example, Spectre calculates the number of bins as given below: `numBins = max(t_mcNumBins, (t_mcNumPoints + t_mcStartingRunNumber -1))`
`?designUnderTest` *`t_designUnderTest`*	Design picked up for the test
`?dutIntances` *`t_dutIntances`*	Number of occurance of DUT intances
`?mcYieldTarget` *`tmcYieldTarget`*	Target yield percentage
`?mcStopMethod` *`t_mcStopMethod`*	Sets the statistical variation method to stop Monte Carlo.

`?mcSigmaScaleValue` *`t_mcSigmaScaleValue`*	Sets a sigma scale value

Value Returned

`t`	Returns `t` if options for montecarlo run are specified.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlMonteCarloOptions(?mcmethod "all" ?mcNumPoints “100” ?samplingMode “lhs”
?saveAllPilots “0” ?monteCarloSeed "" ?mcStartingRunNumber "" ?dutSummary ""
?saveProcess “1” ?saveMismatch “0” ?useReference “0” ?doNominal “1” ?mcNumBins
"100")
```

ocnxlSamplingOptions

```
ocnxlSamplingOptions (
     [ ?points t_numberOfPoints ]
     )
     => t / nil
```

Description

Lets you specify options for sampling run. See help on ocnxlRun () for run modes.

Arguments

`?points` *`t_numberOfPoints`*	Specifies the number of points. The default value for points is *200*.

Value Returned

`t`	Returns `t` if the options for the run are specified.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSamplingOptions(?points "500")
#       Sets 500 as the number of points for sampling run.
```

ocnxlSensitivityOptions

```
ocnxlSensitivityOptions(
     [ ?haveDesignParams t_haveDesignParams ]
     [ ?haveProcessParams t_haveProcessParams ]
     [ ?haveDcOp t_haveDcOp ]
     [ ?haveDesignParamSteps t_haveDesignParamSteps ]
     [ ?designParamPercentage t_designParamPercentage ]
     [ ?processSigmaSteps t_processSigmaSteps ]
     [ ?processConfidenceIntervalUniform t_processConfidenceIntervalUniform ]
     [ ?processMethod t_processMethod ]
     [ ?dcOpInfo t_dcOpInfo ]
     [ ?algoritm t_algoritm ]
     [ ?statisticalInfo t_statisticalInfo ]
     )
     => nil
```

Description

Specifies options for the Sensitivity Analysis run mode. See help on ocnxlRun for help on run modes. This command works only in XL mode.

Arguments

`?haveDesignParams` *`t_haveDesignParams`*

Specifies if you want to vary global variables and device parameters. Default value is 0. If you set this to 1, ensure that you have specified at least one sweep variable or parameter and also created a reference point. You also need to set either *`t_haveDesignParamSteps`* or *`t_designParamPercentage`*.

`?haveProcessParams` *`t_haveProcessParams`*

Specifies if you want to vary the statistical process and mismatch parameters. Default value is 0.

`?haveDcOp` *`t_haveDcOp`*

Specifies if you want to save the sensitivity data for specific DC operating point parameters. Default value is 0.

`?haveDesignParamSteps` *`t_haveDesignParamSteps`*

Specifies that you want to vary global variable and device parameter values by a single step from the reference values specified for global variables and parameters in the reference point.

`?designParamPercentage` *`t_designParamPercentage`*

Specifies the percentage of the range of a variable or parameter's value from the reference values by which the process parameters need to be varied. Value range is between 0 and 100.

`?processSigmaSteps` *`t_processSigmaSteps`*

Specifies the number of standard deviations for statistical parameters with normal or log normal distribution. Default value is `1`.

`?processConfidenceIntervalUniform` *`t_processConfidenceIntervalUniform`*

Specifies the percentage range by which statistical parameters with uniform distribution need to be varied. Value range is between `0` and `50`.

`?processMethod` *`t_processMethod`*

	Specifies whether the statistical parameters to be used are process, mismatch or both. Default value is `process`. Possible values are `process`, `mismatch` and `all`.
`?dcOpInfo` *`t_dcOpInfo`*	Specifies DC operating point parameters as input parameters for the sensitivity analysis run.
`?algorithm` *`t_algorithm`*	Name of the algorithm to be used.

Value Returned

`nil`	Returns nil

Example

```
ocnxlSensitivityOptions( ?haveDesignParams "1" ?haveProcessParams "0" ?haveDcOp
"1" ?haveDesignParamSteps "1" ?designParamPercentage "10" ?processSigmaSteps "1"
?processConfidenceIntervalUniform "20" ?processMethod "Process" ?dcOpInfo
"voltage_divider:voltage_divider:1%/R0%i%analogLib/res/spectre%Master%analogLib/
res/spectre#"
nil
```

ocnxlSizeOverCornersOptions

```
ocnxlSizeOverCornersOptions(
     [ ?soclazy t_soclazy ]
     [ ?socoptmethod t_socoptmetjod
     [ ?socreferencepoint t_socreferencepoint ]
     [ ?soceffort t_soceffort ]
     [ ?soctimelimit t_spctimelimit ]
     [ ?socmaxpoints t_socmaxpoints ]
     [ ?sociterations t_sociterations ]
     [ ?socstopifnoimprovement t_socstopifnoimprovement ]
     )
     => nil
```

Description

Provides run options for the Size Over Corners run.

Arguments

`?socoptmethod` *`t_socoptmethod`*

Specifies between Optimization algorithms for iteration sizing runs. The default value is `neocircuitGlobal`. Other values are `brentPowell`, `hookeJeeves`, and `conjugateGradient`.

`?socreferencepoint` *t_socreferencepoint*

Specifies if reference point should be used. The default value is `0`.

`?soceffort` *`t_soceffort`*

This argument is currently not supported.

`?soctimelimit` *`t_soctimelimit`*

Specifies the time limit for the run.

`?socmaxpoints` *`t_socmaxpoints`*

Specifies the maximum number of points processed per iteration. The default value is `3000`.

`?sociterations` *`t_sociterations`*

Specifies the maximum number of sizing iterations for the Size Over Corners run. The default value is `3`.

`?socstopifnoimprovement` *`t_socstopifnoimprovement`*

Specifies if the optimization run should stop if there is no improvement. The default value is `0`.

Value Returned

`nil`	Returns `nil`.

ocnxlStartingPoint

```
ocnxlStartingPoint(
     l_startingPointDetails
     )
     => t / nil
```

Description

Lets you specify a reference point—a starting place for sizing—for Improve Yield, Global Optimization, Feasibility Analysis or Monte Carlo runs.

Arguments

l_startingPointDetails	A list of elements where each element is: (*t_type t_varName t_value*) Where: ■ *t_type* can be a `variable` or `parameter`. ■ *t_varName* is the name of the variable or parameter ■ *t_value* is the value of the variable or parameter.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlStartingPoint('(("variable" "CAP" "2p")
(“parameter” "ether_adcflash/adc_cascode_opamp/schematic/M2/fw" "16.3u")))
```

ocnxlSweepsAndCornersOptions

```
ocnxlSweepsAndCornersOptions(
     [ ?submitpointenabled t_submitpointenabled ]
     )
     => t / nil
```

Description

Lets you specify options for Single Run, Sweeps and Corners runs.

Arguments

`?submitpointenabled` *`t_submitpointenabled`*	Specify `t` to override the active setup with the submit point information.

Value Returned

`t`	Returns `t` if the options are set successfully.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSweepsAndCornersOptions(t) => t
```

ocnxlWorstCaseCornersOptions

```
ocnxlWorstCaseCornersOptions(
     [ ?statisticalInfo t_statisticalInfo ]
     [ ?algorithm t_algorithm ]
     )
     => t / nil
```

Description

Sets the algorithm to be used while running the worst case corner simulation.

Arguments

`?algorithm` *`t_algorithm`*	Specifies the algorithm based on which you want to create the worst case corners. Possible values: `OFAT 3-level`, `OFAT Sweep`, `2^K Factorial`, `Central Composite Design`, and `Full Factorial`.

Value Returned

`t`	Returns `t` if the option is set successfully.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlWorstCaseCornersOptions( ?algorithm "OFAT 3-level" ?grouprun "1" )
```

ocnxlYieldEstimationOptions

```
ocnxlYieldEstimationOptions(
     [ ?useReference t_useReference ]
     [ ?mcMethod t_mcMethod ]
     [ ?samplingMethod t_samplingMethod]
     [ ?mcNumPoints t_mcNumPoints ]
     [ ?mcNumBins t_mcNumBins]
     [ ?monteCarloSeed  t_monteCarloSeed ]
     [ ?speccornerselection t_speccornerselection]
     [ ?haveYieldToStart t_yieldToStart ]
     [ ?yisToStart t_yisToStart ]
     [ ?useVarReduction t_varReductionBy ]
     [ ?varReductionParam t_varReductionParam]
     [ ?numlmport t_numlmport]
     [ ?iterations t_iterations ]
     [ ?designUnderTest t_designUnderTest ]
     [ ?dutInstances t_dutInstances]
     [ ?dutSummary t_dutSummary]
     [ ?ignoreFlag t_ignoreFlag ]
     [ ?yeMethod t_yeMethod ]
     [ ?yeSpecTolerance t_yeSpecTolerance ]
     [ ?yeAngleTolerance t_yeAngleTolerance ]
     )
     => t / nil
```

Description

Lets you specify options for High Yield Estimation run mode. See help on `ocnxlRun()` for run modes.

Arguments

`?useReference t_useReference`

Specifies whether to use a schematic point or a reference point that you have created as a starting place for sizing. The possible values are `0` and `1`. The default value is `0`.

`?mcMethod t_mcMethod` Specifies the yield estimation method to be used. The default value is `all`. The possible values are `global`, `mismatch` and `all`.

`?samplingMethod t_samplingMethod`

Sets the default statistical sampling method for improve yield runs. The default value is `random`. Possible values are `random`, `orthogonal`, and `lhs` (Latin Hypercube).

`?mcNumPoints t_mcNumPoints`

Sets the number of Monte Carlo points you want to simulate. The default value is `200`.

`?mcNumBins t_mcNumBins`

If the selected sampling method is `lhs`, this argument specifies the number of bins (or subdivisions) for `lhs`. Set this value when `samplingMode` is `lhs`.

Default for `mcNumBins` is "". If not set, simulator uses its own default number of bins. For example, Spectre calculates the number of bins as given below:

```
numBins = max(t_mcNumBins, (t_mcNumPoints +
t_mcStartingRunNumber -1))
```

`?monteCarloSeed t_monteCarloSeed`

Specifies a different seed for Monte Carlo runs. Default for `monteCarloSeed` is `12345`.

`?haveYieldToStart t_haveYieldToStart`

Specifies if there is a yield in sigma value from which the high yield estimation method should be applied. The default value is `1`. Possible values `0` and `1`.

`?yisToStart` *`t_yisToStart`*

Specifies a yield in sigma value from which the high yield estimation method should be applied. The default value is `3.0`.

`UseVarReduction`

Enables or disables the statistical variable reduction method.

Default value `auto`, enables this method. Possible values `auto` and `disabled`.

`varReductionParam`

Specifies the parameters for variable reduction method.

`iterations`

Optional argument to specify the number of sizing/Monte Carlo iterations run for each specification.

The default value is `10`.

`dutSummary`

Optional argument to specify a list of design under test (DUT) instances for improve yield runs. In this list, you can specify the instances and devices to which mismatch variations must be applied. The format to specify the list is as given below:

```
<testname%instances%Libname/Cellname/
Viewname%Master#testname%instances%modelname%Subcir
cuit#testname%instances%Schematic%Schematic>
```

where two DUT instances in the list are separated by a `#(hash)`.

```
For example:
"opamp090:full_diff_opamp_AC:2:1%/
I21%acOpenDiff%Subcircuit#opamp090:full_diff_opamp_
AC:2:1%/I0/I1%opamp090/ampn/
schematic%Master#opamp090:full_diff_opamp_AC:2:1%/
I0/M5A, /I0/M3A%Schematic%Schematic"
```

Default for `dutSummary` is "".

`ignoreFlag`

Optional argument to specify if the user wants to apply mismatch variations to instances specified with `dutSummary`.

Default for `ignoreFlag` is `0`. Set it to `1` if you do not want to apply mismatch variations to instances.

`yeMethod`

Specifies the estimation method to be used. This argument takes only one value `Worst Case Distance`.

yeSpecTolerance

Specifies the specification tolerance value. This argument is used to modify the convergence criteria to be used for WCD calculation.

Default value: 0.02

Valid value range: `0.01` to `0.1`

Note: The WCD algorithm converges only when the following two conditions are met:

- Specification value error ratio is less than the value specified by *yeSpecTolerance*, where the specification value error ratio is calculated as:

  ```
  spec_value_error_ratio = abs(spec_value -
  spec_target) / abs(nominal_spec_value -
  spec_target)
  ```

- Angle between the WCD point vector and gradient vector is less than the value specified by *yeAngleTolerance*.

yeAngleTolerance

Specifies the angle tolerance value between WCD point vector and gradient vector.

Default value: `8.0`

Valid value range: `1.0` to `15.0`

For more details, refer to the note given for *yeSpecTolerance*.

Value Returned

`t`	Returns `t` if the options are specified.
`nil`	Returns `nil` otherwise

Example

```
ocnxlYieldEstimationOptions( ?useReference "0" ?mcMethod "all" ?samplingMode
"random" ?mcNumPoints "300" ?mcNumBins "" ?monteCarloSeed "" ?haveYieldToStart "1"
?yisToStart "3.0" ?varReductionBy "auto" ?iterations "20" ?yeMethod "Worst Case
Distance" )
```

ocnxlYieldImprovementOptions

```
ocnxlYieldImprovementOptions(
     [ ?iymethod t_iymethod ]
     [ ?refPoint t_refPoint ]
     [ ?algorithm t_algorithm ]
     [ ?timeLimit t_timeLimit ]
     [ ?iterations t_iterations ]
     [ ?numPoints t_numPoints ]
     [ ?sigmaTarget t_sigmaTarget ]
     [ ?stopIfNoImprovement t_stopIfNoImprovement]
     [ ?runFullEvaluation t_runFullEvaluation ]
     [ ?optimizationMethod t_optimizationMethod ]
     [ ?effort t_effort ]
     [ ?saveCorner t_saveCorner]
     [ ?iysamplingmethod t_iysamplingmethod ]
     [ ?iymontecarlodonominal t_iymontecarlodonominal ]
     [ ?iymontecarloseed t_iymontecarloseed ]
     [ ?iymcnumpoints t_iymcnumpoints ]
     [ ?iymontecarlostartingrun t_iymontecarlostartingrun ]
     [ ?WYCmethod t_WYCmethod ]
     [ ?dutSummary t_dutSummary ]
     [ ?ignoreFlag t_ignoreFlag ]
     )
     => t / nil
```

Description

Lets you specify options for improve yield runs. See help on `ocnxlRun()` for run modes.

Arguments

t_iymethod	The yield improvement method to be used. The default value is `all`. The possible values are `global`, `mismatch` and `all`.
t_refPoint	Specifies whether to use a schematic point or a reference point that you have created as a starting place for sizing. Possible values are `0` and `1`. The default value is `0`.
t_algorithm	The default value is `0`. The possible values are `0` and `1`.
t_timeLimit	Sets a time limit for the improve yield run. The default value is "". The `timeLimit` is in minutes.
t_iterations	Specifies the number of sizing/Monte Carlo iterations, The default value is 3.
t_numPoints	Specifies the maximum number of points processed per iteration. The default value is 3000.
t_sigmaTarget	Allows you to increase the mean of the goal distribution to target (of goal) value even after achieving 100% yield. ADE GXL allows you to achieve 4, 5, or even 6 sigma designs. The default value is `6`. The possible values are `4`, `5`, and `6`.
t_stopIfNoImprovement	Specifies if the yield improvement run must be stopped if there is no yield improvement. The default value is `0`. The possible values are `0 and 1`.
t_runFullEvaluation	Sets the program to run full optimization. Default for `runFullEvaluation` is `0`. Possible values are `0` and `1`.
t_optimizationMethod	Sets the optimization method. Default for `optimizationMethod` is `global`. Possible values are `global` and `local`.
t_effort	Specifies the effort level if you are using local optimization. Default for effort is `fine`. Possible values are `fine` and `coarse`.
t_saveCorner	Specifies if the corner is to be saved.

t_iysamplingmethod	Sets the default statistical sampling method for improve yield runs. The default value is `random`. Possible values are `random` and `lhs`.
t_iymontecarlodonominal	Specifies whether to run a simulation at the reference point prior to beginning the improve yield process. Possible values are `0` and `1`. If set to 1, Spectre will run a simulation at the reference point, and, if this fails, then the sampling process is not initiated and the simulation stops. The default value is 1.
t_iymontecarloseed	Specifies a different seed for Monte Carlo runs. Default for `monteCarloSeed` is `12345`.
t_iymcnumpoints	Sets the number of Monte Carlo points you want to simulate. The default value is `nil`.
t_iymontecarlostartingrun	Specifies the run that Monte Carlo begins with. The default value is 1.

t_dutSummary	Specifies a list of design under test (DUT) instances for improve yield runs. In this list, you can specify the instances and devices to which mismatch variations must be applied. The format to specify the list is as given below:
	
	`<testname%instances%Libname/Cellname/Viewname%Master#testname%instances%modelname%Subcircuit#testname%instances%Schematic%Schematic>`
	
	where two DUT instances in the list are separated by a `#(hash)`.
	
	`For example: “opamp090:full_diff_opamp_AC:2:1%/I21%acOpenDiff%Subcircuit#opamp090:full_diff_opamp_AC:2:1%/I0/I1%opamp090/ampn/schematic%Master#opamp090:full_diff_opamp_AC:2:1%/I0/M5A, /I0/M3A%Schematic%Schematic”`
	
	Default for `dutSummary` is "".

t_ignoreFlag	Specifies if the user wants to apply mismatch variations to instances specified with `dutSummary`. Default for `ignoreFlag` is `0`. Set it to `1` if you do not want to apply mismatch variations to instances.

Value Returned

`t`	Returns `t` if the options are specified.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlYieldImprovementOptions(?iymethod "mismatch" )
# Sets mismatch as the method for yield improvement run.
```

Commands for OCEAN XL Runs

- ocnxlGetRunDistributeOptions
- ocnxlOpenResults
- ocnxlOutputSummary
- ocnxlRun
- ocnxlRunSetupSummary
- ocnxlSetRunDistributeOptions
- ocnxlWaitUntilDone

ocnxlGetRunDistributeOptions

```
ocnxlGetRunDistributeOptions(
     )
     => l_runOptions / nil
```

Description

Returns the run options set for the current setup database.

Arguments

None

Value Returned

l_runOptions	List of run options specified for the current setup database. This list contains the following three values: RunIn: Describes how multiple simulations need to run. Valid values are `Parallel` or `Serial`. DivideJobs: Describes how the ICRPs can be divided among the simulation runs. Valid values are `Specify` or `Equally`. JobLimit: Describes the maximum number of jobs that can run when Divide Jobs is set to `Specify`.
nil	Unsuccessful operation

Example

```
runOpt = ocnxlGetRunDistributeOptions()
runOpt~>??
(JobLimit 2 DivideJobs Specify RunIn
    Parallel
)
```

Related Functions

ocnxlSetRunDistributeOptions

ocnxlOpenResults

```
ocnxlOpenResults(
     [ ?testName t_testName ]
     )
     => t / nil
```

Description

Opens the simulation results at the end of an Ocean XL simulation.When you save an OCEAN script from the ADE XL environment, this function is automatically added to the script.

Arguments

`?testName` *`t_testName`*	Name of the test for which the results are to be opened. In case of a multi-test setup, by default, the results are opened for the first test.l

Value Returned

`t`	If the results are opened successfully.
`nil`	If unsuccessful.

Example

```
;The following code snippet shows how the results can be opened for the test
;"opamplib:ampTest:1" at the end of simulation and use ocnPrint to print the
;outputs.
;
ocnxlRun( ?mode 'sweepsAndCorners ?nominalCornerEnabled t ?allCornersEnabled t
?allSweepsEnabled t)
ocnxlOutputSummary(?exprSummary t ?specSummary t ?detailed t ?wave t)

ocnxlOpenResults(?testName "opamplib:ampTest:1")

selectResults('tran)
ocnPrint( ?output "./bandwidth.txt" ?width 20 ?numSpaces 1 bandwidth(VT("/out") 3
"low"))
```

ocnxlOutputSummary

```
ocnxlOutputSummary(
     [ ?exprSummary g_exprSummary ]
     [ ?specSummary g_specSummary ]
     [ ?yieldSummary g_yieldSummary ]
     [ ?detailed g_detailed ]
     [ ?wave g_wave ]
     [ ?forRun runID ]
     [ ?fileName t_filePath ]
     [ ?printHeader? g_printHeader ]
     )
     => t / nil
```

Description

Generates output summary.

Arguments

`?exprSummary` *`g_exprSummary`*

Determines whether to write the expression information that reports results for outputs having expressions.

Possible values:

- `'OnlySummary`: Writes only the summary expression information. Ignores the value of `?detailed`.
- `'OnlyDetailed`: Writes only the detailed expression information.
- `'BothSummaryAndDetailed`: Writes both summary and detailed information.
- `nil`: Does not write any information about the output expressions.
- `t`: Writes the summary expression information. In addition, it adds the detailed expression information if the `?detailed` argument is `t`.

Default value: `t`

`?specSummary` *`g_specSummary`*

Determines whether to write the specification information that reports results for outputs having specifications.

Possible values:

- `'OnlySummary`: Writes only the summary specification information. Ignores the value of `?detailed`.
- `'OnlyDetailed`: Writes only the detailed specification information.
- `'BothSummaryAndDetailed`: Writes both summary and detailed information.
- `nil`: Does not write any information about the specifications.
- `t`: Writes the summary specification information. In addition, it adds the detailed specification information if the `?detailed` argument is `t`.

Default value: `t`

`?yieldSummary` *`g_yieldSummary`*

Writes detailed yield summary.

Default value: `t`

`?detailed` *`g_detailed`*

Writes the details of expressions and specifications. Otherwise, only summary is printed.

Default value: `t`

Note: Value of this argument is ignored when ?specSummary or ?exprSummary is set to `'OnlySummary`, `'OnlyDetailed`, `'BothSummaryAndDetailed`.

`?wave` *`g_wave`*

Writes the value of expressions evaluating to a waveform as `wave`. When set to `nil`, does not write the expressions with waveforms.

Default value: `t`

`?forRun` *`x_runID`*

RunId for which you want to display the output summary.

Note: Use this argument when the *`?waitUntilDone`* argument of the ocnxlRun command is set to `nil`.

`?printHeader?` *`g_printHeader`*

Writes a default header before the output summary.

Possible values:

- `t`: Always writes a header
- `'WhenFileName`: Writes a header only when fileName is provided by using the `fileName` argument.
- `nil`: Does not write a header.

Default value: `'WhenFileName`

Value Returned

`t`	Returns `t` if the summary is generated.
`nil`	Returns `nil` otherwise.

Example

```
Example 1:
#This will print the detailed expression and specification information. Printing
of the outputs that evaluate to waveforms will be skipped.
ocnxlOutputSummary(?exprSummary 'OnlyDetailed ?specSummary 'OnlyDetailed ?wave
nil)
=>
Detailed Expression Summary:
============================

Parameters: None.
     Test: opamplib:ampTest:1
     Nominal Corner:
           output          value
             MAX            -1.35091
             MIN            -1.50789



Detailed Spec Summary:
======================

Parameters: None.
    Test: opamplib:ampTest:1
    Nominal Corner:
                spec   status   value           spec details
                 MAX   pass     -1.35091        lt      1.2     0.0
                 MIN   fail     -1.50789        range   1.2     2.3
t
```

```
Example 2:
ocnxlOutputSummary()
# This will print the detailed and summary information of expressions and specs for
each sweep point and each corner.

Example 3:
ocnxlOutputSummary(?exprSummary nil)
# This will print the detailed and summary informaition of output expressions. It
will not print any information about the specifications.
Example 4:
ocnxlOutputSummary(?specSummary nil ?detailed nil)
# This will print only the summary of output expressions for each sweep point across
all corners. This will not print any details for expressions. This will also not
print any spec details/summary.

Example 5:

ocnxlOutputSummary(?exprSummary t ?specSummary t ?detailed t ?wave t ?fileName
"myoutputfile")

#Writes all the summary to the file myoutputfile with default header as following
#
#Ocean XL Output Summary for run Ocean.<runNumber> on <DateTime>

(ocnxlOutputSummary ?forRun runid2 ?detailed nil)
#
#Ocean XL Output Summary for the run with runID as runid2

Example 6:
# When " sstatus(fullPrecision t)" & "?printHeader? t"
# This will display the value of the output if "sstatus(fullPrecision t)" and print
the header at the top if "?printHeader?" is set to "t".

ocean> sstatus(fullPrecision t)
=>t
ocean> ocnxlOutputSummary(?exprSummary t ?specSummary t ?detailed t ?wave t
?printHeader? t)
=>
Ocean XL Output Summary for run Ocean.10 on Dec 5 01:05:46 2018

Detailed Expression Summary:
============================
```

```
Parameters: None.
     Test: AC
     Nominal Corner:
           output          value
             /vin          wave
             /out          wave
              PM           27.33218483010016


Expression Summary:
============================

Parameters: None.
     Test: AC
     Nominal Corner:
           output          value
             /vin          wave
             /out          wave
              PM           27.33218483010016


Detailed Spec Summary:
======================
Parameters: None.
     Test: AC
     Nominal Corner:
           spec   status  value               spec details
             PM   pass    27.33218483010016    lt      45.0    0.0


Spec Summary:
=============
Parameters: None.
     Test: AC
          spec   status  value               spec details
            PM   pass    27.33218483010016    lt      45.0    0.0


Example 7:

# When " sstatus(fullPrecision nil)" & "?printHeader? nil"

# This will display only the default digits as output if "sstatus(fullPrecision
nil)" and will not print the header at the top if "?printHeader?" is set to "nil".


ocean> sstatus(fullPrecision nil)

=>nil

ocean> ocnxlOutputSummary(?exprSummary t ?specSummary t ?detailed t ?wave t
?printHeader? nil)

=>

Detailed Expression Summary:
============================

Parameters: None.
     Test: AC
     Nominal Corner:
           output          value
             /vin          wave
```

```
            /out          wave
             PM           27.33218


Expression Summary:
===========================

Parameters: None.
     Test: AC
     Nominal Corner:
           output          value
             /vin          wave
             /out          wave
              PM           27.33218


Detailed Spec Summary:
=====================
Parameters: None.
     Test: AC
     Nominal Corner:
           spec   status  value      spec details
             PM   pass    27.33218    lt   45.0   0.0


Spec Summary:
=============
Parameters: None.
     Test: AC
          spec   status  value      spec details
            PM   pass    27.33218    lt   45.0   0.0
```

Example 8:

```
ocnxlOutputSummary(?fileName "myoutputfile" ?printheader? 'WhenFileName)
```

```
# This will write the summary output to the "myoutputfile" file when the ?fileName
argument is specified.
```

Related Functions

ocnxlRun

ocnxlRun

```
ocnxlRun(
     [ ?mode s_mode ]
     [ ?nominalCornerEnabled g_nominalCornerEnabled ]
     [ ?allCornersEnabled g_allCornersEnabled ]
     [ ?allSweepsEnabled g_allSweepsEnabled ]
     [ ?verboseMode g_verboseMode ]
     [ ?waitUntilDone g_waitUntilDone ]
     [ ?runFinishedCallback g_runFinishedCallback ]
     )
     => t / nil / runID
```

Description

Specifies the run mode for simulation and whether to run the nominal corner, corners and sweeps during simulation. Also specifies whether to report completion of points during simulation.

Arguments

s_mode

Lets you run simulations in one of following modes:

- 'sweepsAndCorners
- 'localOptimization
- 'globalOptimization
- 'monteCarlo
- 'yieldImprovement
- 'sampling
- 'sensitivity
- 'feasibilityAnalysis
- 'Size Over Corners

`?nominalCornerEnabled` *g_nominalCornerEnabled*

Accepts boolean values `t` or `nil`. The default value is `t`. If set to `nil`, ADE XL excludes nominal corners from the simulation run.

`?allCornersEnabled` *g_allCornersEnabled*

Accepts boolean values `t` or `nil`. The default value is `t`. If set to `nil`, ADE XL excludes all corners from the simulation run.

`?allSweepsEnabled` *g_allSweepsEnabled*

Accepts boolean values `t` or `nil`. The default value is `t`. If set to `nil`, ADE XL excludes all sweeps from the simulation run.

`?waitUntilDone` *g_waitUntilDone*

Specifies if OCEAN should wait for this run to complete before executing the next command in the script.

Valid values:

- `t`: Specifies that OCEAN should wait for the completion of this run. This is the default value.
- `nil`: Specifies that the you intend to run multiple OCEAN runs in parallel. In this case, ocean assigns a run id to each ocean run.

Note: Set this argument to `nil` to run multiple OCEAN runs in parallel.

`?verboseMode` *`g_verboseMode`*

Accepts boolean values `t` or `nil`. The default value is `t`. If set to `nil`, ADE XL does not report the progress in the simulation of points in the simulation run.

Note: It is recommended that you specify the value `nil` if you have set up a large number of points.

`?runFinishedCallback` *`g_runFinishedCallback`*

Specifies this run is complete and call back the (what).

Value Returned

`t`	Returns t when the run is successful.
`runID`	When the OCEAN runs are run in parallel, that is, when *`?waitUntilDone`* is set to `nil`, returns the run ID on success.
`nil`	Returns nil when the run is unsuccessful.

Examples

Example 1:

The following command runs an already loaded setup and also specifies that no corner should be run.

```
ocnxlRun(?allCornersEnabled nil)
```

No corner will be run but rest of the setup will be run.

Example 2:

This example runs two setups in parallel.

```
ocnxlJobSetup( '( "blockemail" "1" "configuretimeout" "300" "distributionmethod"
"Local" "lingertimeout" "300" "maxjobs" "8" "name" "ADE XL Default"
"preemptivestart" "1" "reconfigureimmediately" "1" "runtimeout" "-1"
"showerrorwhenretrying" "1" "showoutputlogerror" "0" "startmaxjobsimmed" "1"
"starttimeout" "300" ) )

ocnxlLoadSetupState( "C1" 'retain ?tests t ?vars t ?parameters t ?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid1 = ocnxlRun(?waitUntilDone nil)

ocnxlLoadSetupState( "C4" 'retain ?tests t ?vars t ?parameters t ?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid2 = ocnxlRun(?waitUntilDone nil)

(ocnxlWaitUntilDone 'All)

(ocnxlOutputSummary ?forRun runid2 ?detailed nil)

; The previous command displays run summary for the second run
```

Related Functions

ocnxlWaitUntilDone, ocnxlLoadSetupState, ocnxlJobSetup, ocnxlGetHistory, ocnxlSetRunDistributeOptions

ocnxlRunSetupSummary

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

Description

Generates the run setup summary. It shows how many tests, sweeps and corners are there and whether they are enabled.

Arguments

None.

Value Returned

`t`	Returns `t` if the summary is generated.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlRunSetupSummary()
```

ocnxlSetRunDistributeOptions

```
ocnxlSetRunDistributeOptions(
     [ ?RunIn t_runIn]
     [ ?DivideJobs t_divideJobs]
     [ ?JobLimit n_jobLimit]
     )
     => t / nil
```

Description

Sets the specified run option settings for the current setup database.

Arguments

`?RunIn` *`t_runIn`*

Describes how multiple simulations need to run.

Valid values: `Parallel`, `Serial`.

`?DivideJobs` *`t_divideJobs`*

Specifies how the ICRPs are divided among the simulation runs. Valid values: `Specify`, `Equally`.

`?JobLimit` *`n_jobLimit`*

Specifies the maximum number of jobs that can run when ?DivideJobs is set to `Specify`.

Note: This value is not considered when ?DivideJobs is set to `Equally`.

Value Returned

`t`	Returns `t` when the options are successfully set.
`nil`	Returns `nil` otherwise.

Example

The following example sets the run options to run ICRPs in parallel with a maximum of three jobs per run:

```
ocnxlSetRunDistributeOptions( ?RunIn 'Parallel ?DivideJobs 'Specify ?JobLimit 3)
t
```

Related Functions

ocnxlGetRunDistributeOptions

ocnxlWaitUntilDone

```
ocnxlWaitUntilDone(
     x_runID
     'All
     )
     => t / nil
```

Description

This command waits for an active OCEAN XL run to complete. This command works only in XL mode. See help on `ocnSetXLMode()`.

Important

Use this function only when you are running multiple OCEAN runs in parallel, that is, when you have specified the *`?waitUntilDone`* argument of the ocnxlRun command to `nil`. You can enable parallel run in OCEAN XL scripts by using the ocnxlSetRunDistributeOptions function.

Arguments

x_runID	Specifies the ID of the OCEAN XL run for which OCEAN needs to wait to complete before starting execution of the next command. You can specify the runId returned by the ocnxlRun function or the history name or the handle to the setup database for a run.
`'All`	Specify `'All` if you want to wait for all the OCEAN runs that are currently running.

Value Returned

`t`	Returns `t` if the specified runID is found.
`nil`	Returns `nil` otherwise.

Examples

Example 1

In this example, the ocnxlWaitUntilDone command waits for all OCEAN XL runs that are currently running to complete before moving to the next command in the script.

```
ocnxlLoadSetupState( "C1" 'retain ?tests t ?vars t ?parameters t ?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid1 = ocnxlRun(?waitUntilDone nil)

ocnxlLoadSetupState( "C4" 'retain ?tests t ?vars t ?parameters t ?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid2 = ocnxlRun(?waitUntilDone nil)

(ocnxlWaitUntilDone 'All)

ocnxlOutputSummary()
```

Example 2

In this example, the ocnxlWaitUntilDone command waits for the OCEAN XL run with runID as `runid2` to complete before moving to the next command in the script.

```
runid2 = ocnxlRun(?waitUntilDone nil)
```

```
(ocnxlWaitUntilDone runid2)
ocnxlOutputSummary()
```

Related Function

ocnxlRun

Commands for History

- ocnxlGetCurrentHistory
- ocnxlGetCurrentHistoryId
- ocnxlGetHistory
- ocnxlGetOverwriteHistory
- ocnxlGetOverwriteHistoryName
- ocnxlGetReferenceHistory
- ocnxlHistoryPrefix
- ocnxlRenameCurrentHistory
- ocnxlSetOverwriteHistory
- ocnxlSetOverwriteHistoryName
- ocnxlSetReferenceHistory
- ocnxlWriteDatasheet

ocnxlGetCurrentHistory

```
ocnxlGetCurrentHistory(
     )
     => historyName / nil
```

Description

Returns the current history (checkpoint) name.

Arguments

None

Value Returned

`historyName`	Returns the name of the current history.
`nil`	Returns `nil` in case of an error.

Example

```
ocnxlLoadSetupState( "ac_state1" 'retain ?tests t ?vars t ?parameters t
?currentMode t ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runId = ocnxlRun( ?mode 'sweepsAndCorners ?nominalCornerEnabled t
?allCornersEnabled t ?allSweepsEnabled t ?waitUntilDone nil)

; The following function returns the handle to the results database for the current
history.
rdb=axlReadHistoryResDB(ocnxlGetCurrentHistory() ?session ocnxlGetSession())

; The following function returns the point object for design point 1.
pt = rdb->point(1)
; The following code prints corner name, test name, output name and its value
; for each output of type expression
foreach(out pt->outputs(?type 'expr ?sortBy 'corner)
printf("corner=%s, test=%s, output=%s, value=%L\n" out->cornerName out->testName
out->name out->value))
```

When the above script is run, the results are displayed as shown below:

```
corner=C4_0, test=AC, output=gainBwProd(VF("/OUT")), value=1.068285e+09
corner=C4_0, test=AC, output=Current, value=0.0007904204
corner=C4_0, test=AC, output=Gain, value=49.76433
```

```
corner=C4_0, test=AC, output=UGF, value=5.639488e+08
corner=C4_0, test=TRAN, output=SettlingTime, value=5.911562e-09
corner=C4_0, test=TRAN, output=Swing, value=1.26184
corner=C4_1, test=AC, output=gainBwProd(VF("/OUT")), value=5.452747e+08
corner=C4_1, test=AC, output=Current, value=0.0004168163
corner=C4_1, test=AC, output=Gain, value=46.60983
corner=C4_1, test=AC, output=UGF, value=2.736042e+08
corner=C4_1, test=TRAN, output=SettlingTime, value=7.762304e-09
corner=C4_1, test=TRAN, output=Swing, value=1.05484
corner=nominal, test=AC, output=gainBwProd(VF("/OUT")), value=1.068285e+09
corner=nominal, test=AC, output=Current, value=0.0007904204
corner=nominal, test=AC, output=Gain, value=49.76433
corner=nominal, test=AC, output=UGF, value=5.639488e+08
corner=nominal, test=TRAN, output=SettlingTime, value=5.911562e-09
corner=nominal, test=TRAN, output=Swing, value=1.26184
```

ocnxlGetCurrentHistoryId

```
ocnxlGetCurrentHistoryId(
     [ ?returnSingleEntryIfGroupRun t_returnSingleEntryIfGroupRun ]
     )
     => historyID / nil
```

Description

Returns the ID of the current history (checkpoint).

Arguments

`?returnSingleEntryIfGroupRun` *`t_returnSingleEntryIfGroupRun`*

Specifies if a single history ID is to be returned in case of a group run.

Value Returned

`t`	Returns the ID of the current history.
`nil`	Returns `nil` in case of an error.

Example

```
ocnxlGetCurrentHistoryId()
```

ocnxlGetHistory

```
ocnxlGetHistory(
     x_runID
     )
     => x_setupdbHandle / nil
```

Description

Returns the handle to the history setup database associated with a particular run. You can use this handle to work with the history results.

This command works only in XL mode. See help on `ocnSetXLMode()`.

Arguments

x_runID	ID of the run for which the handle to the history setup database is to be returned.

Value Returned

`x_setupdbHandle`	Returns handle to the history setup database.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlLoadSetupState( "ac_state1" 'retain ?tests t ?vars t ?parameters t
?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runid2 = ocnxlRun(?waitUntilDone nil)

ocnxlLoadSetupState( "tran_state2" 'retain ?tests t ?vars t ?parameters t
?currentMode t
    ?runOptions t ?specs t ?corners t ?extensions t
    ?modelGroups nil ?relxanalysis nil )

runId = ocnxlRun( ?mode 'sweepsAndCorners ?nominalCornerEnabled t
?allCornersEnabled t ?allSweepsEnabled t ?waitUntilDone nil)

histId = ocnxlGetHistory(runId)

ocnxlWaitUntilDone('All)

psfDir = axlGetPointPsfDir(histId "mdltest:testinv:1" ?cornerName ""
?designPointId 1)
```

Related Functions

ocnxlSetRunDistributeOptions, ocnxlRun

ocnxlGetOverwriteHistory

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

Description

Returns the status of overwrite history.

Arguments

None

Value Returned

`t`	Returns `t` if overwrite history is enabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGetOverwriteHistory()
t
```

ocnxlGetOverwriteHistoryName

```
ocnxlGetOverwriteHistoryName(
     )
     => t_historyName / nil
```

Description

Returns name of the history to be overwritten.

Arguments

None

Value Returned

`t_historyName`	Returns name of the history to be overwritten.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlGetOverwriteHistoryName()
Interactive.4
```

ocnxlGetReferenceHistory

```
ocnxlGetReferenceHistory(
     )
     => t_referenceHistoryName / nil
```

Description

Gets the name of the reference history currently set in the OCAEN XL.

Argument

None

Value Returned

`t_referenceHistoryName`	Name of the reference history currently set.
`nil`	If no reference history is set.

Example

```
ocnxlSetXLMode()
...
...
ocnxlSetReferenceHistory(ocnxlGetCurrentHistory())
t
ocnxlGetReferenceHistory()
"Interactive.1"
```

ocnxlHistoryPrefix

```
ocnxlHistoryPrefix(
     t_prefixName
     )
     => t / nil
```

Description

Sets the prefix used in the names of history items created by OCEAN XL runs.

Arguments

t_prefixName	The prefix to be used in the names of history items.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlHistoryPrefix("check")
```

Creates history items with names like `check.0`, `check.1`, and so on.

ocnxlRenameCurrentHistory

```
ocnxlRenameCurrentHistory(
     t_newNameForHistory
     )
     => t / nil
```

Description

Renames the current history (checkpoint).

Arguments

t_newNameForHistory	New name for the current history.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlRenameCurrentHistory("myHistory")
```

ocnxlSetOverwriteHistory

```
ocnxlSetOverwriteHistory(
     g_historyStatus
     )
     => t / nil
```

Description

Sets the status of overwrite history.

Arguments

g_historyStatus	Enables or disables overwrite history. Default value: `t`

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSetOverwriteHistory(t)t
```

ocnxlSetOverwriteHistoryName

```
ocnxlSetOverwriteHistoryName(
     t_historyName
     )
     => t / nil
```

Description

Sets name of the history to be overwritten.

Arguments

`t_historyName`	Name of the history to be overwritten. **Note:** Ensure that you specify the name of an existing history. If no history exists, then this can be set as `Next History Run`.

Value Returned

`t_historyName`	Returns t is successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSetOverwriteHistoryName("Interactive.4")
```

ocnxlSetReferenceHistory

```
ocnxlSetReferenceHistory(
     t_historyName
     [ ?reuseNetlist t_reuseNetlist ]
     [ ?useReferenceResults t_useReferenceResults ]
     )
     => t_referenceHistoryName / nil
```

Description

Sets a reference history for incremental simulation runs in OCEAN. This command works only in XL mode. See help on `ocnSetXLMode()`.

Argument

t_historyName	Name of reference history that you want to use. Use the ocnxlGetCurrentHistory() function to use the current history or give the name of the reference history that you want to use.
t_reuseNetlist	Specifies whether to reuse the netlist in the subsequent runs. If the design has not changes, you can reuse the netlist. Possible values: `t`: Creates an incremental netlist for the new design points. However, for same design points, netlist from the reference history is reused. `nil`: Always creates a new netlist for the design. Default value: `nil`

t_useReferenceResults	(Optional) Specifies whether to reuse the results from the reference history for the incremental simulation run. Possible values: `new`: Creates a new resultset for the incremental simulation run. Does not use the results from the reference history. `copy`: Copies the simulation results of the reference history to the new history item that is created during the incremental simulation run. With this option, OCEAN XL displays the results for only the updated values. `move`: Moves the simulation results of the reference history to the new history item that is created during the incremental simulation run. Default value: `copy`

Value Returned

`t_referenceHistoryName`	Returns `t` if the name of the reference history is set.
`nil`	Returns `nil` if unsuccessful.

Example

In the following example, during the first OCEAN run, the variable `var1` is sweeped for values `1` and `2`. For the next run, the current history is set as the reference history. The default options specify that the netlist will not be reused, but the reference results will be copied to the incremental run. As a result, in the subsequent run, netlist will be created for the entire design, but the results will be generated only for the new design points with the value of `var1` set to 3, 4, and 5. Results for design points with ABC set to `1` or `2` will be copied to the new history.

```
ocnxlSetXLMode()
...
...
ocnxlSweepVar("var1" "1 2")
...
...
ocnxlRun(...)
...
...
ocnxlSetReferenceHistory(ocnxlGetCurrentHistory())
...
ocnxlSweepVar("var1" "1 2 3 4 5")
```

```
...
ocnxlRun() <--- This will only run three (3 4 5) additional points for netlisting
...
...
ocnxlEndXLMode()
```

ocnxlWriteDatasheet

```
ocnxlWriteDatasheet(
     [ ?name t_datasheetName ]
     [ ?directory t_directory ]
     [ ?resultsSummary g_resultsSummary ]
     [ ?testsSummary g_testsSummary ]
     [ ?detailedResults g_detailedResults ]
     [ ?plots g_plots]
     [ ?designVarsSummary g_designVarsSummary ]
     [ ?paramsSummary g_paramsSummary ]
     [ ?cornersSummary g_cornersSummary ]
     [ ?setupSummary g_setupsummary ]
     [ ?launchBrowser g_launchBrowser ]
     )
     => t / nil
```

Description

This command writes a datasheet from the latest OceanXL run.

Arguments

`?name` *`t_datasheetname`*
: Name of the datasheet to be created.

`?directory` *`t_directory`*
: Directory where the datasheet should be created. If unspecified datasheet will be created in the <lib>/<cell>/adexl/documents directory.

`?resultsSummary` *`g_resultsSummary`*
: Boolean argument that controls whether a results summary sheet will be printed or not. Results summary contains spec sheet pass/fail table. Default is `t`.

`?testsSummary` *`g_testsSummary`*
: Boolean argument that controls whether a tests summary sheet will be printed or not. Tests summary contains details about the tests sweeps and corners. Default is `t`.

`?detailedResults` *`g_detailedResults`*
: Boolean argument that controls whether results for all the points will be generated or not. Default is `t`.

`?designVarsSummary` *`g_designVarsSummary`*
: Boolean argument that controls whether design variable information will be generated or not. Default is `t`.

`?paramsSummary` *`g_paramsSummary`*
: Boolean argument that controls whether parameters information will be generated or not. Default is `t`.

`?cornersSummary` *`g_cornersSummary`*
: Boolean argument that controls whether corners information will be generated or not. Default is `t`.

`?setupsummary` *`g_setupsummary`*
: Boolean argument that controls whether setup information will be generated or not.

`?launchBrowser` *`g_launchBrowser`*
: Boolean argument that controls whether the generated datasheet will be displayed in a browser window. Default is `t`.

`?datasheetName` *`t_datasheetName`*	Specifies a title for the datasheet.
`?plots` *`g_plots`*	Boolean argument that controls whether the generated datasheet will include all the plots. Default is `t`.

Value Returned

`t`	Returns `t` if the datasheet is created successfully.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlWriteDatasheet(?name "My datasheet")
=> t
```

Commands for Parametric Sets

- ocnxlParametricSet
- ocnxlLocalParametricSet
- ocnxlSetAllParametersDisabled
- ocnxlSetAllParameterPSetsDisabled
- ocnxlSetAllVariablePSetsDisabled

ocnxlParametricSet

```
ocnxlParametricSet(
     l_paramList
     )
     => l_paramList
```

Description

Creates a parametric set by using the given list of parameters.

Arguments

l_paramList	List of parameter names to be included in the parametric set.

Example

The following example creates a parametric set with two parameters, *vin_ac* and *vdd*:

```
ocnxlParametricSet('("vin_ac" "vdd"))
```

ocnxlLocalParametricSet

```
ocnxlLocalParametricSet(
     t_testName
     l_paramList
     )
     => t / nil
```

Description

Adds a parametric set for local variables in a `maestro` view.

Note: This function does not check if the variables exist or not.

Arguments

t_testName	Name of the test for local variables.
l_paramList	List of grouped parametric variable names.

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

The following example adds a group parametric variables set (`ldc`, `r0`, `Vdc`) into the test `wcd_math_testcases:test1:1`:

```
ocnxlLocalParametricSet("wcd_math_testcases:test1:1" '("Idc""r0""Vdc"))
```

ocnxlSetAllParametersDisabled

```
ocnxlSetAllParametersDisabled(
     g_disabled
     )
     => t / nil
```

Description

Enables or disables all parameters.

Arguments

`g_disabled`	Specify `t` to disable all parameters, and `nil` to enable all parameters.

Value Returned

`t`	Returns `t` if all parameters are enabled or disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSetAllParametersDisabled(t) => t
```

ocnxlSetAllParameterPSetsDisabled

```
ocnxlSetAllParameterPSetsDisabled(
     g_disabled
     )
     => t / nil
```

Description

Enables or disables all parameter parametric sets.

Arguments

g_disabled	Specify `t` to disable all parameter parameteric sets. Specify `nil` to enable all parameter parameteric sets.

Value Returned

`t`	Returns `t` if all parameter parameteric sets are enabled or disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSetAllParameterPSetsDisabled(t)
```

ocnxlSetAllVariablePSetsDisabled

```
ocnxlSetAllVariablePSetsDisabled(
     g_disabled
     )
     => t / nil
```

Description

Enables or disables all variable parametric sets.

Arguments

g_disabled	Specify `t` to disable all variable parameteric sets. Specify `nil` to enable all variable parameteric sets.

Value Returned

`t`	Returns `t` if all variable parameteric sets are enabled or disabled.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlSetAllVariablePSetsDisabled(t)
```

Commands for Pre-run Scripts

- ocnxlGetPointId
- ocnxlLoadCurrentEnvironment
- ocnxlMCIterNum
- ocnxlPreRunScript
- ocnxlRunCalibration
- ocnxlSetCalibration
- ocnxlSetMCdut
- ocnxlSetPreRunScriptEnabled
- ocnxlUpdatePointVariable

Note: Any variable used in a pre-run script without scope definition will make it a global variable. To avoid this, use the let() command.

ocnxlGetPointId

```
ocnxlGetPointId(
     )
     => x_pointID / nil
```

Description

Returns the ID of the current simulation point. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

None

Value Returned

`x_pointId`	Returns the ID of current simulation point
`nil`	Returns `nil` otherwise

Example

```
id=ocnxlGetPointId()
```

ocnxlLoadCurrentEnvironment

```
ocnxlLoadCurrentEnvironment(
     [ ?noAnalysis g_noAnalysis ]
     )
     => t / nil
```

Description

Loads all the current environment settings, such as variables, analyses, etc., from the main test setup into the pre-run simulation environment. It also sets the results and netlist directory for the pre-calibration simulations based on the results directory for the current point.

For example, if the results directory of point `4` is `$AXL_PROJECT_DIR/myLib/myCell/myView/results/data/Interactive.1/4/myTest`, the netlist directory for the pre-calibration simulation run will be `$AXL_PROJECT_DIR/myLib/myCell/myView/results/data/Interactive.1/4/myTest/preSim/netlist` and the results directory will be `$AXL_PROJECT_DIR/myLib/myCell/myView/results/data/Interactive.1/4/myTest/preSim/psf`. You can specify a different results directory for the pre-calibration simulation run by using the resultsDir OCEAN command.

This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Note: Variables updated with `ocnxlUpdatePointVariable` become part of the current environment. `ocnxLoadCurrentEnvironment` will load all variables from the current environment. If your script requires a variable to be initialized and you also set the value with `ocnxlUpdatePointVariable`, the variable must be explicitly initialized after the call to `ocnxlLoadCurrentEnvironment`. For example, `desVar("myout" 1)`.

Arguments

`?noAnalysis` *`g_noAnalysis`*

Specifies if a pre-run simulation inherits the analysis details from the main test setup.

Possible values:

`t`: The pre-run simulation inherits all of the test setup, excluding the analysis, from the main simulation. You must define the analysis in the pre-run script itself.

`nil`: The pre-run simulation inherits all of the test setup, including the analysis, from the main simulation.

Default value: `nil`

Value Returned

`t`	Returns `t` if the test simulation setup is successfully read.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlLoadCurrentEnvironment(t)
analysis('tran ?stop 10u)
```

Related Functions

ocnxlSetMCdut, ocnxlSetMCignore

ocnxlMCIterNum

```
ocnxlMCIterNum(
     )
     => x_iterNum / nil
```

Description

Returns the current iteration number of the main Monte Carlo simulation run. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

None

Value Returned

`x_iterNum`	Returns the iteration number of main Monte Carlo simulation run
`nil`	Returns `nil` otherwise

Example

```
x=ocnxlMCIterNum()
when( equal(x 1) then
  initialize();
)
```

ocnxlPreRunScript

```
ocnxlPreRunScript(
     t_fileName
     )
     => t / nil
```

Description

Specifies the pre-run script file containing the OCEAN commands that need to be run before the simulation starts. This function must be used within a test setup block (starting with ocnxlBeginTest and ending with ocnxlEndTest) in your OCEAN script file.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

t_fileName	Path to the pre-run script file.

Value Returned

`t`	Returns `t` if file exists.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlBeginTest("myTest")
...
ocnxlPreRunScriptEnabled(t)
ocnxlPreRunScript("/net/scripts/myPreRunScript")
...
ocnxlEndTest()
```

Related Functions

ocnxlSetPreRunScriptEnabled

ocnxlRunCalibration

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

Description

Starts the simulation required to calibrate the simulation setup. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

None

Value Returned

`t`	Returns `t` if the simulation run is successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlRunCalibration()
t
```

ocnxlSetCalibration

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

Description

Sets up a single iteration Monte Carlo calibration run by inheriting statistical parameter information from the main Monte Carlo simulation run. The starting iteration number for the calibration run is set to the current iteration number of the main Monte Carlo simulation run. This command must be used only in a pre-run script.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

None

Value Returned

`t`	Returns `t` if successful
`nil`	Returns `nil` otherwise

Example

```
ocnxlSetCalibration()
t
```

ocnxlSetMCdut

```
ocnxlSetMCdut(
     t_instName
     )
     => t / nil
```

Description

Sets a design instance to be used in a pre-run script for Monte Carlo calibration. If set, the specified subcircuit instance has process and mismatch variations applied to it and the unspecified instances only have process variations. All subcircuits instantiated under the specified instance also have process and mismatch enabled. By default, mismatch variations are applied to all the subcircuit instances in the design and process variations are applied globally. This allows the testbench to change and not affect the variations seen by the original design.

Note: This function is to be used in a pre-run script and only applies to Monte Carlo analysis. Execute this function after ocnxlSetCalibration.

To set the `ignore` parameter, see ocnxlSetMCignore.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

t_instName	Specifies the name of design instance.

Value Returned

`t`	Returns `t` if the design instance is set successfully
`nil`	Returns `nil` otherwise

Example

```
;Set the MC iteration number, etc. to match the main simulation
ocnxlSetCalibration()

;Set the DUT instance of the pre run design
ocnxlSetMCdut("I0")
; For one test you can specify multiple instances to restrict mismatch variation to
; these instances:
ocnxlSetMCdut("I0.M0 I0.M1" )
```

ocnxlSetMCignore

```
ocnxlSetMCignore(
     t_ignoreInstNames
     )
     => t / nil
```

Description

Sets the `ignore` parameter to be used in a pre-run script for Monte Carlo calibration. When the `ignore` parameter is set, no mismatch variation is applied to the specified subcircuit instances or all the subcircuits instantiated under this instance. Otherwise, by default, mismatch variation is applied to all the subcircuit instances in the design.

Calling this function is not required. The `ignore` parameter setting is inherited from the main environment if this function is not called.

To set the `dut` parameter, see ocnxlSetMCdut.

Note: This function is to be used in a pre-run script and only applies to Monte Carlo analysis. Execute this function after ocnxlSetCalibration.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

`t_ignoreInstNames`	Specify a space-separated list of instances. If set, no variation is applied to the specified subcircuit instances. In addition, all subcircuits instantiated under this instance do not have variation enabled. By default, the mismatch is applied to all subcircuit instances in the design and the process is applied globally.

Value Returned

`t`	Returns `t` if the `ignore` parameter is set successfully.
`nil`	Returns `nil` otherwise.

Example

```
;Set the MC iteration number, etc. to match the main simulation
ocnxlSetCalibration()

; ignore mismatch variation on specified instances:
```

```
ocnxlSetMCignore("I2.M10 I2.M11")

; ignore mismatch variation on all instances of I3 and below.:
ocnxlSetMCignore("I3")
```

ocnxlSetPreRunScriptEnabled

```
ocnxlSetPreRunScriptEnabled(
     g_enabled
     )
     => t / nil
```

Description

Specifies if running the pre-run scripts through OCEAN scripts should be enabled.

This command works only in the XL mode. See help on `ocnSetXLMode()`.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

g_enabled	Specifies if running pre-run scripts should be enabled. Default value: `t` Possible values: `t`, `nil`

Value Returned

`t`	Returns `t` if successful.
`nil`	Returns `nil` otherwise.

Example

```
ocnxlBeginTest("myTest")
...
ocnxlSetPreRunScriptEnabled(t)
ocnxlPreRunScript("/net/scripts/myPreRunScript")
...
ocnxlEndTest()
```

ocnxlUpdatePointVariable

```
ocnxlUpdatePointVariable(
     t_paramName
     t_paramValue
     )
     => t / nil
```

Description

Updates the value of a parameter or variable in the simulation setup. This command must be used only in a pre-run script.

Note: The variables updated with `ocnxlUpdatePointVariable` become part of the current environment. If your pre-run script requires a variable to be initialized and you are setting its value with `ocnxlUpdatePointVariable`, the variable must be explicitly initialized after the call to ocnxlLoadCurrentEnvironment. See example.

For more information, see Executing Pre-run Scripts before Simulation Runs.

Arguments

t_paramName	Parameter name
t_paramValue	Parameter value

Value Returned

`t`	Returns `t` if parameter value is successfully updated
`nil`	Returns `nil` otherwise

Example

```
;;   Montecarlo Calibration Example
;; Inherit test setup from the current point
ocnxlLoadCurrentEnvironment( ?noAnalysis t)
desVar("myout" 1)

;; Apply changes if any e.g set a different analysis for calibration run
analysis('dc ?anaName "dc"  ?saveOppoint t ?write "spectre.dc"
        ?oppoint "rawfile"  ?maxiters "150"  ?maxsteps "10000"  ?annotate "status"
        )
;; The following command sets the calibration run. It will iherit main
;; montecarlo options for single iteration run.
ocnxlSetCalibration()
```

```
;; An example to calculate calibrated value using successive approximation method.
;; ocnxlRunCalibration() will run monte carlo simulation for single iteration.
ocnxlRunCalibration()
myout = IDC("/R0/PLUS")

;; Add this value as ADE XL output so that it can be viewed in outputs window.
;; An output name Calibrated_ParamName will be added for each point.
ocnxlAddOrUpdateOutput("myout" myout)

;; Update the main simulation environment with the calibrated result.
ocnxlUpdatePointVariable("myout" sprintf( nil "%L" myout))
```

Commands for Reliability Analysis

- ocnxlAddRelxSetup
- ocnxlAddRelxScenariosSetup
- ocnxlDisableRelxScenariosSetup
- ocnxlDisableRelxSetup
- ocnxlSetRelxAnalysisEnabled
- ocnxlSetRelxScenariosEnabled
- ocnxlSetRelxEnabledForPreRun

ocnxlAddRelxSetup

```
ocnxlAddRelxSetup(
     t_relxSetupName
     t_freshTest
     t_stressTest
     t_agedTest
     [ ?stressVarList l_stressVarList ]
     [ ?agedVarList l_agedVarList ]
     [ ?freshTestEnabled t_freshTestEnabled ]
     [ ?stressTestEnabled t_stressTestEnabled ]
     [ ?agedTestEnabled t_agedTestEnabled ]
     [ ?stressFile t_stressFile ]
     )
     => t / nil
```

Description

Adds a new reliability analysis setup with the specified fresh, stress, and aged tests and any variables for which the values need to be overridden.

Arguments

t_relxSetupName Specifies a unique name for the new reliability analysis setup

t_freshTest Specifies the name of the test you want to use for running fresh simulation

t_stressTest Specifies the name of the test you want to use for running stress simulation

t_agedTest Specifies the name of the test you want to use for running aging simulation

?stressVarList *l_stressVarList*

Provides a list of variables for which you want to modify the values. This list specifies values only for a stress simulation.

?agedVarList *l_agedVarList*

Provides the list of variables for which you want to modify the values. This list specifies values only for an aging simulation.

?freshTestEnabled *t_freshTestEnabled*

Specifies that a frest test is enabled

?stressTestEnabled *t_stressTestEnabled*

Specifies that a stress test is enabled

`?agedTestEnabled` *`t_agedTestEnabled`*

Specifies that an aging test is enabled

`?stressFile` *`t_stressFile`*

Specifies the name of the stress file you want to use

Value Returned

`t`	Returns `t` if the analysis is successfully added
`nil`	Returns `nil` otherwise

Example

```
ocnxlAddRelxSetup("my_relx" "fresh_test" "stress_test" "aged_test"
?stressVarList '(("CAP" "100f") ("RES" "10K")))
```

ocnxlAddRelxScenariosSetup

```
ocnxlAddRelxScenariosSetup(
     t_relxName
     t_freshTest
     t_stressTest
     t_agedTest
     [ ?freshTestEnabled t_freshTestEnabled ]
     [ ?stressTestEnabled t_stressTestEnabled ]
     [ ?agedTestEnabled t_agedTestEnabled ]
     [ ?scenariosList t_scenariosList ]
     [ ?relxOptions t_relxOptions ]
     )
     => t / nil
```

Description

Adds a new scenario setup in the reliability analysis setup.

Arguments

t_relxScenarioName

Specifies the name of the new scenario setup

t_freshTest Specifies the name of the fresh test for this scenario

t_stressTest Specifies the name of the stress test for this scenario

t_agedTest Specifies the name of the aged test for this scenario

?freshTestEnabled *t_freshTestEnabled*

Specifies whether the fresh test is enabled or disabled

?stressTestEnabled *t_stressTestEnabled*

Specifies whether the stress test is enabled or disabled

?agedTestEnabled *t_agedTestEnabled*

Specifies whether the aged test is enabled or disabled

?scenariosList *t_scenariosList*

Specifies the list of reliability scenarios and corner values for the fresh, stress, and aged tests in each scenario.

?relxOptions *t_relxOptions*

Specifies the reliability options to be used for this scenario

Value Returned

`t`	Returns `t` if the scenario is successfully added to the setup
`nil`	Returns `nil` otherwise

Example

```
ocnSetXLMode("assembler")
ocnxlProjectDir( "./simulation" )
ocnxlTargetCellView( "bertlink" "test" "maestro" )

ocnxlAddRelxScenariosSetup( "test" "bertlink:osc13:1" "bertlink:osc13:1"
"bertlink:osc13:1" ?freshTestEnabled t ?stressTestEnabled t ?agedTestEnabled t
?scenariosList '("Scenario_1 C0 C4 C1" "Scenario_2 _default C3 C1") ?relxOptions
'(nil simulator "spectre" relxOpts (((relxOpts showunageddevices) "none")
((relxOpts enableRelxpert) t) ((relxOpts anaMode) "TMI") ((relxOpts
enableSelfheating) t) ((relxOpts gaSaveResults) "none") ((relxOpts nativeMode)
"Spectre native"))))
ocnxlAddRelxScenariosSetup( "gradualaging" "gradualaging" "gradualaging"
"gradualaging" ?freshTestEnabled t ?stressTestEnabled t ?agedTestEnabled t
?scenariosList '("Scenario_1 C1 C3 C2" "Scenario_2 C0 C4 C1") ?relxOptions '(nil
simulator "spectre" relxOpts (((relxOpts enableGradualAging) t) ((relxOpts
agingPointsType) "Gradual aging") ((relxOpts gaAgePoints) "10 20") ((relxOpts
showunageddevices) "none") ((relxOpts anaMode) "TMI") ((relxOpts
enableSelfheating) t) ((relxOpts gaSaveResults) "none") ((relxOpts mosAgingTime)
"10 20") ((relxOpts nativeMode) "Spectre native") ((relxOpts enableAging) t))))
```

ocnxlDisableRelxScenariosSetup

```
ocnxlDisableRelxScenariosSetup(
     t_relxSetupName
     )
     => t / nil
```

Description

Disables the specified reliability scenario setup.

Arguments

t_relxScenarioName

Specifies the name of the reliability scenario setup to be disabled

Value Returned

`t`	Returns `t` if the scenario setup is successfully disabled
`nil`	Returns `nil` otherwise

Example

```
ocnSetXLMode("assembler")
ocnxlProjectDir( "./simulation" )
ocnxlTargetCellView( "bertlink" "test" "maestro" )

ocnxlAddRelxScenariosSetup( "test" "bertlink:osc13:1" "bertlink:osc13:1"
"bertlink:osc13:1" ?freshTestEnabled t ?stressTestEnabled t ?agedTestEnabled t
?scenariosList '("Scenario_1 C0 C4 C1" "Scenario_2 _default C3 C1") ?relxOptions
'(nil simulator "spectre" relxOpts (((relxOpts showunageddevices) "none")
((relxOpts enableRelxpert) t) ((relxOpts anaMode) "TMI") ((relxOpts
enableSelfheating) t) ((relxOpts gaSaveResults) "none") ((relxOpts nativeMode)
"Spectre native"))))

ocnxlDisableRelxScenariosSetup("test")
```

ocnxlDisableRelxSetup

```
ocnxlDisableRelxSetup(
     t_relxSetupName
     )
     => t / nil
```

Description

Disables the specified reliability analysis setup.

Arguments

t_relxSetupName	Specifies the name of the reliability analysis setup that is to be disabled

Value Returned

`t`	Returns `t` if the analysis is successfully disabled
`nil`	Returns `nil` otherwise

Example

```
ocnSetXLMode("assembler")
ocnxlProjectDir( "./simulation" )
ocnxlTargetCellView( "bertlink" "test" "maestro" )

ocnxlDisableRelxSetup("my_relx")
```

ocnxlSetRelxAnalysisEnabled

```
ocnxlSetRelxAnalysisEnabled(
     g_enable
     )
     => t / nil
```

Description

Enables or disables reliability analysis for the setup based on the input argument.

Arguments

g_enable	Enables reliability analysis if the value specified is `t`. Disables reliability analysis if the value specified is `nil`.

Value Returned

`t`	Returns `t` if the analysis is successfully enabled
`nil`	Returns `nil` otherwise

Example

```
ocnxlSetRelxAnalysisEnabled(t)
```

ocnxlSetRelxScenariosEnabled

```
ocnxlSetRelxScenariosEnabled(
     g_enabled
     )
     => t / nil
```

Description

Enables or disables the complete reliability scenarios setup.

Argument

g_enabled	Specifies the enabled or disabled status for the reliability scenario setup

Value Returned

`t`	Returns `t` if the scenario setup is successfully disabled
`nil`	Returns `nil` otherwise

Example

```
ocnSetXLMode("assembler")
ocnxlProjectDir( "./simulation" )
ocnxlTargetCellView( "bertlink" "test" "maestro" )

ocnxlAddRelxScenariosSetup( "test" "bertlink:osc13:1" "bertlink:osc13:1"
"bertlink:osc13:1" ?freshTestEnabled t ?stressTestEnabled t ?agedTestEnabled t
?scenariosList '("Scenario_1 C0 C4 C1" "Scenario_2 _default C3 C1") ?relxOptions
'(nil simulator "spectre" relxOpts (((relxOpts showunageddevices) "none")
((relxOpts enableRelxpert) t) ((relxOpts anaMode) "TMI") ((relxOpts
enableSelfheating) t) ((relxOpts gaSaveResults) "none") ((relxOpts nativeMode)
"Spectre native"))))

ocnxlSetRelxScenariosEnabled(t)
```

ocnxlSetRelxEnabledForPreRun

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

Description

Updates the reliability setup based on the corresponding setup from the current environment. Use this function in the pre-run script to print the reliability setup in the pre-run netlist.

Note: Ensure that all the current environment settings are loaded from the main test setup into the pre-run simulation environment before you invoke this function.

Argument

None

Value Returned

`t`	Returns `t` if the operation is successful
`nil`	Returns `nil` if there is an error

Example

Run the `ocnxlLoadCurrentEnvironment` function to load the current environment settings into the pre-run simulation environment.

```
ocnxlLoadCurrentEnvironment(?noAnalysis t)
```

Use the following function in the pre-run script to update the reliability setup from the current environment and to print it in the pre-run netlist.

```
ocnxlSetRelxEnabledForPreRun()
```

Commands for EAD

You can use the OCEAN commands described in this section to configure the settings to be considered by a simulation whose results are to be used in the Electrically-Aware Design flow. The settings correspond to the UI options that can be set using the EAD Setup form described in *Chapter 2 of Virtuoso Electrically Aware Design Flow Guide*.

- ocnxlEADAddMeasurement
- ocnxlEADCreateDataSet
- ocnxlEADEnableLiveProcessing
- ocnxlEADSelectAllSignals
- ocnxlEADSetDutMaster
- ocnxlEADSetHierarchyLevel
- ocnxlEADSetWaveFormClipping

ocnxlEADAddMeasurement

```
ocnxlEADAddMeasurement(
     t_measName
     [ ?n_scaleFactor scaleFactor ]
     )
     => t / nil
```

Description

Specifies the name of the current or voltage data to be saved from simulation results. A new output is created in the outputs setup of the adexl or maestro view.

Arguments

t_maesName	Name of the current or voltage data to be saved.
?n_scaleFactor *n_scaleFactor*	Option used to scale the electrical data while transferring from schematic to layout. Default value: `1.0`

Value Returned

`t`	The electrical data option and scale factor is saved successfully in the setup
`nil`	The electrical data is not saved

Example

The following example code adds three measurements:

```
ocnxlEADAddMeasurement("Idc" ?scaleFactor 1)
ocnxlEADAddMeasurement("Isignal")
ocnxlEADAddMeasurement("Vmin" ?scaleFactor 1.5)
```

ocnxlEADCreateDataSet

```
ocnxlEADCreateDataSet(
     [ ?lib t_libName ]
     [ ?cell t_cellName ]
     [ ?view t_viewName ]
     [ ?userDataSetName t_dataSetName ]
     )
     => t / nil
```

Description

Creates and saves an EAD dataset for all or for the specified master DUTs. . This command is equivalent to the *Create* command in the EAD results view of ADE XL or ADE Assembler.

Arguments

`?lib` *`t_libName`*	Name of the library.
`?cell` *`t_cellName`*	Name of the cell.
`?view` *`t_viewName`*	Name of the view.
`?userDataSetName` *`t_dataSetName`*	Name to be used for the dataset. Default value: `"dataset"`

Value Returned

`t`	When the dataset is saved successfully
`nil`	When the dataset is not saved

Example

The following example creates a dataset with the name `myDataSet` for all master DUTs:

```
ocnxlEADCreateDataSet(?userDataSetName "myDataSet")
=> t
```

The following example creates datasets, `EM1` and `EM2` for the specified master DUTs:

```
ocnxlEADCreateDataSet(?lib "myLib1" ?cell "myCell1" ?view "schematic"
?userDataSetName "EM1")
ocnxlEADCreateDataSet(?lib "myLib2" ?cell "myCell2" ?view "schematic"
```

```
?userDataSetName "EM2")
=> t
```

ocnxlEADEnableLiveProcessing

```
ocnxlEADEnableLiveProcessing(
     g_enabled
     )
     => t / nil
```

Description

Specifies whether the current waveform has to be processed during simulation or post simulation.

Arguments

g_enabled	Specifies when to process the current waveforms. Possible values: ■ `t`: Processes the current waveform during simulation. When you set this argument to `t`, only the processed electrical data is saved to the disk. This data is used while performing static electromigration checks. This command is equivalent to the *Process Waveforms Inside Simulator* option on the EAD Setup form. ■ `nil`: Processes the current waveform after simulation. When you set this argument to `nil`, the current waveforms are saved on the disk. The saved data can be processed later while performing dynamic electromigration checks, for example, Peak EM or RMS. This command is equivalent to the *Process Waveforms Post Simulation* option on the EAD Setup form.

Value Returned

`t`	When the specified option is saved successfully in the setup
`nil`	When the specified option is not saved in the setup

Example

The following example shows how to use this command:

```
ocnxlEADEnableLiveProcessing(nil)
=> t
```

ocnxlEADSelectAllSignals

```
ocnxlEADSelectAllSignals(
     g_enabled
     )
     => t / nil
```

Description

Specifies whether the current (or voltage) data is to be saved for **all** the instance terminals and terminals (or signals for the voltage data); or **only** for a selected set of instance terminals and terminals (or signals for the voltage data) in the selected design. This command is equivalent to the *All Signals* field on the EAD Setup form. The signals can be selected using the Parasitics & Electrical Setup assistant in Virtuoso Schematic Editor XL.

Arguments

g_enabled	Enables or disables saving of data for all or selected instance terminals, and terminals or signals. Possible values: ■ `t`: Current or voltage data will be saved for all the instance terminals and terminals or signals in the selected design. ■ `nil`: Current or voltage data will be saved only for selected set of instance terminals and terminals or signals in the selected design.

Value Returned

`t`	When the specified option is saved successfully in the setup
`nil`	When the specified option is not saved in the setup

Example

The following example shows how to use this command:

```
ocnxlEADSelectAllSignals(t)
=> t
```

ocnxlEADSetDutMaster

```
ocnxlEADSetDutMaster(
     t_libName
     t_cellName
     )
     => t / nil
```

Description

Saves the library and cell names of the DUT master for which EM analysis is to be performed.

Arguments

t_libName	Library name of your design
t_cellName	Cell name of your design

Value Returned

`t`	When the library and cell names are saved successfully in the setup
`nil`	When the specified options are not saved in the setup

Example

The following example shows how to use this command:

```
ocnxlEADSetDutMaster("testLib" "buffer")
=> t
```

ocnxlEADSetHierarchyLevel

```
ocnxlEADSetHierarchyLevel(
     x_hierLevel
     )
     => t / nil
```

Description

Sets the value for the design hierarchy level upto which the current or voltage data for all the instance terminals, and terminals or signals will be saved. This command is equivalent to the *Hierarchy Stop Level* field on the EAD Setup form.

Arguments

x_hierLevel	A non-negative numeric value that specifies the number of hierarchy levels for which current or voltage data is to be saved.

Value Returned

`t`	When the value for hierarchy level is saved successfully in the setup
`nil`	When the specified options are not saved in the setup

Examples

The following examples show how to use this command:

```
; example 1
ocnxlEADSetHierarchyLevel(99)
=> t
; example 2

ocnxlEADSetHierarchyLevel(0)
=> t
```

ocnxlEADSetWaveFormClipping

```
ocnxlEADSetWaveFormClipping(
     [ ?t_clipFrom t_clipFrom ]
     [ ?t_clipTo t_clipTo ]
     )
     => t / nil
```

Description

Sets a start time and an end time for which an electrical waveform is to be processed. This command is equivalent to the *Clip Waveforms - From* and *To* fields on the EAD Setup form.

Arguments

`?t_clipFrom` *`t_clipFrom`*	Starting time from which the waveform is to be clipped Default value: `nil`
`?t_clipTo` *`t_clipTo`*	Ending time upto which the waveform is to be clipped Default value: `nil`

Value Returned

`t`	When the value for waveform clipping is saved successfully
`nil`	When the specified options are not saved successfully

Example

The following example shows how to use this command:

```
ocnxlEADSetWaveFormClipping(?clipFrom "0.01u" ?clipTo "0.1u")
=> t
```