Virtuoso ADE SKILL Reference - Part II

Product Version ICADVM20.1
October 2020

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

Contents

Preface

This manual describes the SKILL functions that you can use with Virtuoso Analog Design Environment XL and GXL. This manual assumes you are familiar with the Cadence SKILL™ language.

The files containing the SKILL functions provided for use with Analog Design Environment XL and Analog Design Environment GXL are installed in various subdirectories under `your_install_dir/tools/dfII/group/davinci/src`. You can check the introductory paragraph of each chapter for specific directory locations.

This manual assumes you are familiar with the Cadence SKILL™ language.

The preface contains the following:

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

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*

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Virtuoso Design Environment SKILL Reference*
- *Cadence Application Infrastructure User Guide*

Virtuoso Tools

- *Cadence SKILL Language User Guide*.
- *Cadence SKILL Language Reference*.
- *Cadence SKILL++ Object System Reference*.
- *OCEAN Reference*.
- *Virtuoso Analog Design Environment XL User Guide*
- *Virtuoso Analog Design Environment GXL User Guide*
- *Virtuoso Design Environment SKILL Functions Reference*

Additional Learning Resources

Video Library

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

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

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

Virtuoso Videos Book

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

Rapid Adoption Kits

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

In addition, Cadence offers the following training courses on Virtuoso Analog Design Environment XL and the related flows:

- Virtuoso Analog Design Environment
- Virtuoso Schematic Editor
- Analog Modeling with Verilog-A
- Behavioral Modeling with Verilog-AMS
- Real Modeling with Verilog-AMS
- Spectre Simulations Using Virtuoso ADE
- Virtuoso UltraSim Full-Chip Simulator
- Virtuoso Simulation for Advanced Nodes

Cadence also 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

Session-Related Functions

Working with ADE XL Session in SKILL Scripts

Every ADE XL instantiation has an ADE XL session associated with it. If the ADE XL GUI is open, you can access the session by using the axlGetWindowSession function. It returns the session of the active ADE XL window.

If you are using a SKILL script in the non-GUI mode, you need to create a new ADE XL session by using the axlCreateSession function.

By using the handle to the session, you can access the existing setup database or create a new database. For details on the setup-related SKILL functions, refer to Chapter 2.

All the session-related functions are listed in the table given below.

Session-Related SKILL Functions

Function	Description
axlCloseSession	Closes the specified ADE XL session.
axlCloseSessionInWindow	Closes the ADE XL session in the current window, if there is one opened.
axlCreateSession	Creates a new ADE XL session with the specified name. You can use this function to create a new session before running an ADE XL SKILL code in the non-GUI mode,
axlGetMainSetupDB	Returns a handle to the working setup database of the named ADE XL session.
axlGetSessionCellName	Returns the cell name associated with the session or setup database.
axlGetSessionLibName	Returns the library name associated with the given session or setup database.

Session-Related SKILL Functions, *continued*

Function	Description
axlGetSessionViewName	Returns the view name associated with the given session or setup database.
axlGetSessionWindowNumber	Returns a unique integer representing a number corresponding to a given ADE (G) XL session name.
axlGetToolSession	ADE XL internally maintains a unique in-memory identifier for each active test. This function returns that unique identifier for the specified test.
axlGetWindowSession	Returns the ADE XL session associated with a window.
axlGetCurrentResultSimulationHost	This is a callback function that runs from a `Results` table context menu.
axlIsSessionReadOnly	This functions determines whether the specified session is opened in read-only or edit mode.
axlIsValidAXLSession	Checks whether the specified ADE XL session is a valid session.
axlMainAppSaveSetup	Saves the ADE state and ADE (G)XL setup database information associated with an ADE (G)XL session to relevant persistent files on disk. This function is useful only in the non-GUI mode.
axlNoSession	Returns `t` if there is no ADE XL session in the current window.
axlRemoveSetupState	Removes the specified setup state for the given session.
axlSaveSetupState	Saves a setup state for the specified session.
axlSessionConnect	Register a SKILL callback to be connected to a known signal or trigger emitted from an ADE (G) XL session.
axlSessionDisconnect	Disconnects the specified SKILL callback connected to one or more known signals emitted by ADE (G) XL session.
axlSessionRegisterCreationCallback	Registers a SKILL function as callback to be called whenever the event for which it is registered is occurred.
axlSessionSignalList	Returns a list of all the signals or triggers that are emitted from a given ADE (G) XL session. You can create custom callback functions to be executed when these events are triggered. For more details, refer to Working with Signals or Triggers.

Session-Related SKILL Functions, *continued*

Function	Description
axlSessionSignalSignature	Returns the signature of a given signal that is emitted by an ADE (G)XL session. This function serves as a utility function to determine how to implement the `slot` or callback function in `SKILL`.
axlSetMainSetupDB	Sets the working setup database for an ADE XL session to the setup database specified by the given setupDBPath. This function is useful when you create a new session in a SKILL script and then you want to setup a database for that.
axlSetMainSetupDBLCV	Sets the working setup database for a given ADE XL session to the setup database specified by the given library, cell, or view.
axlSetupStates	Retrieves a list of setup states from the given session.
axlSuppressPersistedQuestionDialog	Suppresses the question dialog for a specified `msgId` in a Virtuoso session.
axlShowPersistedQuestionDialog	Shows the suppressed question dialog for a specified `msgId` in a Virtuoso session.
axlToolSetOpPointInfo	Adds the signal specified for the `oppoint` type item to the Output Setup table in a test setup and returns the signal object.

axlCloseSession

```
axlCloseSession(
     t_session
     )
     => t / nil
```

Description

Closes the specified ADE XL session.

Argument

t_session	Name of the session you want to close.

Value Returned

`t`	Successful close operation.
`nil`	Unsuccessful close operation.

Example

The following example closes the session `session0`.

```
axlCloseSession( "session0" )
t
```

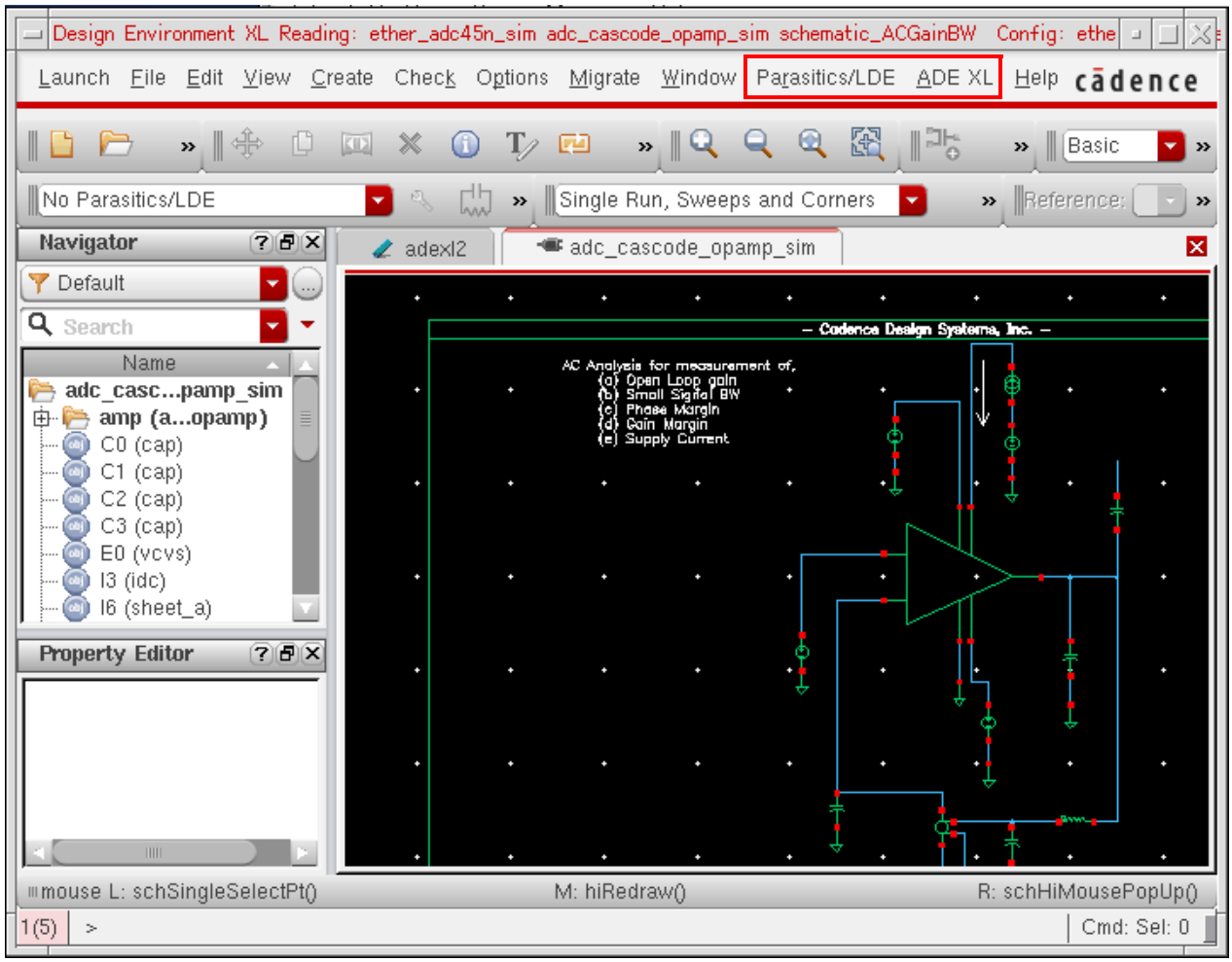
axlCloseSessionInWindow

```
axlCloseSessionInWindow(
      [ w_window ]
      )
      => t / nil
```

Description

Closes the ADE XL session in the current window, if there is one opened.

When you open a schematic or a layout in a new tab in an ADE XL session window, the menus related to ADE XL are also displayed on the schematic or layout tab, as highlighted in the figure shown below.



The axlCloseSessionInWindow function is used to close those assistants and to show only the default menu and assistants of Virtuoso Schematic Editor or Virtuoso Layout Editor.

Arguments

w_window	The window ID of the schematic or layout tab that has associated ADE XL content. If not specified, the ADE XL assistants are closed from the current window.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example closes the ADE XL menus and assistants from the current window.

```
axlCloseSessionInWindow( )
```

axlCreateSession

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

Description

Creates a new ADE XL session with the specified name. You can use this function to create a new session before running an ADE XL SKILL code in the non-GUI mode,

Note: If the ADE XL GUI is already open and you need to execute ADE XL SKILL commands from the CIW, you do not need to create a new ADE XL session. Instead, you can only get a handle to the already open ADE XL session by using the axlGetWindowSession function.

Argument

t_session	Name to be used for the new ADE XL session.

Value Returned

t_sessionName	Returns name of the session, if the session is successfully created.
nil	Unsuccessful operation.

Examples

Example 1:

The following code creates a new ADE XL session with the name `mysession`.

```
s1 = (axlCreateSession "mysession")
"mysession"
```

Example 2:

The following code creates a new ADE XL session with a random number in the session name.

```
sessionName = strcat("mysession" (sprintf nil "%d" random()))
axlSession = axlCreateSession(sessionName)
axlSetMainSetupDBLCV(axlSession "mylib" "mycell" "adexl_view")
```

Exmaple 3:

The following example code shows how to create a new session and add an output expression:

```
sessionName = strcat("mysession" (sprintf nil "%d" random()))
axlSession = axlCreateSession(sessionName)
sdb=axlSetMainSetupDBLCV( axlSession "myLib" "mycell" "adexl")
axlAddOutputExpr(axlSession "mytest" "output1" ?expr "ymax(deriv(VT(\"/OUT\")))")
axlSaveSetup(axlSession)
axlCommitSetupDB( sdb )
axlCloseSetupDB( sdb )
```

Related Functions

If you are running an ADE XL script in the standalone mode, after creating a new session, you can set up a database for the session by using the axlSetMainSetupDBLCV function.

axlGetMainSetupDB

```
axlGetMainSetupDB(
     t_session
     )
     => x_mainSDB / nil
```

Description

Returns a handle to the working setup database of the named ADE XL session.

Argument

t_session	Session name.

Value Returned

x_mainSDB	Handle to the setup database.
nil	Unsuccessful operation.

Examples

The following example shows how to get a handle to the setup database associated with the current ADE XL session:

```
session = axlGetWindowSession()
=> 1001
axlGetMainSetupDB( session )
=> 1002
```

Using this handle to the database, you can now work with various objects of this database. For example, you can create or modify a test, change the values of variables, or create or modify corners.

axlGetSessionCellName

```
axlGetSessionCellName(
     g_value
     )
     => t_cellName / nil
```

Description

Returns the cell name associated with the session or setup database.

Argument

g_value	Name of the ADE (G) XL session or handle to the setup database.

Value Returned

t_cellName	Name of the cell corresponding to the given ADE (G) XL session name or setup database.
nil	Unsuccessful operation.

Examples

You can get the cell name associated with a session by using the session name, as shown below.

```
session = axlGetWindowSession(hiGetCurrentWindow())
cellName = axlGetSessionCellName(session)
"adc_cascode_opamp_sim"
```

You can also get the cell name associated with a session by using the handle to database of that session, as shown below.

```
setupDBId= axlGetMainSetupDB(axlGetWindowSession(hiGetCurrentWindow()))
cellName = axlGetSessionCellName(setupDBId)
"adc_cascode_opamp_sim"
```

Related Functions

axlGetMainSetupDB, axlGetWindowSession

axlGetSessionLibName

```
axlGetSessionLibName(
     g_value
     )
     => t_libName / nil
```

Description

Returns the library name associated with the given session or setup database.

Argument

g_value	Name of the ADE (G) XL session or handle to the setup database.

Value Returned

t_libName	Name of the library corresponding to the given ADE (G) XL session name.
nil	Unsuccessful operation.

Examples

You can get the library name associated with a session by using the session name, as shown below.

```
session = axlGetWindowSession(hiGetCurrentWindow())
libName = axlGetSessionLibName(session
"ether_adc45n_sim"
```

You can get the library name associated with a session by using the handle to database of that session, as shown below:

```
x_mainSDB= axlGetMainSetupDB(axlGetWindowSession(hiGetCurrentWindow()))
libName = axlGetSessionLibName(x_mainSDB)
"ether_adc45n_sim"
```

Related Functions

axlGetMainSetupDB, axlGetWindowSession

axlGetSessionViewName

```
axlGetSessionViewName(
     g_value
     )
     => t_viewName / nil
```

Description

Returns the view name associated with the given session or setup database.

Argument

g_value	Name of the ADE (G) XL session or handle to the setup database.

Value Returned

t_viewName	Name of the view corresponding to the given ADE (G) XL session name.
nil	Unsuccessful operation.

Examples

You can get the view name associated with a session by using the name of that session, as shown below.

```
session = axlGetWindowSession(<adexl-window-id>)
libName = axlGetSessionViewName(session)
"adexl"
```

You can get the view name associated with a session by using the handle to database of that session, as shown below.

```
x_mainSDB= axlGetMainSetupDB(axlGetWindowSession(hiGetCurrentWindow()))
viewName = axlGetSessionViewName(x_mainSDB)
"adexl"
```

Related Functions

axlGetMainSetupDB, axlGetWindowSession

axlGetSessionWindowNumber

```
axlGetSessionWindowNumber(
     t_sessionName
     )
     => x_number / nil
```

Description

Returns a unique integer representing a number corresponding to a given ADE (G) XL session name.

Note: This function is applicable only when ADE (G) XL is opened in the GUI mode. The window number returned by this function is the number displayed in the lower left corner of the ADE XL window.

Argument

t_sessionName	ADE (G) XL session name.

Value Returned

x_number	Unique integer representing a number corresponding to the given ADE (G) XL session name.
nil	Unsuccessful operation.

Example

The following example shows how the ADE XL session ID is returned for the session corresponding to the current window.

```
sessionNum = axlGetSessionWindowNumber(axlGetWindowSession())
1
```

Related Functions

axlGetWindowSession

axlGetToolSession

```
axlGetToolSession(
     t_sessionName
     t_testName
     [ ?history x_history ]
     )
     => g_sessionid / nil
```

Description

ADE XL internally maintains a unique in-memory identifier for each active test. This function returns that unique identifier for the specified test.

You can use the session identifier to directly modify variables for a test in the session. For example, you can modify the temperature value for a test, as shown in the example given below.

Arguments

t_sessionName	ADE (G)XL session name.
t_testName	Test name
?history *x_history*	Integer value representing the history entry. Use this argument when you want to work on the given history run of the test.

Value Returned

g_sessionid	Returns a unique in-memory id.
nil	Unsuccessful operation.

Examples

Example 1:

The following example shows how to add a model library file and section for a test in ADE XL. The script gets access to the test session identifier, `testsess`, by using the axlGetToolSession function. It further uses `testsess` to remove all the model library selections for that test and adds new model file sections.

```
session=axlGetWindowSession()
=> "session1"
x_mainSDB=axlGetMainSetupDB(session)
=> 1001
t1= axlGetTests(x_mainSDB)
=> (1015
("AC" "TRAN")
)
testsess=axlGetToolSession(session "AC")
=> sevSession1
testsess=asiGetSession(testsess)
=> stdobj@0x19827c6c
asiAddModelLibSelection(testsess "models/spectre/gpdk045/gpdk045.scs" "NN")
=> t
```

Example 2:

The following example shows how you can use the history for a particular test to get access to the unique identifier for a test in that history. You can use this identifier to further work with the OASIS session.

```
axlsession=axlGetWindowSession( hiGetCurrentWindow())
=> "session0"
testnames = (cadr (axlGetTests (axlGetMainSetupDB axlsession)))
=> ("ACGainBW" "PSR" "SlewRate" "CMRR" "Offset")
x_mainSDB=axlGetMainSetupDB(axlsession)
=> 2808
historyNames = (axlGetHistory x_mainSDB)
=> (2852
("Interactive.1" "Interactive.2" "Interactive.3" "Interactive.4" "Interactive.5")
)
firsthistory = axlGetHistoryEntry(x_mainSDB caadr(historyNames))
=> 2853
(axlGetToolSession(axlsession (car testnames) ?history firsthistory)
=> sevSession1
```

Note: To know more about the SKILL functions to be used while working with an OASIS object, refer to Virtuoso Analog Design Environment L SKILL Reference.

axlGetWindowSession

```
axlGetWindowSession(
     [ w_window ]
     )
     => t_sessionName / nil
```

Description

Returns the ADE XL session associated with a window.

There is a session object associated with each ADE XL instantiation. If you have an ADE XL session open, you can retrieve that session by using this function.

Note: If you are working in the non-GUI mode, you need to create a new session by using the axlCreateSession function and then associate a setup database with it.

Argument

w_window	Window ID

Value Returned

t_sessionName	Returns the session name.
nil	Unsuccessful operation.

Examples

You can use any one of the following two examples to return the session of the active ADE XL window:

Example 1

```
axlGetWindowSession()
=> "session0"
```

Example 2

```
axlGetWindowSession( hiGetCurrentWindow())
=>"session0"
```

axlGetCurrentResultSimulationHost

```
axlGetCurrentResultSimulationHost(
     t_sessionName
     )
     => t_hostName
```

Description

This is a callback function that runs from a *Results* table context menu.

It returns the name of the host where the simulation was performed for the currently selected test and data point in the *Results* table.

Argument

t_sessionName	Specifies the name of the ADE XL session from which the simulation was run.

Value Returned

t_hostName	Returns the host name corresponding to the specified ADE XL session.

Example

The following example shows how you can use the axlGetCurrentResultSimulationHost function to print the simulation host name.

```
;Define custom menu function to print the simulation host of the selected result
(defun DEMOprintSimulationHost(adeSession)
   (let ()
   (printf "Selected History = %s\n" (axlGetHistoryName (axlGetCurrentHistory
   adeSession->axlsession)))
   (printf "Selected Test = %L\n" adeSession->axlTestName)
   ;Get the selected point ID indirectly from the data dir
   (printf "Selected Point ID = %L\n" (cadr (reverse (parseString
   (adeSession->axlCurrentDataDir) "/"))))
   (printf "\tHostname = %s\n" (axlGetCurrentResultSimulationHost adeSession))
   (printf "\tData dir = %s\n" adeSession->axlCurrentDataDir)
   )
)

adeSession=axlGetWindowSession( hiGetCurrentWindow())
```

```
(sprintf nil "DEMOprintSimulationHost('%L)" adeSession )
```

axlIsSessionReadOnly

```
axlIsSessionReadOnly(
     t_axlSession
     )
     => t / nil
```

Description

This functions determines whether the specified session is opened in read-only or edit mode.

Argument

t_axlsession	Specifies the name of the session.

Value Returned

`t`	The specified session is opened in the read-only mode.
`nil`	The specified session is opened in the edit mode.

Example

The following example shows that `session0` is opened in the read-only mode.

```
axlIsSessionReadOnly("session0")
=> t
```

axlIsValidAXLSession

```
axlIsValidAXLSession(
     t_session
     )
     => t / nil
```

Description

Checks whether the specified ADE XL session is a valid session.

Argument

t_session	Name of the ADE XL session to be validated.

Value Returned

`t`	The specified ADE XL session exists.
`nil`	The specified ADE XL session does not exist.

Example

The following example shows that the specified session, `session1`, is valid.

```
session=axlGetWindowSession()
=> "session1"
axlIsValidAXLSession(session)
=> t
```

axlMainAppSaveSetup

```
axlMainAppSaveSetup(
     [ t_sessionName ]
     )
     => t / nil
```

Description

Saves the ADE state and ADE (G)XL setup database information associated with an ADE (G)XL session to relevant persistent files on disk. This function is useful only in the non-GUI mode.

For example, if you modify the design variables or corners in ADE XL, you can save the changes in the setup by using this function.

Arguments

t_sessionName	ADE (G)XL session name.

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Example

The following example creates a new session, adds a new variable, and saves the details from the currently active session to the disk.

```
sessionName=strcat("mysession" (sprintf nil "%d" random()))
axlSession=axlCreateSession(sessionName)
x_mainSDB=axlSetMainSetupDBLCV(axlSession car(LibraryName) cadr(LibraryName)
ViewName)
axlPutVar(x_mainSDB "varname" "varvalue")
axlMainAppSaveSetup(axlSession)
axlCloseSession(axlSession)
```

axlNoSession

```
axlNoSession(
     [ w_window ]
     )
     => t / nil
```

Description

Returns `t` if there is no ADE XL session in the current window.

Arguments

w_window	Window name.

Value Returned

`t`	There is no ADE XL session in the current window.
`nil`	There is an ADE XL session in the current window.

Example

The following function specifies if there is any active ADE XL session in the current window.

```
axlNoSession( )
t
```

axlRemoveSetupState

```
axlRemoveSetupState(
     t_sessionName
     t_stateName
     )
     => t / nil
```

Description

Removes the specified setup state for the given session.

Arguments

t_sessionName	Session name.
t_stateName	State name.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example removes the `state1` state that is saved for session `session1`:

```
session = (axlGetWindowSession)
"session1"
(axlSetupStates session)
("state1")

(axlRemoveSetupState session "state1")
t
```

axlSaveSetupState

```
axlSaveSetupState(
     t_session
     t_stateName
     l_tags
     [ ?inReadOnly readOnlyAction ]
     )
     => t / nil
```

Description

Saves a setup state for the specified session.

Argument

t_session	Session name.
t_stateName	Name to be used for the saved state.
l_tags	List of tags to be saved with the state. Available tags are: tests - Testbench setups vars - Global variables parameters - Parameters and their values currentMode - Run mode runOptions - Simulation options for different run modes and the run distribute options specs - Parameter specifications corners - Corner details modelGroups - Model groups extensions - Extensions relxanalysis - Reliability analysis setup details All - Details of all tests, vars, parameters, currentMode, runOptions, specs, corners, modelGroups, extensions, and relxanalysis.
`?inReadOnly` *readOnlyAction*	Specifies the action to be performed in read only mode. Possible values: `'error`: Displays an error. This is the default value. `'useSaveDir`: Saves the setup state in the save directory. `'useprojectDir`: Saves the setup state in the project directory.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

Example 1:

The following example shows how you can save corner changes from a session to a saved state:

```
session = (axlGetWindowSession)
=>"session1"
; returns handle to the current ADE XL session

x_mainSDB=axlGetMainSetupDB(session)
=>1001
; returns handle to the setup database of the session

c1 = axlPutCorner( x_mainSDB "c1" )
=>1235
; adds a new corner c1

axlPutVar( c1 "VIN_CM" "1.06 1.08" )
=>1237
; Sets corner variables

axlSaveSetupState(session "CornersState1" `("corners" "vars"))
=>t
; saves the corner details in a setup state named CornersState1
```

Example 2:

To save all the tags from the current session to a state, use the `all` tag as shown in the following example:

```
session = (axlGetWindowSession)
=>"session1"
x_mainSDB=axlGetMainSetupDB(session)
=>1001
axlSaveSetupState(session "state2" "All")
=>t
```

axlSessionConnect

```
axlSessionConnect(
     t_sessionName
     t_signalName
     s_callbackFunction
     )
     => t / nil
```

Description

Register a SKILL callback to be connected to a known signal or trigger emitted from an ADE (G) XL session.

Arguments

t_sessionName	ADE (G) XL session name.
t_signalName	Name of the signal or trigger emitted by the ADE (G)XL session for which to register a callback. To see the list of signals emitted by ADE (G)XL sessions, see axlSessionSignalList.
s_callbackFunction	Symbol representing the callback function to be called when the signal is emitted.

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

The following example shows how to connect a custom function, `_myRunModeChangedCB`, with the runModeChanged trigger.

```
session = axlGetWindowSession(hiGetCurrentWindow())
(axlSessionConnect session "runModeChanged" '_myRunModeChangedCB)
```

Whenever the simulation run mode is changed, the SKILL function `_myRunModeChangedCB` will be called.

Note: It is important to register the callback during the ADE XL session start to connect the trigger with the custom function. For this, you can use the axlSessionRegisterCreationCallback function in the .cdsinit file.

For more examples, refer to Working with Signals or Triggers.

Related Functions

axlSessionRegisterCreationCallback, axlSessionSignalList, axlSessionSignalSignature axlSessionDisconnect

axlSessionDisconnect

```
axlSessionDisconnect(
     t_sessionName
     s_callbackFunction
     )
     => t / nil
```

Description

Disconnects the specified SKILL callback connected to one or more known signals emitted by ADE (G) XL session.

Arguments

t_sessionName	ADE (G) XL session name.
s_callbackFunction	Symbol representing the callback function to be disconnected.

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

The following example disconnects the `myFunc` callback from the attached signal or trigger:

```
session = axlGetWindowSession(hiGetCurrentWindow())
axlSessionDisconnect(session 'myFunc)
```

For more examples, refer to Working with Signals or Triggers.

Related Functions

axlSessionConnect, axlSessionRegisterCreationCallback, axlSessionSignalList, axlSessionSignalSignature

axlSessionRegisterCreationCallback

```
axlSessionRegisterCreationCallback(
     s_callbackFunction
     )
     => t / nil
```

Description

Registers a SKILL function as callback to be called whenever the event for which it is registered is occurred.

Arguments

s_callbackFunction	SKILL symbol representing the callback function to be called upon creation of a new ADE (G) XL session.

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

For example, refer to Example 1: To automatically disable corners when ADE XL is launched.

Related Functions

axlSessionConnect, axlSessionDisconnect, axlSessionSignalList, axlSessionSignalSignature

axlSessionSignalList

```
axlSessionSignalList(
     t_session
     )
     => l_signals / nil
```

Description

Returns a list of all the signals or triggers that are emitted from a given ADE (G) XL session. You can create custom callback functions to be executed when these events are triggered. For more details, refer to Working with Signals or Triggers.

Arguments

t_session	Name of the ADE (G) XL session.

Value Returned

l_signals	List of signals that are returned from ADE (G) XL. For more information, see Table 1-1 on page 53.
nil	Unsuccessful registration.

Table 1-1 Available signals

Signal	Change in State
cornersUpdated	When corners are updated **Syntax** `cornersUpdated(t_sessionName)`
createdTest	When a test is created **Syntax:** `createdTest(t_sessionName t_testName)`

Table 1-1 Available signals

Signal	Change in State
eligibleReferenceHistoryItemsChanged	When there is a change in the list of history items that are eligible to be used as reference history items **Syntax:** `eligibleReferenceHistoryItemsChanged(t_sessionName)`
initializedTest	When a test is enabled **Syntax:** `initializedTest(t_sessionName t_testName)`
ocnPostRunCommandWrite	After writing the ocnxlRun command in the OCEAN script being written from ADE XL. **Syntax:** `ocnPostRunCommandWrite(t_sessionName g_filePointer)`
ocnPreRunCommandWrite	Before writing the ocnxlRun command in the OCEAN script being written from ADE XL. **Syntax:** `ocnPreRunCommandWrite(t_sessionName g_filePointer)`
parametersUpdated	When the values of parameters are updated **Syntax:** `parametersUpdated(t_sessionName)`
pointSimulationCompleted	When simulation for a point is completed **Syntax:** `pointSimulationCompleted(t_sessionName x_historyHSDB t_testName x_pointId)`
postCloseCellView	After the ADE XL cellview is closed **Syntax:** `postCloseCellView(t_sessionName t_lib t_cell t_view t_mode )`

Table 1-1 Available signals

Signal	Change in State
postCreateDatasheet	After a datasheet is created **Syntax:** `postCreateDatasheet(t_sessionName x_hsdb t_datasheetDir)`
postCreateHistoryEntry	After a history item is created **Syntax:** `postCreateHistoryEntry(t_sessionName x_hsdb)`
postCreateMainSetupDB	After a setup database is created **Syntax:** `postCreateMainSetupDB(t_sessionName x_hsdb)`
postCreatedTest	After a test is created **Syntax:** `postCreatedTest(t_sessionName t_testName)`

Table 1-1 Available signals

Signal	Change in State
postExportResults	After exporting the simulation result data in the HTML or CSV format **Syntax:** `postExportResults(t_sessionName x_historyHSDB t_exportedType t_exportedFilePath x_exportSuccessful)` where, `t_exportedType` specifies the source names from where you are exporting results. For example, a Results view or other specification analysis views. Possible values that this argument can take are listed below: ■ `Detail` ■ `Detail - Transpose` ■ `Optimization` ■ `Summary` ■ `Yield` ■ `Spec Summary` ■ `Spec Comparison` ■ `Spec Comparison View`
postImportSetup	After importing the simulation setup from an existing ADE XL view to the current ADE XL view **Syntax:** `postImportSetup(t_sessionName t_newSetupPath l_importTags t_historyName t_operation)`
postInstall	After opening the ADE XL GUI **Syntax:** `postInstall(t_sessionName)`
postInstallSchematic	After opening the schematic in ADE XL **Syntax:** `postInstallSchematic(t_sessionName)`

Table 1-1 Available signals

Signal	Change in State
postLoadMainSetupDB	After a setup database is loaded **Syntax:** `postLoadMainSetupDB(t_sessionName x_hsdb)`
postLoadSetupState	After a setup state is loaded **Syntax:** `postLoadSetupState(t_sessionName t_stateName)`
postRestoreHistory	After a history item is restored **Syntax:** `postRestoreHistory(t_sessionName x_hsdb)`
postSaveSetupState	After a setup state is saved **Syntax:** `postSaveSetupState(t_sessionName t_stateName l_saveTags)`
postSaveSimulationResults	After saving the simulation results for a history to the given destination directory **Syntax:** `postSaveSimulationResults(t_sessionName x_sdbHandle t_historyName t_destinationDir x_copyPSFResults)` where, *`x_copyPSFResults`* takes the boolean value specified for the *Copy PSF Results* option on the **Save Results** form, which opens when you save results for a history.
postViewHistoryResults	After the results for a history item are displayed **Syntax:** `postViewHistoryResults(t_sessionName x_historyHSDB t_resultsDBPath)`
preCreateDatasheet	Before a datasheet is created **Syntax:** `preCreateDatasheet(t_sessionName x_hsdb t_datasheetDir)`

Table 1-1 Available signals

Signal	Change in State
preCreateHistoryEntry	Before a history item is created **Syntax:** `preCreateHistoryEntry(t_sessionName x_hsdb)`
preDestroySession	Before closing anADE (G) XL session **Syntax:** `preDestroySession(t_sessionName)`
preExportResults	Before exporting the results data in the HTML or CSV format **Syntax:** `preExportResults(t_sessionName x_historyHSDB t_exportedType t_exportedFilePath)` where, `t_exportedType` specifies the name of the user interface from where you are exporting results. For example, a Results view or other specification analysis views. Possible values that this argument can take are listed below: ■ `Detail` ■ `Detail - Transpose` ■ `Optimization` ■ `Summary` ■ `Yield` ■ `Spec Summary` ■ `Spec Comparison` ■ `Spec Comparison View`
preImportSetup	Before a setup is imported **Syntax:** `preImportSetup(t_sessionName t_newSetupPath l_importTags t_historyName t_operation)` For the list of available tags to be used for l_importTags, refer to axlImportSetup.

Table 1-1 Available signals

Signal	Change in State
preInstallCellView	Before opening an ADE XL view **Syntax:** `preInstallCellView(t_sessionName t_lib t_cell t_view t_mode )`
preLoadSetupState	Before a setup state is loaded **Syntax:** `preLoadSetupState(t_sessionName t_stateName)`
preRemoveTest	Before a test is deleted **Syntax:** `preRemoveTest(t_sessionName t_testName)`
preRestoreHistory	Before a history item is restored **Syntax:** `preRestoreHistory(t_sessionName x_hsdb)`
preRun	Before a simulation run is started **Syntax:** `preRun(t_session x_setupdb t_mode t_testName)`
preSaveSetupState	Before a setup state is saved **Syntax:** `preSaveSetupState(t_sessionName t_stateName l_saveTags)` For the list of available tags to be used for `l_saveTags`, refer to axlSaveSetupState.
preSaveSimulationResults	Before saving the simulation results **Syntax:** `preSaveSimulationResults(t_sessionName x_sdbHandle t_historyName t_destinationDir x_copyPSFResults)` where, *`x_copyPSFResults`* takes the boolean value specified for the *Copy PSF Results* option on the **Save Results** form, which opens when you save results for a history.

Table 1-1 Available signals

Signal	Change in State
preViewHistoryResults	Before the results for a history item is viewed **Syntax:** `preViewHistoryResults(t_sessionName x_historyHSDB t_resultsDBPath)`
referenceHistoryItemChanged	When the reference history item is changed **Syntax:** `referenceHistoryItemChanged(t_name)`
removedHistoryEntry	When a history item is deleted **Syntax:** `removedHistoryEntry(t_sessionName x_hsdb)`
removedTest	When a test is deleted **Syntax:** `removedTest(t_sessionName t_testName)`
renamedHistoryEntry	When a history item is renamed **Syntax:** `renamedHistoryEntry(t_sessionName x_hsdb t_oldName)`
renamedTest	When a test is renamed **Syntax:** `renamedTest(t_sessionName t_originalName t_newName)`

Table 1-1 Available signals

Signal	Change in State
runFinished	When a simulation run is completed **Syntax:** `runFinished(t_sessionName x_runId x_runhsdb x_errorCode)` where, `t_sessionName` is the name of the active session. `x_runId` is the unique ID associated with the run. `x_runhsdb` is the handle to the database of the run. `x_errorCode` has not been implemented yet. It is always `nil`.
runFinishedConclusion	After all the simulations are run in ADE GXL. **Note:** An optimization simulation run in ADE GXL might run a collection of various runs internally. For example, the High Yield Estimation runs various Monte Carlo simulations. After each Monte Carlo run, the `runFinished` signal is emitted. However, after all the required simulations are complete, the `runFinishedConclusion` signal is emitted. **Syntax:** `runFinishedConclusion(t_sessionName x_runId x_runhsdb)`
runFinishedPostPlot	After plotting the results of a simulation. **Syntax:** `runFinishedPostPlot(t_sessionName x_runId x_runhsdb)`
runFinishedPrePlot	Before plotting the results of a simulation **Syntax:** `runFinishedPrePlot(t_sessionName x_runId x_runhsdb)`

Table 1-1 Available signals

Signal	Change in State
runModeChanged	When the simulation run mode is changed **Syntax:** `runModeChanged(t_sessionName t_newRunMode)`
runOptionsUpdated	When the run options are updated **Syntax:** `runOptionsUpdated(t_sessionName)`
runPaused	When a simulation run is stopped **Syntax:** `runPaused(t_sessionName x_runId x_isPaused)`
runProgress	When simulation for a design point is complete. **Syntax:** `runProgress(t_sessionName x_runid x_numFinished x_numSubmitted)`
runStarted	When a simulation run is started. **Syntax:** `runStarted(t_sessionName x_runId x_runhsdb)`
setSessionSetupDB	After the creation of a session. **Syntax:** `setSessionSetupDB(t_sessionName x_hsdb)` If you want to perform a custom action after the session and setup database is created, it is recommended to register a callback for `postCreateSessionDB` instead of `setSessionSetupDB`.

Table 1-1 Available signals

Signal	Change in State
setupSaved	This trigger is called in the following cases: ■ When a `maestro` view is saved using the *Session - Save* menu command ■ When an `adexl` view is saved using the *File - Save* menu command ■ When a simulation run starts ■ When the setup is saved while exiting the *maestro* or *adexl* cellview **Note:** This trigger works only in the write mode. **Syntax:** `setupSaved(t_sessionName)`
startedNewManualTuningRun	When the manual tuning run is started **Syntax:** `startedNewManualTuningRun(t_sessionName x_runId x_runhsdb)`
testStatusUpdated	After a test is enabled or disabled in ADE XL **Syntax:** `testStatusUpdated(t_sessionName t_testName, x_enabled)`
updatedSetupStates	After the list of setup states is changed **Syntax:** `updatedSetupStates(t_sessionName)`

Table 1-1 Available signals

Signal	Change in State
updatedTest	After any changes are made to the test. For example, a change in the test output, variable value, analysis, temperature, model file, or simulation file. The event is not triggered when there is a change in the status of an analysis or design variable, or when the state or active setup is saved. **Syntax:** `updatedTest(t_sessionName t_testName)`
useIncrementalChanged	When the options for incremental simulation are modified **Syntax:** `useIncrementalChanged()`
variablesUpdated	When variable values are updated **Syntax:** `variablesUpdated(t_sessionName)`

Example

The following example shows how you can see the list of available signals:

```
session = axlGetWindowSession(hiGetCurrentWindow())
axlsignals = axlSessionSignalList(session)
```

To see examples on how to execute a callback when a signal is emitted, refer to Working with Signals or Triggers.

Related Functions

axlSessionConnect, axlSessionRegisterCreationCallback, axlSessionSignalSignature, axlSessionDisconnect

axlSessionSignalSignature

```
axlSessionSignalSignature(
     t_session
     t_signal
     )
     => t_signature / nil
```

Description

Returns the signature of a given signal that is emitted by an ADE (G)XL session. This function serves as a utility function to determine how to implement the `slot` or callback function in `SKILL`.

Arguments

t_session	String representing the ADE (G) XL session.
t_signal	Name of a known signal that is emitted by ADE (G) XL session.

Value Returned

`t_signature`	Returns the signature of the given signal.
`nil`	Unsuccessful registration.

Example

The following example shows how to use the axlSessionSignalSignaure function to find out the signature of the runModeChanged signal:

```
session = axlGetWindowSession(hiGetCurrentWindow())
signature = axlSessionSignalSignature(session "runModeChanged")
signature => "runModeChanged(QString)"
```

To see examples on how to execute a callback when a signal is emitted, refer to Working with Signals or Triggers.

Related Functions

axlSessionConnect, axlSessionRegisterCreationCallback, axlSessionSignalList, axlSessionSignalList

axlSetMainSetupDB

```
axlSetMainSetupDB(
     t_session
     t_setupdbPath
     )
     => x_hsdb / nil
```

Description

Sets the working setup database for an ADE XL session to the setup database specified by the given setupDBPath. This function is useful when you create a new session in a SKILL script and then you want to setup a database for that.

Arguments

t_session	Session name.
t_setupdbPath	Path to a setup database file located in the ADE XL view. The setup database is typically named as `data.sdb`.

Value Returned

x_hsdb	Setup database handle.
`nil`	Unsuccessful operation.

Example

The following example creates a new session, `data_session`, and then sets a new database for that.

```
data_session = ( axlCreateSession "data_session" )
x_mainSDB = axlSetMainSetupDB( "data_session" "data.sdb" )
4001
```

Reference

axlCreateSession

axlSetMainSetupDBLCV

```
axlSetMainSetupDBLCV(
     t_session
     t_libName
     t_cellName
     t_viewName
     [ ?mode t_mode ]
     )
     => x_mainSDB / nil
```

Description

Sets the working setup database for a given ADE XL session to the setup database specified by the given library, cell, or view.

This function is useful in pointing to an existing database after you create a new ADE XL session.

Arguments

t_session	Session name.
t_libName	Library name
t_cellName	Cell Name
t_viewName	View Name
?mode *t_mode*	Access mode Valid values: ■ "a" for append mode. This is the default mode. ■ "r" for read-only mode

Value Returned

x_mainSDB	Handle to the setup database
nil	Unsuccessful operation

Example

The following example creates a new ADE XL session, `session0`, and sets up a database to the existing database for the adexl view of the `myCell` cell in the `myLib` library.

```
axlCreateSession("session0")
x_mainSDB=axlSetMainSetupDBLCV("session0" "myLib" "myCell" "adexl" ?mode "r")
```

Note: Only read-only access has been provided to the setup DB.

axlSetupStates

```
axlSetupStates(
     t_session
     )
     => l_states
```

Description

Retrieves a list of setup states from the given session.

Arguments

t_session	Session Name.

Value Returned

l_states	List of states.

Example

The following example gets a list of existing states for the current session and then uses one of the states to modify corners.

```
session = (axlGetWindowSession)
"session1"

x_mainSDB=axlGetMainSetupDB(session)
1001
axlSetupStates(session)
("state1" "tCorners_state")
; load one state
axlLoadSetupState(session "tCorners_state" "All" 'overwrite)
t

; add a corner
c1 = axlPutCorner( x_mainSDB "c1" )
3841

; add variables to the corner
axlPutVar( c1 "VDD" "1.06 1.08 2.1" )
3843
axlPutVar( c1 "temp" "-40 0 75" )
3845

; save the state
axlSaveSetupState(session "tCorners_state" `("corners"))
t
```

axlSuppressPersistedQuestionDialog

```
axlSuppressPersistedQuestionDialog(
     x_msgId
     )
     => t / nil
```

Description

Suppresses the question dialog for a specified *msgId* in a Virtuoso session.

Note: This function works only for the message boxes that contain the *Do not show this dialog again* check box.

Arguments

x_msgId	Id of the message to be suppressed.

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Example

The following example suppresses the question dialog for *msgId* 1234 in the current Virtuoso session:

```
axlSuppressPersistedQuestionDialog(1234)
=> t
```

The question dialog will not be displayed for *msgId* 1234.

axlShowPersistedQuestionDialog

```
axlShowPersistedQuestionDialog(
     x_msgId
     )
     => t / nil
```

Description

Shows the suppressed question dialog for a specified *msgId* in a Virtuoso session.

Note: This function works only for the message boxes that contain the *Do not show this dialog again* check box.

Arguments

x_msgId	Id of the message to be shown.

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Example

The following example shows the question dialog for *msgId* 1234 in the current Virtuoso session:

```
axlShowPersistedQuestionDialog(1234)
=> t
```

The question dialog be shown for *msgId* 1234.

axlToolSetOpPointInfo

```
axlToolSetOpPointInfo(
     g_sessionId
     t_testName
     [ ?instanceName t_instanceName ]
     [ ?parameters t_parameters ]
     )
     => o_sevOpPoint / nil
```

Description

Adds the signal specified for the `oppoint` type item to the Output Setup table in a test setup and returns the signal object.

Arguments

g_sessionId	Id of the current session.
t_testName	Name of the test.
?instanceName *t_instanceName*	Name of the instance.
?parameters *t_parameters*	Operating point parameters related to the specified instance.

Value Returned

o_sevOpPoint	Returns the signal object when adding the signal output to the test successfully.
`nil`	Returns `nil` if the operation is unsuccessful.

Example

The following example shows how to add an `oppoint` type signal to the output setup:

```
sessionId=axlGetWindowSession(hiGetCurrentWindow)
testName="myoalib:ampTest:1"
axlToolSetOpPointInfo(sessionId testName ?instanceName "/R2" ?parameters "res")
=>sevOpPointStruct@0x2f207140
```

Working with Signals or Triggers

The ADE (G)XL session functions are a known set of `states`. Any transition from one state to another is called an `event`. You can specify customized actions to be automatically performed whenever an event occurs. You can do this by registering callbacks for these events or signals. In Trolltech QT's terminology, these events are known as `signals` and the callbacks are known as `slots`.

To execute callbacks or slots for ADE XL signals, you need to perform the following tasks:

1. Define a callback function

   Define any SKILL function that you need to call when an event occurs in ADE G(XL). It is recommended to define this procedure in the `.cdsinit` file.

   Note: You can use any other ADE XL SKILL function in this callback function.

2. Connect the defined callback function with the signal

   Connect the custom SKILL function defined in step 1 with the required signal or event by using the axlSessionConnect SKILL function in the `.cdsinit` file.

3. Register the callback when a ADE (G)XL session is launched

   To make the callback function available for calling, it is required to register the function by using the `axlSessionRegisterCreationCallback` SKILL function in the `.cdsinit` file.

Tip

To know the signature of a trigger, you can use the axlSessionSignalSignature SKILL function.

The following examples shows how you can register callbacks and call them at runtime from an ADE XL session:

- Example 1: To automatically disable corners when ADE XL is launched
- Example 2: To automatically exit Virtuoso or ADE (G)XL when after the run has finished.
- Example 3: To send a mail after a run is finished
- Example 4: To automatically print the job policy settings to the CDS.log when an ADE XL run starts
- Example 5: To automatically change the ADE XL job policy configuration based on the simulation setup

- Example6: To automatically perform a task after the setup is saved
- Example 7: To automatically add a variable when an ADE test is created

Example 1: To automatically disable corners when ADE XL is launched

1. Define the following function in the .cdsinit file.

```
(defun LJNpostInstall_disableCorners (session)
    (let (sdb)
        sdb = (axlGetMainSetupDB session) ; ADE XL setup DB handle
        (axlSetAllCornersEnabled sdb nil)
    )
)
```

2. Connect ADE XL triggers in .cdsinit.

```
(defun LJNConnectADEXLTriggers_disableCorners (session_name)
    (axlSessionConnect session_name "postInstall" 'LJNpostInstall_disableCorners)
)
```

3. Register a callback to connect the triggers on ADE XL session start

```
(axlSessionRegisterCreationCallback 'LJNConnectADEXLTriggers_disableCorners)
```

Example 2: To automatically exit Virtuoso or ADE (G)XL when after the run has finished.

1. Register a callback to connect the triggers on ADE XL session start.

```
(axlSessionRegisterCreationCallback 'LJNConnectADEXLTriggers_ExitRunFinished)
```

2. Connect ADE XL triggers.

```
(defun LJNConnectADEXLTriggers_ExitRunFinished (session_name)
    (let ()

        ;pop-up the purge data dialog when the user hits the Run button
        ;the user should save all data and stop any interactive work within the
        ;virtuoso session to allow for auto-exit
        ;select the All button, select save, select OK
        (axlSessionConnect session_name "preRun" 'LJNSaveDataPreRun)

        ;uncomment one of the axlSessionConnect lines to either
        ;exit virtuoso after the run has finished
        (axlSessionConnect session_name "runFinishedConclusion"
'LJNExitVirtuosoADEXLRunFinishedConclusion)

        ;or exit just ADE (G)XL after the run has finished
      ;(axlSessionConnect session_name "runFinishedConclusion"
'LJNExitADEXLRunFinishedConclusion)
)
```

```
)
```

3. Give the user an opportunity to save design data when the ADE XL run has started. If any data has been modified - this will prevent auto-exit at the end of the run.

```
(defun LJNSaveDataPreRun (session sdb mode testname)
     (let ( )
        (ddsHiCloseData)
     )
)
```

Example 3: To send a mail after a run is finished

The following example shows how you can connect a procedure with the runFinished trigger to send an e-mail on completion of an ADE XL run.

1. register a callback to connect the triggers on ADE XL session start.

```
(axlSessionRegisterCreationCallback 'LJNaxlConnectADEXLTriggers_emailRunFinished)

;connect ADE XL triggers
(defun LJNaxlConnectADEXLTriggers_emailRunFinished (session_name)
    (axlSessionConnect session_name "runFinished" 'LJNaxlRunFinished_email)
)
```

2. Define a handler to connect to a trigger

```
(defun LJNaxlRunFinished_email (session runId runsdb errorCode)
    (let (history libName cellName viewName message command email)

    email = "user@cadence.com"

    ;send email with subject e.g. "ADE XL run finished Interactive.0 <libname>
<cellname> <viewname>"
    history = (axlGetHistoryName (axlGetCurrentHistory session))
    libName = (axlGetSessionLibName session)
    cellName = (axlGetSessionCellName session)
    viewName = (axlGetSessionViewName session)

message = (strcat "ADE XL run finished " history " " libName " " cellName " "
viewName)

    ;mutt email command (no body) e.g.:  mutt -s 'the subject' user@address.com <
/dev/null
    command = (strcat "mutt -s '" message "' " email " < /dev/null")
    (system  command)

    ;Print message to CIW
    (printf "\n%s\n" message)
    )
)
```

Example 4: To automatically print the job policy settings to the CDS.log when an ADE XL run starts

1. Register a callback to connect the triggers on ADE XL session start.

```
(axlSessionRegisterCreationCallback 'LJNConnectADEXLTriggers_printJobPolicyInfo)
(defun LJNConnectADEXLTriggers_printJobPolicyInfo (sessionName)
    (let ()
        ;When an ADE (G)XL  run starts print the job policy settings to the CDS.log
        (axlSessionConnect sessionName "runStarted"
'LJNrunStarted_printJobPolicyToLog)
    )
)


(defun LJNrunStarted_printJobPolicyToLog (session runID histEntry)
    ;Print run info and job policy settings to CDS.log
    (let (checkPoint)
        checkPoint = (axlGetHistoryCheckpoint histEntry)
        (printf "Run started %s \n" (getCurrentTime))
        (printf "History Name =  %s\n" (axlGetHistoryName histEntry))
        (printf "Run Mode =  %s\n" (axlGetCurrentRunMode checkPoint))
        (LJNaxlPrintJobPolicyInfoToLog session)
    )
)


(defun LJNaxlPrintJobPolicyInfoToLog (session)
    ;Print job policy settings to CDS.log
    (let (jp)
        jp = (axlGetAttachedJobPolicy session "ICRP")
        ;(printf "Job policy =  %L\n" jp)
        (LJNaxlGetJobPolicyInfoPrintParam "Job policy name " jp->name)
        (when jp->distributionmethod == "Command"
            (LJNaxlGetJobPolicyInfoPrintParam "command" jp->jobsubmitcommand)
        )
        (LJNaxlGetJobPolicyInfoPrintParam "\tmax jobs" jp->maxjobs)
        (LJNaxlGetJobPolicyInfoPrintParam "\tstart immediately"
jp->preemptivestart)
        (LJNaxlGetJobPolicyInfoPrintParam "\tstart timeout" jp->starttimeout)
        (LJNaxlGetJobPolicyInfoPrintParam "\tconfigure timeout"
jp->configuretimeout)
        (LJNaxlGetJobPolicyInfoPrintParam "\trun timeout" jp->runtimeout)
        (LJNaxlGetJobPolicyInfoPrintParam "\tlinger timeout" jp->lingertimeout)
        (LJNaxlGetJobPolicyInfoPrintParam "\tshow output log on error"
jp->showoutputlogerror)
        (LJNaxlGetJobPolicyInfoPrintParam "\tshow errors even if retrying test"
jp->showerrorwhenretrying)
        (LJNaxlGetJobPolicyInfoPrintParam "\treassign immediately for new run"
jp->reconfigureimmediately)
        (LJNaxlGetJobPolicyInfoPrintParam "\tblock email" jp->blockemail)
    )
)

(defun LJNaxlGetJobPolicyInfoPrintParam (str param)
    (when (stringp param) (printf "%s = %s\n" str param))
)

(defun LJNaxlPrintJobPolicyInfo ()
;for interactive usage
```

```
    (LJNaxlPrintJobPolicyInfoToLog (axlGetWindowSession))
)
```

Example 5: To automatically change the ADE XL job policy configuration based on the simulation setup

1. Register a callback to connect the triggers on ADE XL session start

```
(axlSessionRegisterCreationCallback 'LJNConnectADEXLTriggers_JobPolicyPerTest)
```

2. Connect ADE XL triggers

```
(defun LJNConnectADEXLTriggers_JobPolicyPerTest (session_name)
    (let ()
        ;Pre run to modify the job policy of each test on run
        (axlSessionConnect session_name "preRun"
'LJNpreRun_checkTestAndApplyJobPolicy)
    )
)


(defun LJNpreRun_checkTestAndApplyJobPolicy (session sdb mode testName)
    (let (tests)
        ;when statement is for SKILL Lint to use args mode testName
        (when (and mode testName)
            tests = (cadr (axlGetTests sdb) )
            (foreach test tests
                (LJNcheckTestAndApplyJobPolicy session test)
            )
        )
    )
)

(defun LJNcheckTestAndApplyJobPolicy (session testName)
   ;A different job policy is applied to the ADE XL test based on the Simulation
Performance Mode.
   ;This could be extended to check other simulation options.
   (let (testSession oSession mode jobPolicy)
        testSession = (axlGetToolSession session testName) ;sev session
        oSession = (sevEnvironment testSession) ;oasis session
        ;simulator = (sevSimulator testSession)
        mode = asiGetHighPerformanceOptionVal(oSession 'uniMode)

        (case mode
           ("Spectre" jobPolicy=(axlGetAttachedJobPolicy session "ICRP" testName)
                ;uncomment the printf to see what you can modify
                ;(printf "job policy = %L\n" jobPolicy)
                ;example that sets the job policy method to a LSF command string
                jobPolicy->distributionmethod = "Command"
                jobPolicy->jobsubmitcommand = "bsub -R \"OSREL==EE50\" -Is -J
SPECTRE"
                jobPolicy->name = "LSF Spectre"
                (axlAttachJobPolicy session jobPolicy->name "ICRP" jobPolicy
testName)
            )
            ("APS" jobPolicy=(axlGetAttachedJobPolicy session "ICRP" testName)
```

```
                ;uncomment the printf to see what you can modify
                ;(printf "job policy = %L\n" jobPolicy)
                ;example that sets the job policy method to a LSF command string
                jobPolicy->distributionmethod = "Command"
                jobPolicy->jobsubmitcommand = "bsub -R \"OSREL==EE50\" -Is -J APS"
                jobPolicy->name = "LSF APS"
                (axlAttachJobPolicy session jobPolicy->name "ICRP" jobPolicy
testName)
            )
        )
    t)
)
```

Example6: To automatically perform a task after the setup is saved

This example shows how you can connect a procedure with the `setupSaved` trigger to perform a task after the setup is saved. Perform the following steps to setup this trigger:

1. In the `.cdsinit` file, define the function that you want to call when the `setupSaved` trigger is used.

```
(defun myfunc (session)
    printf("\nSetup is saved!!\n")
)
```

Note: Here, the `session` is obtained from the `sessionName` argument of the `setupSaved` trigger.

2. Connect ADE trigger to your function in the `.cdsinit` file.For example, `myfunc` in this case:

```
(defun connectADEXLTriggers_setupSaved (session)
    (axlSessionConnect session "setupSaved" 'myfunc)
)
```

3. Register a callback to connect the triggers on ADE session start.

```
(axlSessionRegisterCreationCallback 'connectADEXLTriggers_setupSaved)
```

The message, Setup is saved!!, is printed when the `setupSaved` trigger is used.

Example 7: To automatically add a variable when an ADE test is created

The following example shows how you can connect a procedure with the postCreatedTest trigger to automatically add a variable when an ADE test is created.

1. Define the following function in the .cdsinit file before launching virtuoso.

```
(defun ADEexampletriggerpostCreatedTest (session testname)
    (maeSetVar"VDD" "1.1" ?typeName "test" ?typevalue (list testnmae))
)
```

2. Register a callback to connect the triggers.

```
(axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session
"postCreatedTest" 'ADEexampletriggerpostCreatedTest)))
```

2

Setup Database Functions

Setup Database SKILL Functions

Function	Description
axlCloseSetupDB	Closes an open ADE XL setup database.
axlCommitSetupDB	Saves the setup database file, for instance, the maestro.sdb or data.sdb file.
axlCommitSetupDBAndHistoryAs	Saves the setup database along with history entries under a new name.
axlCommitSetupDBas	Saves the setup database under a new name.
axlDiffSetup	Compares two setup databases and reports the differences.
axlDeleteNote	Deletes a note from the given test, history, corner, parameters, variable or Reliability Analysis setup.
axlGetActiveSetup	Returns a handle to the active setup. You can use this handle to get or set setup details for the active setup.
axlGetCopyRefResultsOption	Returns if the simulation results are required to be copied or moved from the reference history based on the settings in the setup database.
axlGetActiveSetup	Returns a handle to the parent of the specified setup database element.
axlGetEnabled	Checks whether a setup database element is enabled or not.
axlExportSetup	Exports the setup from the currently loaded setupdb to a different file. The list of tags passed are the top-level elements like vars, tests, etc to export.
axlGetActiveSetup	Returns a handle to the active setup. You can use this handle to get or set setup details for the active setup.

Setup Database SKILL Functions, *continued*

Function	Description
axlGetHistoryGroupChildrenEntry	Finds a history entry in a group run by name and returns a handle to it.
axlGetNote	Returns note(s) from the specified test, history, corner, parameters, variable or Reliability Analysis setup.
axlGetPointNetlistDir	Returns the netlist directory for a particular corner and design point combination in the given test run of the specified history.
axlGetPointPsfDir	Returns the psf directory for a particular corner and design point combination in the given test run of the given history.
axlGetPointRunDir	Returns the psf directory for a particular corner and design point combination in the given test run of the given history.
axlGetPointTroubleshootDir	Returns the troubleshoot directory for a particular corner and design point combination in a specified test run of the given history.
axlGetReferenceHistoryItemName	Returns the name of the reference history for the active setup or checkpoint.
axlGetResultsLocation	Returns the results location for the specified setup database.
axlGetReuseNetlistOption	Checks if the option to use reference netlist is enabled for the setup database. This option helps in reusing the netlist of the reference history for the incremental simulation run.
axlGetScript	Finds a script by name and returns a handle to it.
axlGetSessionFromSetupDB	Determines the session associated with the provided setupdb handle.
axlGetScriptPath	Returns the path of a script.
axlGetScripts	Returns a list containing a handle to all scripts for this database entry and a list of all script names.
axlGetSessionFromSetupDB	Determines the session associated with the provided setupdb handle.

Setup Database SKILL Functions, *continued*

Function	Description
axlGetSetupDBDir	Returns the directory of the specified setup database.
axlGetSetupInfo	Returns the setup information for the complete ADE XL setup or a specific test. This includes the number of corners, sweeep points, and data points in the setup.
axlGetTopLevel	Returns a handle to the setup database containing the specified element.
axlGetUseIncremental	Checks if using reference results as cache during incremental run is enabled for the active setup or checkpoint.
axlImportSetup	Imports the setup from a file. The list of tags passed are the top-level elements like vars, tests, etc to import.
axlLoadSetupState	Loads a setup state.
axlNewSetupDB	Opens the named setup database and returns its handle. If the named setup database does not already exist, this function creates one and returns a handle to it.
axlNewSetupDBLCV	Creates a new setup db in the specified lib, cell, view location. It automatically overwrites any existing setup db in any of the above mentioned locations.
axlPutNote	Adds a note to the given test, history, corner, parameters, variable or Reliability Analysis setup.
axlPutScript	Inserts or finds a script by name, sets its path, and returns a handle to that script.
axlPutTest	Finds or inserts a test into the setup database and returns a handle to that test.
axlRemoveElement	Removes an element and all its children from the setup database.
axlResetActive	Resets the active setup database.

Setup Database SKILL Functions, *continued*

Function	Description
axlSaveSetup	Saves the setup database file and test state files for the current window. The behavior of this function is similar to File – Save. If the cellview was opened in the read-only mode, a new library, cell, and view value must be provided for which you need to save the setup details.
axlSaveSetupToLib	Saves the setup database to the specified lib/cell/view.
axlSDBDebugPrint	Prints the in-memory database rooted at the supplied handle.
axlSDBGetChild	Works as a wrapper around `axlSDBGetChildren()` and returns a setup database handle to a single child element.
axlSDBGetChildren	Returns child elements of the provided setup database handle.
axlSDBGetChildVal	Works as a wrapper around `axlSDBGetChild()` and returns the value for a single child element.
axlSDBGetExtension	Returns the named setup database extension.
axlSDBGetName	Returns the name of the provided setup database handle.
axlSDBGetValue	Returns the value of the provided setup database handle.
axlSDBHp	Returns whether the supplied argument is a valid setup database handle.
axlSDBPutExtension	Adds an extension to the ADE setup database under which customization can be created.
axlSDBSetChild	Creates a new child element under the provided setup database handle.
axlSDBSetMultipleEntry	Sets a setup database name to be a multiple entry name.
axlSDBSetValue	Sets the value of the provided setup database handle.

Setup Database SKILL Functions, *continued*

Function	Description
axlSetAllSweepsEnabled	Sets the selection status of the Point Sweeps check box in the Run Summary assistant pane.
axlSetEnabled	Enables or disables a setup database element, such as a test or a variable.
axlSetReferenceHistoryItemName	Sets the reference history name for the active setup or checkpoint. You can reuse the results or netlist from the reference history during an incremental simulation run. The reference history name set using this function also appears in the Reference field on the Reference History toolbar.
axlSetReuseNetlistOption	Enables or disables the option to use the reference netlist for the active setup or checkpoint. If this option is enabled, netlist of the design is reused for the incremental run. Otherwise, the design in renetlisted.
axlSetUseIncremental	Enables or disables the setup database option in active setup or checkpoint for using reference results as cache during incremental run. This function selects or clears the Use reference netlist check box on the Reference History form.
axlSetScriptPath	Sets the path of a script.
axlWriteDatasheet	Creates a datasheet for the specified history entry.
axlWriteDatasheetForm	Causes a form to appear so that you can specify various options for generating a datasheet.

axlCloseSetupDB

```
axlCloseSetupDB(
     x_sdb
     )
     => t / nil
```

Description

Closes an open ADE XL setup database.

Argument

x_sdb	Setup database.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlCloseSetupDB(sdbh)
t
```

axlCommitSetupDB

```
axlCommitSetupDB(
     x_hsdb
     )
     => x_hsdb
```

Description

Saves the setup database file, for instance, the maestro.sdb or data.sdb file.

Caution

This function does not save the test states file. Ensure that you use this function only when you need to save the setup database, else use the axlSaveSetup or maeSaveSetup function.

Argument

x_hsdb	Setup database handle.

Value Returned

x_hsdb	Setup database handle.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlCommitSetupDB( data_sdb )
1002
```

Reference

axlCreateSession, axlSetMainSetupDB

axlCommitSetupDBAndHistoryAs

```
axlCommitSetupDBAndHistoryAs(
     x_hsdb
     t_setupdbName
     )
     => x_hsdb
```

Description

Saves the setup database along with history entries under a new name.

Arguments

x_hsdb	Setup database handle.
t_setupdbName	New setup database file name.

Value Returned

x_hsdb	Setup database handle.

Example

```
axlCommitSetupDBAndHistoryAs(1002 "newData.sdb")
1004
```

axlCommitSetupDBas

```
axlCommitSetupDBas(
     x_hsdb
     t_setupdbName
     )
     => x_hsdb
```

Description

Saves the setup database under a new name.

Arguments

x_hsdb	Setup database handle.
t_setupdbName	New setup database file name.

Value Returned

x_hsdb	Setup database handle.

Example

```
axlCommitSetupDBas(1002 "data.sdb")
t
```

axlDiffSetup

```
axlDiffSetup(
     x_handlea
     x_handleb
     )
     => l_diffs
```

Description

Compares two setup databases and reports the differences.

Arguments

x_handlea	Setup handle.
x_handleb	Setup handle.

Value Returned

l_diffs	List of differences.

Example

```
::::::::::::::
diffa.sdb
::::::::::::::
<?xml version="1.0"?>
<setupdb>data
    <active>Active Setup
        <vars>
            <var>VDC
                <value>2.7</value>
            </var>
            <var>RLoad
                <value>1M</value>
            </var>
        </vars>
    </active>
    <history>History
```

```
    </history>
</setupdb>
::::::::::::::
diffb.sdb
::::::::::::::
<?xml version="1.0"?>
<setupdb>data
    <active>Active Setup
        <vars>
            <var>RLoad
                <value>10M</value>
            </var>
            <var>CLoad
                <value>1.5p</value>
            </var>
        </vars>
    </active>
    <history>History
    </history>
</setupdb>
--------------------------------------
\i diffah = axlNewSetupDB("diffa.sdb")
\t 2138
\p >
\i diffbh = axlNewSetupDB("diffb.sdb")
\t 2139
\p >
\i axlDiffSetup(diffah diffbh)
\t ("++ (active=Active Setup/vars=/var=CLoad)" "| (active=Active
Setup/vars=/var=RLoad/value=1M) -> 10M" "-- (active=Active
Setup/vars=/var=VDC)")
\p >
```

axlDeleteNote

```
axlDeleteNote(
     x_mainSDB
     t_item
     t_name
     )
     => t / nil
```

Description

Deletes a note from the given test, history, corner, parameters, variable or Reliability Analysis setup.

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

Arguments

x_mainSDB	Handle to the main setup database.
t_item	Type of the element from which you need to delete a note. Valid values: `"test"`, `"history"`, `"corner"`, `"globalvar"`, `"parameters"`, or `"relxanalysis"`
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 for corner C0:
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlDeleteNote(x_mainSDB "corner" "C0")
=> t
```

Related Functions

axlPutNote

axlGetCopyRefResultsOption

```
axlGetCopyRefResultsOption(
     x_hsdb
     )
     => t / nil
```

Description

Returns if the simulation results are required to be copied or moved from the reference history based on the settings in the setup database.

Argument

x_hsdb	Setup database handle.

Value Returned

`t`	Specifies that the simulation results need to be copied from the reference history.
`nil`	Specifies that the simulation results need to be moved from the reference history.

Example

```
sdb=axlGetMainSetupDB(axlGetWindowSession())
axlGetCopyRefResultsOption(sdb)
t
```

axlGetElementParent

```
axlGetElementParent(
     x_element
     )
     => x_parent / nil
```

Description

Returns a handle to the parent of the specified setup database element.

Argument

x_element	Setup database element handle.

Value Returned

x_parent	Handle to the parent of *x_element*. For example, if *x_element* is the handle to a variable's value, *x_parent* is the handle to the variable; if *x_element* is the handle to a variable, *x_parent* is the handle to the set of variables; and so on up to the top-level setup database handle.
g_errorOrZero	Error message if input argument is invalid or zero if the element has no parent.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetElementParent( axlGetHistoryEntry( data_sdb "data_design_verification" ) )
1004
```

Reference

axlCreateSession, axlSetMainSetupDB, axlGetHistoryEntry

axlGetEnabled

```
axlGetEnabled(
     x_element
     )
     => t / nil
```

Description

Checks whether a setup database element is enabled or not.

Argument

x_element	Setup database element handle.

Value Returned

t	Element is enabled.
nil	Element is not enabled.

Example

```
axlGetEnabled(1021)
nil
```

axlGetLocalResultsDir

```
axlGetLocalResultsDir(
     x_historyHandle
     )
     => t_dirPath / nil
```

Description

A local results directory associated with a run on a remote machine.

Argument

x_historyHandle	Handle to a history item.

Value Returned

t_dirPath	Path to the local results directory associated with a run on a remote machine.
nil	If the handle to the history is invalid.

Example

```
session = axlGetWindowSession()
"session1"
sdb = (axlGetMainSetupDB session)
1675
h = axlGetHistoryEntry(sdb "Interactive.0")
1712
axlGetLocalResultsDir(h)
"/tmp/machineName_userName_134646275"
```

axlIsLocalResultsDir

```
axlIsLocalResultsDir(
     x_historyHandle
     )
     => t / nil
```

Description

Returns the status of Use Local Simulation Results Directory flag for the specified history item.

Argument

x_historyHandle	Handle to the history item.

Value Returned

t	The flag is enabled.
nil	The flag is not enabled.

Example

```
session = axlGetWindowSession()
"session1"
sdb = (axlGetMainSetupDB session)
1675
h = axlGetHistoryEntry(sdb "Interactive.0")
1712
axlIsLocalResultsDir(h)
t
```

axlExportSetup

```
axlExportSetup(
     t_session
     x_hsdb
     t_path
     l_tags
     )
     => t / nil
```

Description

Exports the setup from the currently loaded setupdb to a different file. The list of tags passed are the top-level elements like vars, tests, etc to export.

Arguments

t_session	Session Name.
x_hsdb	Handle to a setup database.
t_path	Setup path.
l_tags	Setup handle. The pre defined values for l_tags are: vars, tests, parameters, corners, runoptions and scripts.

Value Returned

`t`	Successful Operation
`nil`	Unsuccessful Operation

Example

```
session = axlGetWindowSession()
"session1"
sdb=axlGetMainSetupDB(session)
2141
axlExportSetup(session sdbh "/tmp/exported.sdb" `("vars"))
t
exported.sdb:
<?xml version="1.0"?>
```

```
<setupdb>exported
    <active>Active Setup
        <vars>
            <var>CLoad
                <value>1.5p</value>
            </var>
            <var>RLoad
                <value>10M</value>
            </var>
            <var>VDC
                <value>2.7</value>
            </var>
        </vars>
    </active>
    <history>History</history>
</setupdb>
```

axlGetHistoryGroupChildren

```
axlGetHistoryGroupChildren(
     x_element
     )
     => l_children
```

Description

Returns a list containing a handle to all history children entries in the history group and a list of names of all the history children entries.

Argument

x_element	Setup database element handle.

Value Returned

l_children	A list that contains the following: ■ a handle to all history entries in the history group ■ a list of names of all the history entries.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetHistoryGroup( data_sdb "Group.0" )
=> 1169
axlGetHistoryGroupChildren( 1169 )
=> (1172 ("Group.0.Run.0" "Group.0.Run.1" "Group.0.Run.2"))
```

Here, 1172 is the handle to the history entries and `Group.0.Run.0`, `Group.0.Run.1`, and `Group.0.Run.2` are the names of history entries.

axlGetActiveSetup

```
axlGetActiveSetup(
     x_mainSDB
     )
     => x_activeSetup
```

Description

Returns a handle to the active setup. You can use this handle to get or set setup details for the active setup.

Argument

x_mainSDB	Handle to the main setup database.

Value Returned

x_activeSetup	Handle to the active setup.

Example

The following example gets the status of the Overwrite History save option:

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))
axlGetOverwriteHistory(x_activeSetup)
```

axlGetHistoryGroupChildrenEntry

```
axlGetHistoryGroupChildrenEntry(
     x_childrenHandle
     t_name
     )
     => x_history / 0
```

Description

Finds a history entry in a group run by name and returns a handle to it.

Arguments

x_childrenHandle	Setup database handle to the children of a group run.
t_name	Name of the history entry

Value Returned

x_history	Handle to a history entry.
0	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetHistoryGroup( data_sdb "Group.0" )
=> 1169
axlGetHistoryGroupChildren( 1169 )
=>(1172 ("Group.0.Run.0" "Group.0.Run.1" "Group.0.Run.2"))
axlGetHistoryGroupChildrenEntry( 1172 "Group.0.Run.0" )
=>1173
```

Reference

axlCreateSession, axlSetMainSetupDB, axlGetActiveSetup

axlGetNote

```
axlGetNote(
     x_hsdb
     t_item
     t_name
     )
     => t_note / nil
```

Description

Returns note(s) from the specified test, history, corner, parameters, variable or Reliability Analysis setup.

Arguments

x_hsdb	Handle to the main setup database.
t_item	Type of the element from which you want to get notes. Valid values: `""test"`, `"history"`, `"corner"`, `"globalvar"`, `"parameters"`, or `"relxanalysis"`
t_name	Name of the element from which you want to get notes.

Value Returned

t_note	Note from the specified element.
`nil`	Unsuccessful operation.

Example

```
; The following example code shows how to retrieve a note for corner C0:
axlGetNote(1001 "corner" "C0")
=> "Notes -- name : Corner C0, temp : 30, CAP : 2p"
```

Related Functions

axlDeleteNote

axlPutNote

axlGetPointNetlistDir

```
axlGetPointNetlistDir(
     x_historyID
     t_testName
     [ ?cornerName t_cornerName ]
     [ ?designPointId x_designPointId ] )
     => t_pointNetlistDir / nil
```

Description

Returns the netlist directory for a particular corner and design point combination in the given test run of the specified history.

Arguments

x_historyID	History ID
t_testName	Name of the test
?cornerName t_cornerName	Name of the corner
?designPointId x_designPointID	Point ID

Important

If cornerName is specified then pointID must also be specified. If both `cornerName` and `pointID` arguments are not specified, top PSF level netlist directory is returned.

Value Returned

t_pointNetlistDir	Name of the netlist directory.
`nil`	Returns `nil` otherwise

Example

Example 1:

```
axlGetPointNetlistDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12") "opamplib:ampTest:1")
```

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/psf/op
amplib:ampTest:1/netlist"
```

Example 2:

```
axlGetPointNetlistDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)
```

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/2/opam
plib:ampTest:1/netlist"
```

axlGetPointPsfDir

```
axlGetPointPsfDir(
     x_historyID
     t_testName
     [ ?cornerName t_cornerName ]
     [ ?designPointId x_designPointId ]
     )
     => t_pointPsfDir / nil
```

Description

Returns the psf directory for a particular corner and design point combination in the given test run of the given history.

Arguments

x_historyID	History ID
t_testName	Name of the test
?cornerName t_cornerName	Name of the corner
?designPointId x_designpointID	Point ID

Important

If cornerName is specified then pointID must also be specified. If both `cornerName` and `pointID` arguments are not specified, top psf level directory is returned.

Value Returned

t_pointPsfDir	Name of the point psf directory.
`nil`	Returns `nil` otherwise

Example

Example 1:

```
axlGetPointPsfDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12") "opamplib:ampTest:1")
```

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/psf/op
amplib:ampTest:1/psf"
```

Example 2:

```
axlGetPointPsfDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)
```

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/2/opam
plib:ampTest:1/psf"
```

axlGetPointRunDir

```
axlGetPointRunDir(
     x_historyID
     t_testName
     [ ?cornerName t_cornerName ]
     [ ?designPointId x_designPointId ]
     )
     => t_pointRunDir / nil
```

Description

Returns the run directory for a particular corner and design point combination in the given test run of the specified history.

Arguments

x_historyID	History ID
t_testName	Name of the test
?cornerName *t_cornerName*	Name of the corner
?designPointId *x_designPointID*	Point ID

Important

If cornerName is specified then pointID must also be specified. If both `cornerName` and `pointID` arguments are not specified, top PSF level point run directory is returned.

Value Returned

t_pointRunDir	Name of the point run directory.
`nil`	Returns `nil` otherwise

Example

Example 1:

```
axlGetPointRunDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")"Interactive.12
") "opamplib:ampTest:1")

"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/
psf/opamplib:ampTest:1/"
```

Example 2:

```
axlGetPointRunDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")"Interactive.12
") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)

"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/
2/opamplib:ampTest:1/"
```

axlGetPointTroubleshootDir

```
axlGetPointTroubleshootDir(
     x_historyID
     t_testName
     [ ?cornerName t_cornerName ]
     [ ?designPointId x_designPointId ]
     )
     => t_trblDir / nil
```

Description

Returns the troubleshoot directory for a particular corner and design point combination in a specified test run of the given history.

Arguments

x_historyID	History ID
t_testName	Name of the test
?cornerName *t_cornerName*	Name of the corner
?designPointId *x_designPointID*	Point ID

Important

If cornerName is specified then pointID must also be specified. If both `cornerName` and `pointID` arguments are not specified, top PSF level directory is returned.

Value Returned

t_trblDir	Name of the troubleshoot directory.
`nil`	Returns `nil` otherwise

Example

Example 1:

```
axlGetPointTroubleshootDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12.TS.0") "opamplib:ampTest:1")
```

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12.TS.0/p
sf/opamplib:ampTest:1/troubleshoot"
```

Example 2:

```
axlGetPointTroubleshootDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12.TS.0") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)
```

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12.TS.0/2
/opamplib:ampTest:1/troubleshoot"
```

axlGetReferenceHistoryItemName

```
axlGetReferenceHistoryItemName
     x_hsdb
     )
     => t_referenceHistoryName
```

Description

Returns the name of the reference history for the active setup or checkpoint.

Argument

x_hsdb	Setup database handle to the active setup or checkpoint.

Value Returned

t_referenceHistoryName	Name of the reference history.

Example

```
data_session = ( axlCreateSession "data_session" )
axlGetReferenceHistoryItemName(axlGetMainSetupDB(data_session))
"Interactive.0"
```

Reference

axlCreateSession, axlGetMainSetupDB

axlGetResultsLocation

```
axlGetResultsLocation(
     x_hsdb
     )
     => t_resultsLocation / nil
```

Description

Returns the results location for the specified setup database.

Note: If adexl.results_saveResDir is set to a location other than the default setting, axlGetResultsLocation will use its setting to determine the results location. But if it is set to the default location, axlGetResultsLocation will use the adexl.results_saveDir setting to determine the results location.

Argument

x_hsdb	Setup database handle.

Value Returned

t_resultsLocation	Results location which includes a directory named from the setup database name prefix.
nil	Unsuccessful get operation.

Example

If you do not set the adexl.results_saveDir environment variable in your .cdsenv:

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetResultsLocation( data_sdb )
"myLib/myCell/adexl/results/data"
```

If you set the adexl.results_saveDir environment variable to RESULTS as follows:

```
adexl.results_saveDir string "RESULTS"
```

this function returns the following instead:

```
"RESULTS/myLib/myCell/adexl/results/data"
```

Here is another example (where adexl.results_saveDir is not set):

```
resultsLoc = (axlGetResultsLocation (axlGetHistoryEntry (axlGetMainSetupDB
axlGetWindowSession() ) ) )
"opamplib/ampTest/adexl/results/data"
```

Reference

axlCreateSession, axlGetHistoryEntry, axlGetMainSetupDB, axlSetMainSetupDB

axlGetReuseNetlistOption

```
axlGetReuseNetlistOption(
     x_hsdb
     )
     => t / nil
```

Description

Checks if the option to use reference netlist is enabled for the setup database. This option helps in reusing the netlist of the reference history for the incremental simulation run.

Argument

x_hsdb	Setup database handle to active setup or checkpoint.

Value Returned

`t`	Specifies that the option to use reference netlist is enabled.
`nil`	Specifies that the option to use reference netlist is not enabled and a new netlist needs to be created.

Example

```
data_session = ( axlCreateSession "data_session" )
axlGetReuseNetlistOption(axlGetMainSetupDB(data_session))
t
```

axlGetScript

```
axlGetScript(
     x_element
     t_scriptName
     )
     => x_script / nil
```

Description

Finds a script by name and returns a handle to it.

Arguments

x_element	Setup database element handle.
t_scriptName	Script name.

Value Returned

x_script	Script handle.
nil	Unsuccessful add operation.

Example

```
axlGetScript(1021 "myScript.nam")
=> 1045
```

axlGetScriptPath

```
axlGetScriptPath(
     x_script
     )
     => t_path / nil
```

Description

Returns the path of a script.

Argument

x_script	Script handle.

Value Returned

t_path	Script path.
nil	Unsuccessful operation.

Example

```
axlGetScriptPath(1045)
=> "myData/scripts"
```

axlGetScripts

```
axlGetScripts(
     x_element
     )
     => l_scripts / nil
```

Description

Returns a list containing a handle to all scripts for this database entry and a list of all script names.

Argument

x_element	Setup database element handle.

Value Returned

l_scripts	List containing a handle to all scripts for this database entry and a list of all script names.
nil	Unsuccessful operation.

Example

```
axlGetScripts 1045
'((1001 "script1.ocn")
  (1002 "script2.ocn")
 )
```

axlGetSessionFromSetupDB

```
axlGetSessionFromSetupDB(
     x_hsdb
     )
     => t_sessionName / nil
```

Description

Determines the session associated with the provided setupdb handle.

Argument

x_hsdb	Determines the session associated with the handle to a setup database.

Value Returned

t_sessionName	Session name
nil	Unsuccessful operation.

Example

```
db=axlGetMainSetupDB("session0")
1001
axlGetSessionFromSetupDB(db)
"session0"
```

axlGetSetupDBDir

```
axlGetSetupDBDir(
     x_hsdb
     )
     => t_dir / nil
```

Description

Returns the directory of the specified setup database.

Argument

x_hsdb	Handle to the specified setup database.

Value Returned

t_dir	Setup database directory.
nil	Unsuccessful get operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetSetupDBDir( data_sdb )
"myLib/myCell/adexl"
```

Reference

axlCreateSession, axlSetMainSetupDB

axlGetSetupInfo

```
axlGetSetupInfo(
     t_sessionName
     [ ?testName testName ]
     )
     => r_setupInfo / nil
```

Description

Returns the setup information for the complete ADE XL setup or a specific test. This includes the number of corners, sweeep points, and data points in the setup.

Argument

t_sessionName	Name of the active ADE XL session
?testName *testName*	Test name for which the setup information is to be retrieved. If the test name is not provided, ADE XL displays the setup information for all the tests. Default value: ""

Value Returned

r_setupInfo	A struct containing the count of corners, sweeps, and data points.
nil	Unsuccessful get operation.

Example

The following example code shows how you can view the setup information for the active ADE XL setup:

```
session = axlGetWindowSession()
=> "session0"

x_mainSDB = axlGetMainSetupDB(session)
=>1001
```

```
;; display the total number of points, corners, and data points for all the corners
axlGetSetupInfo(session)
=> (nil Corners 6 SweepPoints 1 DataPoints 7)
;
;; display the number of points, corners, and data points for the AC test
axlGetSetupInfo(session ?testName "AC")
=> (nil Corners 3 SweepPoints 1 DataPoints 3)
;
;; display the number of points, corners, and data points for the TRAN test
x_setupInfo = axlGetSetupInfo(session ?testName "TRAN")
=> (nil Corners 3 SweepPoints 2 DataPoints 6)

;; you can use the handle to the returned struct to get the count of
;; corners and sweep points individually as shown below.
;
x_setupInfo->Corners
=>3
;
x_setupInfo->SweepPoints
=>2
;
x_setupInfo->DataPoints
=>6
```

axlGetTopLevel

```
axlGetTopLevel(
     x_element
     )
     => x_hsdb / g_errorOrZero
```

Description

Returns a handle to the setup database containing the specified element.

Argument

x_element	Setup database element handle.

Value Returned

x_hsdb	Setup database handle.
g_errorOrZero	Error message if input argument is invalid or zero for otherwise unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetTopLevel( axlGetHistoryEntry( data_sdb "data_design_verification" ) )
1004
```

Reference

axlCreateSession, axlSetMainSetupDB, axlGetHistoryEntry

axlGetUseIncremental

```
axlGetUseIncremental(
     x_hsdb
     )
     => t / nil
```

Description

Checks if using reference results as cache during incremental run is enabled for the active setup or checkpoint.

Argument

x_hsdb	Setup database handle to the active setup or checkpoint.

Value Returned

t	Specifies that using reference results as cache during incremental run is enabled.
nil	Specifies that using reference results as cache during incremental run is not enabled.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlGetUseIncremental(data_sdb)
=> t
```

Reference

axlCreateSession, axlSetMainSetupDB,

axlImportSetup

```
axlImportSetup(
     t_session
     t_path
     l_tags
     [ t_historyName ]
     [ s_operation ]
     )
     => t / nil
```

Description

Imports the setup from a file. The list of tags passed are the top-level elements like vars, tests, etc to import.

Arguments

t_session	Session Name.
t_path	Setup path.
l_tags	List of tags.
t_historyName	Name of the history from where you need to import the setup.
s_operation	Operation Type Valid values: `"merge"`, `"retain"` or `"overwrite"` For more information about the merge, retain and overwrite options, see the *Analog Design Environment XL User Guide*.

Value Returned

`t`	Successful Operation
`nil`	Unsuccessful Operation

Example

```
session = axlGetWindowSession(hiGetCurrentWindow())
"session1"
```

```
axlImportSetup(session "/tmp/exported.sdb" `("vars") "" 'retain)
t
```

axlLoadSetupState

```
axlLoadSetupState(
     t_session
     t_stateName
     l_tags
     s_operation
     )
     => t / nil
```

Description

Loads a setup state.

Arguments

t_session	Session Name.
t_stateName	Setup state name.
l_tags	List of tags. You can use one or more tags from the following list: `tests` - Testbench setups `vars` - Global variables `parameters` - Parameters and their values `currentMode` - Run mode `runOptions` - Simulation options for different run modes and the run distribute options `specs` - Parameter specifications `corners` - Corner details `modelGroups` - Model groups `extensions` - Extensions `relxanalysis` - Reliability analysis setup details `All` - Loads all the details for tests, vars, parameters, currentMode, runOptions , specs, corners, modelGroups, extensions, and relxanalysis from the specified state.
s_operation	Operation to handle the existing details Valid values: "merge", "retain", "overwrite", and "imply".

Value Returned

`t`	Successful Operation
`nil`	Unsuccessful Operation

Example

```
session = (axlGetWindowSession)
"session1"
(axlSetupStates session)
"state1"
```

```
(axlLoadSetupState session "state1" `("vars") 'merge)
t
axlLoadSetupState(session "save_all_state" "All" 'overwrite)
t
```

axlNewSetupDB

```
axlNewSetupDB(
     t_setupdbName
     )
     => x_hsdb / nil
```

Description

Opens the named setup database and returns its handle. If the named setup database does not already exist, this function creates one and returns a handle to it.

Argument

t_setupdbName	Setup database name.

Value Returned

x_hsdb	Setup database handle.
nil	Unsuccessful open/create operation.

Example

```
axlNewSetupDB( "data" )
1
```

axlNewSetupDBLCV

```
axlNewSetupDBLCV(
     t_libraryName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Creates a new setup db in the specified lib, cell, view location. It automatically overwrites any existing setup db in any of the above mentioned locations.

Arguments

t_libraryName	Library Name
t_cellName	Cell Name
t_viewName	View Name

Value Returned

t	successful operation
nil	Unsuccessful operation

Example

```
sdb = axlNewSetupDBLCV("myLib" "myCell" "myView")
```

axlPutNote

```
axlPutNote(
     x_mainSDB
     t_item
     t_name
     t_note
     )
     => t / nil
```

Description

Adds a note to the given test, history, corner, parameters, variable or Reliability Analysis setup.

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

Arguments

x_mainSDB	Handle to the main setup database.
t_item	Type of the element to which you need to add a note.
	Valid values: `"test"`, `"history"`, `"corner"`, `"globalvar"`, `"parameters"`, or `"relxanalysis"`
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 corner C0:
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlPutNote(x_mainSDB "corner" "C0" "this is a fast corner")
=> t
```

Related Functions

axlDeleteNote

axlPutScript

```
axlPutScript(
     x_element
     t_scriptName
     t_path
     )
     => x_script / nil
```

Description

Inserts or finds a script by name, sets its path, and returns a handle to that script.

Arguments

x_element	Setup database element handle.
t_scriptName	Script name.
t_path	Path.

Value Returned

x_script	Script handle.
nil	Unsuccessful operation.

Example

```
axlPutScript (1004 "scriptname" "/path/to/script/file")
1005
```

axlPutTest

```
axlPutTest(
     x_hsdb
     t_test
     t_tool
     )
     => x_test / nil
```

Description

Finds or inserts a test into the setup database and returns a handle to that test.

Arguments

x_hsdb	Setup database handle.
t_test	Test name.
t_tool	Tool name.

Value Returned

x_test	Test handle.
nil	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlPutTest( data_sdb "data_dead_band" "ADE")
2201
```

Reference

axlCreateSession, axlSetMainSetupDB

axlRemoveElement

```
axlRemoveElement(
     x_element
     )
     => t / nil
```

Description

Removes an element and all its children from the setup database.

Argument

x_element	Setup database element handle.

Value Returned

t	Successful remove operation.
nil	Unsuccessful remove operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlRemoveElement( axlGetHistoryEntry( data_sdb "data_design_verification" ) )
t
```

Reference

axlCreateSession, axlSetMainSetupDB, axlGetHistoryEntry

axlResetActive

```
axlResetActive(
     x_hsdb
     )
     => t / nil
```

Description

Resets the active setup database.

Argument

x_hsdb	Setup database handle.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlResetActive( 1003 )
t
```

axlSaveSetup

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

Description

Saves the setup database file and test state files for the current window. The behavior of this function is similar to *File – Save*. If the cellview was opened in the read-only mode, a new library, cell, and view value must be provided for which you need to save the setup details.

Argument

None.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlSaveSetup( )
=>t
```

axlSaveSetupToLib

```
axlSaveSetupToLib(
     x_hsdb
     t_libName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Saves the setup database to the specified lib/cell/view.

Arguments

x_hsdb	Setup database handle.
t_libName	Library name.
t_cellName	Cell name.
t_viewName	View name.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlSaveSetupToLib( 1001 "lib" "cell" "view" )
t
```

axlSDBDebugPrint

```
axlSDBDebugPrint(
     x_SDBH
     )
     => x_SDBH
```

Description

Prints the in-memory database rooted at the supplied handle. Useful for debugging so that SDB modifications can be seen without saving the XML to disk.

Arguments

x_SDBH	Setup database handle.

Value Returned

x_SDBH	Handle of the supplied *x_SDBH*, but the return value is inconsequential as this function is called for the printed tree.

Example

;; given mainSDBH pointing to current setup and some variables

```
varsSDBH = car( axlGetVars(mainSDBH))
axlSDBDebugPrint(varsSDBH) ->

  1297--\vars|''
    1304---\var|RES  <enabled: 1>
        0---->value|4k
    1338---\var|BOB
    0---->value|1 2 3
```

axlSDBGetChild

```
axlSDBGetChild(
     x_SDBH
     t_name
     [ t_value ]
     )
     => x_hsdb / nil
```

Description

Works as a wrapper around `axlSDBGetChildren` and returns a setup database handle to a single child element. An error will be reported if multiple matching elements are found.

Arguments

x_SDBH	Setup database handle.
t_name	Name of the element.
t_value	Value of multi-entry names.

Value Returned

x_SDBH	Setup database handle to the child element.
`nil`	Unsuccessful operation.

Example

```
extensionSDBH = (axlSDBPutExtension mainSDBH "test extension")
1234 = (axlSDBSetChild extensionSDBH "King" "Louis XIV")


axlSDBGetChild( extensionSDBH "King")
=> 1234
```

axlSDBGetChildren

```
axlSDBGetChildren(
     x_SDBH
     [ t_name ]
     [ t_value ]
     S_returnAs
     )
     => l_children / nil
```

Description

Returns child elements of the provided setup database handle.

Arguments

x_SDBH	Setup database handle.
t_name	Optional filtration by element name. If not provided, all child elements of the parent will be returned.
t_value	Optional filtration by element value. If provided, *t_name* must also be specified.
S_returnAs	Format of the return value. Valid values are: ■ `'handles`: Returns a list of the setup database handles of hte child elements. ■ `'assoc`: Returns a list of name-value pairs, such as ((elt1Name elt1Val) (elt2Name elt2Val)). Defaul value: `'handles`

Value Returned

l_children	A list corresponding to the value set for *s_returnAs*.
nil	Unsuccessful operation.

Example

```
;; given x_mainSDB pointing to current setup
extensionSDBH = axlSDBPutExtension( x_mainSDB "test extension")
axlSDBSetChild( extensionSDBH "King" "Louis XIV")

axlSDBGetChildren(extensionSDBH ?name "King" ?returnAs 'assoc)
=> (("King" "Louis XIV"))

axlSDBSetMultipleEntry( "King")
axlSDBSetChild( extensionSDBH "King" "Henry VII")

axlSDBGetChildren(extensionSDBH ?name "King" ?returnAs 'handles)
  => '(1852 1854)

axlSDBGetChildren(extensionSDBH ?name "King"  ?returnAs 'assoc)
  =>  (("King" "Louis XIV") ("King" "Henry VII"))

axlSDBGetChildren(extensionSDBH ?name "King" ?value "Louis XIV" ?returnAs 'assoc)
  => (("King" "Louis XIV"))
```

axlSDBGetChildVal

```
axlSDBGetChildVal(
     x_SDBH
     t_name
     )
     => t_value / nil
```

Description

Works as a wrapper around `axlSDBGetChild` and returns the value for a single child element.

Arguments

x_SDBH	Setup database handle.
t_name	Name of the element.

Value Returned

t_value	Value of the child element.
`nil`	Unsuccessful operation.

Example

```
extensionSDBH = axlSDBPutExtension( mainSDBH "test extension")
axlSDBSetChild( extensionSDBH "King" "Louis XIV")


axlSDBGetChildVal( extensionSDBH "King")
=> "Louis XIV"
```

axlSDBGetExtension

```
axlSDBGetExtension(
     x_mainSDBH
     t_extensionName
     )
     => x_SDBH / nil
```

Description

Returns the named setup database extension.

Arguments

x_mainSDBH	Main setup database handle.
t_extensionName	Name of an extension previously added with `axlSDBPutExtension()`.

Value Returned

x_SDBH	Setup database handle of the named extension.
`nil`	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup that's had an extension previously added
with:
axlSDBPutExtension( mainSDBH "test extension")

axlSDBGetExtension( mainSDBH "test extension")
=> x_SDBH
```

axlSDBGetName

```
axlSDBGetName(
     x_SDBH
     )
     => t_name / nil
```

Description

Returns the name of the provided setup database handle.

Arguments

x_SDBH	Setup database handle.

Value Returned

t_name	Name of the provided setup database handle.
nil	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup and some variables
varSDBH = axlGetVar(mainSDBH "VDC")
axlSDBGetName(varSDBH)
=> "var"
```

axlSDBGetValue

```
axlSDBGetValue(
     x_SDBH
     )
     => t_value / nil
```

Description

Returns the value of the provided setup database handle.

Arguments

x_SDBH	Setup database handle.

Value Returned

t_value	Value of the provided setup database handle.
nil	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup and some variables
varSDBH = axlGetVar(mainSDBH "VDC")
axlSDBGetName(varSDBH)
=> "VDC"
```

axlSDBHp

```
aaxlSDBHp(
     g_potentialSDBH
     )
     => t / nil
```

Description

Returns whether the supplied argument is a valid setup database handle.

It is more accurate than checking whether a handle is greater than zero because it verifies that the element exists. Therefore, this function will return `nil` if a previously valid but now invalid (stale) handle is provided.

Arguments

g_potentialSDBH	Setup database handle.

Value Returned

`t`	The setup database handle is valid.
`nil`	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup and variable "VDC"
varSDBH = axlGetVar(mainSDBH "VDC")
axlSDBHp(varSDBH)
=> t
```

To remove the variable:

```
axlRemoveElement(varSDBH)

;; now the handle is invalid
axlSDBHp(varSDBH)
=> nil
```

axlSDBPutExtension

```
axlSDBPutExtension(
     x_mainSDBH
     t_extensionName
     )
     => x_SDBH / nil
```

Description

Adds an extension to the ADE setup database under which customization can be created.

Arguments

x_mainSDBH	Main setup database handle.
t_extensionName	Name of the extension.

Value Returned

x_SDBH	Setup database handle of the named extension.
nil	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup
extensionSDBH = axlSDBPutExtension( mainSDBH "test extension")
```

axlSDBSetChild

```
axlSDBSetChild(
     x_SDBH
     t_name
     t_value
     )
     => x_SDBH / nil
```

Description

Creates a new child element under the provided setup database handle.

Arguments

x_SDBH	Setup database handle under which a new child element is to be added **Note:** To avoid database corruption and undefined behavior, the provided handle's ancestor should be a previously added extension created with `axlSDBPutExtension()`.
t_name	Name of the new element. Following are the naming conventions: ■ must start with an upper-case letter ■ can contain letters, digits, underscores, hyphens, and periods ■ cannot contain whitespace **Note:** Names beginning with lower-case letters are reserved for future ADE expansion.
t_value	Value of the new element.

Value Returned

x_SDBH	Setup database handle of the child element.
`nil`	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup
extensionSDBH = axlSDBPutExtension( mainSDBH "test extension")
kingSDBH = axlSDBSetChild( extensionSDBH "King" "Louis XIV")
```

Note: Any name is either single or multi-entry. A single-entry element is differentiated by its name only, whereas a multi-entry element is differentiated by both name and value. All children of a particular element must be unique.

Single-entry:

```
extensionSDBH = axlSDBPutExtension( mainSDBH "test extension")
kingSDBH = axlSDBSetChild( extensionSDBH "King" "Louis XIV")
```

Since "King" is a single-entry name (the default), there can only be one child of extensionSDBH having "King" as its name. Attempting to specify another child element, such as:

```
axlSDBSetChild( extensionSDBH "King" "Henry VIII")
```

will update the value of the pre-existing element, and not create a new element. A name can be made multi-entry by calling `axlSDBSetMultipleEntry()`, after which calls to `axlSDBSetChild()` with same name and different value will create new chilld elements. For example:

```
;; create an aggregating child under king.  This isn't necessary but keeps the
database tidy
;; we are not supplying a value because this is only demarcating a collection
ministersSDBH = axlSDBSetChild( kingSDBH "Ministers" "")

;; set the Minister tag as multi-entry.  No session or SDB handle is provided, and
this affects *all* open ADE sessions and databases in the current
;; virtuoso invocation.
axlSDBSetMultipleEntry( "Minister")

;; OK, now we can add multiple ministers
axlSDBSetChild( ministersSDBH "Minister" "Alice")
axlSDBSetChild( ministersSDBH "Minister" "Bob")
axlSDBSetChild( ministersSDBH "Minister" "Eve")
```

axlSDBSetMultipleEntry

```
axlSDBSetMultipleEntry(
     t_name
     )
     => t / nil
```

Description

Sets a setup database name to be a multiple entry name. By default, a setup database handle is a single entry name. A single-entry element is differentiated only by its name, whereas a multi-entry element is differentiated by both name and value.

Since all children of a particular element must be unique, this means that a single-entry name can only be specified at most once whereas a multi-entry name can have siblings having the same name. Some examples of built-in multiple entry names in ADE include var, corner, or test.

The table containing names specified to be multi-entry is shared across all ADE sessions in the same virtuoso process, and therefore no session or setup database handle is necessary. The list of multi-entry names will be written upon setup database save and restored upon load; it is not necessary to call `axlSDBSetMultipleEntry()` at each session invocation.

Arguments

t_name	Name of the element to be set as multi-entry. Following are the naming conventions: ■ must start with an upper-case letter ■ can contain letters, digits, underscores, hyphens, and periods ■ cannot contain whitespace **Note:** Names beginning with lower-case letters are reserved for future ADE expansion.

Value Returned

`t`	Successfully set the setup database handle as a multi-entry element.
`nil`	Unsuccessful operation.

Example

```
axlSDBSetMultipleEntry("Ministers")
=> t
```

axlSDBSetValue

```
axlSDBSetValue(
     x_SDBH
     t_newValue
     )
     => t / nil
```

Description

Sets the value of the provided setup database handle.

An example of failure is providing a value that would cause this setup database element to have the same name and value pair as another child of its parent.

Arguments

x_SDBH	Setup database handle.
t_newValue	The new value to be set to the setup database handle.

Value Returned

`t`	Successfully set new value to the setup database handle.
`nil`	Unsuccessful operation.

Example

```
;; given mainSDBH pointing to current setup
extensionSDBH = axlSDBPutExtension( mainSDBH "test extension")
kingSDBH = axlSDBSetChild( extensionSDBH "King" "Louis XIV")
axlSDBSetValue(kingSDBH "myVDC")
=> "Henry VII"
=> t
```

axlSetAllSweepsEnabled

```
axlSetAllSweepsEnabled(
     x_hsdb
     g_enableStatus
     )
     => t / nil
```

Description

Sets the selection status of the *Point Sweeps* check box in the Run Summary assistant pane.

Arguments

x_hsdb	Setup database handle.
g_enableStatus	Option for setting the selection status. Valid values: ■ `1` - selects the Point Sweeps check box. ■ `0` - deselects the Point Sweeps check box.

Value Returned

`t`	Successful select or deselect operation.
`nil`	Unsuccessful select or deselect operation.

Example

To select the Point Sweeps check box.

```
db=axlGetMainSetupDB("session0")
1001
axlSetAllSweepsEnabled(db 1)
t
```

To deselect the Point Sweeps check box.

```
db=axlGetMainSetupDB("session0")
1001
axlSetAllSweepsEnabled(db 0)
t
```

axlSetCopyRefResultsOption

```
axlSetCopyRefResultsOption(
     x_hsdb
     g_value
     )
     => x_hsdb / 0
```

Description

Sets whether the simulation results need to be copied or moved from the reference history.

Arguments

x_hsdb	Setup database handle.
g_value	Boolean value. `t` specifies that the simulation results need to be copied and `nil` specifies that they need to be moved.

Value Returned

x_hsdb	Setup handle is returned if the option is set successfully
`0`	Not successful

Example

```
sdb=axlGetMainSetupDB(axlGetWindowSession())
axlSetCopyRefResultsOption(sdb t)
t
```

axlSetEnabled

```
axlSetEnabled(
     x_element
     g_enable
     )
     => t / nil
```

Description

Enables or disables a setup database element, such as a test or a variable.

Arguments

x_element	Setup database element handle.	
g_enable	Enable flag Valid Values:	
	`nil`	Disabled
	any other value	Enabled

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example code shows how to use the axlSetEnabled function to enable or disable different elements in the setup database:

```
x_mainDB = axlGetMainSetupDB(axlGetWindowSession())
=>1001
;; Enable a test
testHandle = axlGetTest( x_mainDB "data_dead_band" )
=> 1005
axlSetEnabled( testHandle t )
=> 1
;; Disable tests
foreach(test cadr(axlGetTests(1001))
        axlSetEnabled( axlGetTest(1001 test ) nil ))
=> ("testName1" "testName2")
```

```
;; Disable variables
foreach( param cadr( axlGetVars( x_mainDB ) )
        axlSetEnabled( axlGetVar( x_mainDB param ) nil ))
=> ("CAP" "R0" "R1" "LENGTH")
```

Reference

axlCreateSession, axlSetMainSetupDB, axlGetTests, axlGetTest, axlGetVars, axlGetVar

axlSetReferenceHistoryItemName

```
axlSetReferenceHistoryItemName(
     x_hsdb
     t_referenceHistoryName
     )
     => x_hsdb / 0
```

Description

Sets the reference history name for the active setup or checkpoint. You can reuse the results or netlist from the reference history during an incremental simulation run. The reference history name set using this function also appears in the *Reference* field on the Reference History toolbar.

For more details, refer to Running an Incremental Simulation in the *Analog Design Environment User Guide*.

Arguments

x_hsdb	Setup database handle to the active setup or checkpoint.
t_referenceHistoryName	Name of the reference history.

Value Returned

x_hsdb	Setup handle is returned if the history name is set successfully
0	Not successful

Examples

The following example shows how to reuse the netlist from the reference history:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlSetReuseNetlistOption(x_mainSDB t)
=> 1859
axlSetReferenceHistoryItemName(x_mainSDB "Interactive.7")
=> 1860
axlSetUseIncremental(x_mainSDB t)
=>1861
```

References

axlSetReuseNetlistOption, axlSetUseIncremental, axlGetReferenceHistoryItemName

axlSetReuseNetlistOption

```
axlSetReuseNetlistOption(
     x_hsdb
     g_value
     )
     => x_hsdb / 0
```

Description

Enables or disables the option to use the reference netlist for the active setup or checkpoint. If this option is enabled, netlist of the design is reused for the incremental run. Otherwise, the design in renetlisted.

For more details, refer to Running an Incremental Simulation in the *Analog Design Environment User Guide*.

Arguments

x_hsdb	Setup database handle to the active setup or checkpoint.
g_value	Boolean value to enable/disable the option to use the reference netlist.

Value Returned

x_hsdb	Setup handle is returned if the option to use the reference netlist is set successfully
0	Not successful

Example

The following example shows how to use the axlSetReuseNetlistOption function to reuse netlist from a reference history for a new simulation:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlSetReuseNetlistOption(x_mainSDB t)
=> 1859
axlSetReferenceHistoryItemName(x_mainSDB "Interactive.7")
=> 1860
```

References

axlSetReferenceHistoryItemName, axlSetUseIncremental, axlGetReuseNetlistOption

axlSetUseIncremental

```
axlSetUseIncremental(
     x_hsdb
     g_value
     )
     => x_hsdb / 0
```

Description

Enables or disables the setup database option in active setup or checkpoint for using reference results as cache during incremental run. This function selects or clears the *Use reference netlist* check box on the Reference History form.

For more details, refer to Running an Incremental Simulation in the *Analog Design Environment User Guide*.

Arguments

x_hsdb	Setup database handle to the active setup or checkpoint.
g_value	Boolean value to enable/disable the option to use the reference results as cache during incremental run.

Value Returned

x_hsdb	Setup handle is returned if the option to use the reference results as cache during incremental run is set successfully
0	Not successful

Example

The following example shows how to use the axlSetUseIncremental function to use the reference results as cache during an incremental run:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlSetUseIncremental(x_mainSDB t)
=>1861
axlSetReferenceHistoryItemName(x_mainSDB "Interactive.7")
=> 1860
```

References

axlSetReferenceHistoryItemName, axlSetReuseNetlistOption

axlSetScriptPath

```
axlSetScriptPath(
     x_script
     t_path
     )
     => t / nil
```

Description

Sets the path of a script.

Arguments

x_script	Script handle.
t_path	Script path.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlSetScriptPath 1045 "myData/myScripts"
t
```

axlWriteDatasheet

```
axlWriteDatasheet(
     t_axlSession
     x_historyEntry
     [ ?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 ]
     [ ?schematicDiagrams g_schematicDiagrams ]
     [ ?launchBrowser g_launchBrowser ]
     [ ?name t_name ]
     )
     => t / nil
```

Description

Creates a datasheet for the specified history entry.

Arguments

t_axlSession Name of the session.

x_historyEntry Integer value representing the history entry.

t_directory Target directory for the datasheet.
Default Value: *libName*/*cellName*/adexl/datasheets

g_resultsSummary Boolean to specify whether or not you want to print a results summary sheet containing specification sheet pass/fail table.
Default Value: t

g_testsSummary Boolean to specify whether or not you want to print a tests summary sheet containing details about the tests, sweeps, and corners.
Default Value: t

?detailedResults *g_detailedResults*

Boolean to specify whether or not you want to generate results for all points.
Default Value: t

?plots *g_plots* Boolean to specify whether or not you want to print the plots in the generated datasheet.
Default Value: t

?designVarsSummary *g_designVarsSummary*

Boolean to specify whether or not you want to generate results for all points.
Default Value: t

?paramsSummary *g_paramsSummary*

Boolean to specify whether or not you want to save the parameters summary in the generated datasheet.
Default Value: t

?cornersSummary *g_cornersSummary*

Boolean to specify whether or not you want to save the corners summary in the generated datasheet.
Default Value: t

?setupSummary *g_setupSummary*

Boolean to specify whether or not you want to save the setup summary in the generated datasheet.
Default Value: t

`?schematicDiagrams` *`g_schematicDiagrams`*

Boolean to specify whether or not you want to save the schematic diagrams in the generated datasheet.
Default Value: `t`

`?launchBrowser` *`g_launchBrowser`*

Boolean to specify whether or not you want to launch a browser window to view the generated datasheet.
Default Value: `t`

`?name` *`t_name`*

Specifies the name of the top level file and directory created for the datasheet. For example, if `?name` is set to `myDatasheet`, the names of top level file and directory are named as `myDatasheet.html` and `myDatasheet/`.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

Example 1:

```
axlGetWindowSession()
=> "session0"
axlWriteDatasheet("session0" axlGetHistoryEntry(1001 "Interactive.20"))
t
```

Example 2:

```
axlGetWindowSession()
=> "session0"
axlWriteDatasheet("session0" axlGetHistoryEntry(1001 "Interactive.20")
?resultsSummary nil ?testsSummary nil)
t
```

axlWriteDatasheetForm

```
axlWriteDatasheetForm(
     x_axlSession
     t_historyEntry
     )
     => t / nil
```

Description

Causes a form to appear so that you can specify various options for generating a datasheet.

Arguments

t_axlSession	String value representing the session name.
x_historyEntry	Integer value representing the history entry.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlGetWindowSession()
=> "session0"
axlWriteDatasheetForm("session0" axlGetHistoryEntry(1001 "Interactive.190"))
t
```

3

Variables-related Functions

The SKILL APIs described in this chapter are helpful in working with the variables associated with the setup database, corner variables and a history checkpoint. By using these functions, you can add a variable, set or get its value or set default variables.

Variables-Related SKILL Functions

Function	Description
axlGetVars	Returns a list containing a handle to the list of variables associated with the given database element (the setup database, a corner, or a history checkpoint) and a list of all variable names.
axlGetVar	Finds a variable associated with the specified database element and returns a handle to it.
axlGetVarValue	Returns value of the given variable.
axlPutVar	Creates or finds a variable by the given name for the specified database element and sets its value.
axlGetAllVarsDisabled	Returns the enabled or disabled status for inclusion of the global variables in an ADE XL simulation. In the ADE XL GUI, this is the selection status of the Global Variables check box in the Data View assistant pane.
axlSetAllVarsDisabled	Sets the selection status of the option to include the global variables in simulations. In the ADE XL GUI, this is the selection status of the Global Variables check box in the Data View assistant pane.
axlSetDefaultVariables	Creates a set of default variables in the Global Variables tree on the Data View pane and in the Parameters and Variables assistant for an ADE XL session.

Variables-Related SKILL Functions, *continued*

Function	Description
axlSetDesignVariablePerTest	Enables or disables a design variable for a specific test. In the ADE XL UI, it selects or clears the check box next to a design variable under a test.

axlGetVar

```
axlGetVar(
     x_element
     t_varName
     )
     => x_var / nil
```

Description

Finds a variable associated with the specified database element and returns a handle to it.

Arguments

x_element	Handle to the specified database element, which can be the setup database, a corner, or a history checkpoint.
t_varName	Variable name.

Value Returned

x_var	Handle to the specified variable.
nil	Unsuccessful operation.

Example

The following example code shows how to use the axlGetVar function to get the handle to a particular variable and then disable it.

```
s1 = axlGetWindowSession()
=> "session0"

x_mainSDB = axlGetMainSetupDB( s1 )
=> 1001

axlGetVars(x_mainSDB)
=> (1862
    ("IREF" "SIDDQ" "VDD" "VIN_CM" "testVar"))

var1 = axlGetVar(x_mainSDB "testVar")
=> 1952
axlRemoveElement(var1)

t
```

Note: A variable is removed from the list of global variables only if it is not a part of any test in the adexl view. If any test contains the variable being deleted, the variable is removed and created again in the Global Variables list.

References

axlGetWindowSession, axlGetMainSetupDB, `axlGetVars`

axlGetVars

```
axlGetVars(
     x_element
     )
     => l_vars / nil
```

Description

Returns a list containing a handle to the list of variables associated with the given database element (the setup database, a corner, or a history checkpoint) and a list of all variable names.

Argument

x_element	Handle to the database element, which can be the setup database, a corner, or a history checkpoint.

Value Returned

l_vars	List containing a handle to all global variables for the database element and a list of all global variable names.
nil	Unsuccessful operation.

Examples

Example 1

The following example code shows how to get the handle of the current setup database and use it to get the list of all the global variables associated with that setup database.

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB=axlGetMainSetupDB(s1)
=> 1001

axlGetVars(x_mainSDB)
=>(1862
    ("IREF" "SIDDQ" "VDD" "VIN_CM"))
```

Example 2

The following example code shows how to get the list of variables associated with the given corner, C1.

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001
axlGetCorners(x_mainSDB)
=> (1003
    ("C0_VDD_1.6_Temp" "C1_VDD_2.0_Temp" "C1_VDD_2.2_Temp" )
)
c1 = axlGetCorner(x_mainSDB "C0_VDD_1.6_Temp")
=> 1984

axlGetVars(c1)
=> (2157
    ("temperature" "VDD")
)
```

Example 3

The following example shows how to get the list of global variables that were used in a history checkpoint.

```
s1 = axlGetWindowSession()
=> "session0"

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

h1 = axlGetHistoryCheckpoint( axlGetHistoryEntry(x_mainSDB "Interactive.2"))
=> 4745

axlGetVars(h1)
=> (4788
    ("IREF" "VDD" "VIN_CM")
)
```

References

axlGetWindowSession, axlGetMainSetupDB, axlGetCorners, axlGetCorner, axlGetHistoryEntry

axlGetVarValue

```
axlGetVarValue(
     x_varHandle
     )
     => t_value / nil
```

Description

Returns value of the given variable.

Arguments

x_varHandle	Handle to a variable.

Value Returned

t_value	Value of the given variable.
error	If the handle to the variable is not valid.

Example

The following example code shows how to get value of a global variable, VDD.

```
s1 = axlGetWindowSession()
"session0"

x_mainSDB=axlGetMainSetupDB( s1 )
=>1001

axlGetVars(x_mainSDB)

=>(15615
    ("IREF" "VDD" "VIN_CM" "_sim_time" "SIDDQ")
)

v2=axlGetVar(x_mainSDB "VDD")
=>15622

v2=axlGetVarValue(v2)
=>"1.1 1.2"
```

Reference

axlGetWindowSession, axlGetMainSetupDB, axlGetVars, axlGetVar

axlPutVar

```
axlPutVar(
     x_element
     t_varName
     t_value
     )
     => x_varHandle / nil
```

Description

Creates or finds a variable by the given name for the specified database element and sets its value.

Arguments

x_element	Handle to the database element, which can be the setup database, a corner, or a history checkpoint.
t_varName	Variable name.
t_value	Variable value.

Value Returned

x_varHandle	Handle to the variable added or modified.
nil	Unsuccessful operation.

Examples

Example 1

The following example code shows how to add a new variable VDD to a corner and set sweep values for that.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>1001
c1 = axlGetCorner(x_mainSDB "C0_Temp")
=>1984
axlPutVar(c1 "VDD" "2.2 1.8")
=>2159
```

Example 2

The following example code shows how to change the value of a global variable IREF.

```
s1 = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB(s1)
=>1001
axlPutVar(x_mainSDB "IREF" "55u")
=>1863
```

Example 3

The following example code shows how to add a config sweep variable.

```
s1 = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB(s1)
=>1001
axlPutVar(x_mainSDB "CONFIG/myLib/myCell" "schematic extracted")
=>1863
```

Reference

axlGetWindowSession, axlGetMainSetupDB, axlGetCorner

axlGetAllVarsDisabled

```
axlGetAllVarsDisabled(
     x_mainSDB
     )
     => t / nil
```

Description

Returns the enabled or disabled status for inclusion of the global variables in an ADE XL simulation. In the ADE XL GUI, this is the selection status of the *Global Variables* check box in the Data View assistant pane.

Argument

x_mainSDB	Setup database handle.

Value Returned

`t`	The Global Variables check box in the Data View assistant pane is not selected.
`nil`	The Global Variables check box in the Data View assistant pane is selected.

Example

The following example code returns the status of the option to include the global variables in ADE XL simulations.

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB=axlGetMainSetupDB(s1)
=> 1001
axlGetAllVarsDisabled(x_mainSDB)
=> nil
```

Here, `nil` implies that the option to include the global variables in ADE XL simulations is enabled.

axlSetAllVarsDisabled

```
axlSetAllVarsDisabled(
     x_mainSDB
     g_enableStatus
     )
     => t / nil
```

Description

Sets the selection status of the option to include the global variables in simulations. In the ADE XL GUI, this is the selection status of the *Global Variables* check box in the Data View assistant pane.

Arguments

x_mainSDB	Setup database handle.
g_enableStatus	Option for setting the selection status.
	0 selects the *Global Variables* check box.
	1 deselects the *Global Variables* check box.

Value Returned

`t`	Successful select or deselect operation.
`nil`	Unsuccessful select or deselect operation.

Example

The following example code selects the *Global Variables* check box in the Data View assistant pane.

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB=axlGetMainSetupDB("s1)
=> 1001
axlSetAllVarsDisabled(x_mainSDB 0)
=> t
```

The following example code clears the *Global Variables* check box in the Data View assistant pane.

```
s1 = axlGetWindowSession()
=> "session0"

x_mainSDB=axlGetMainSetupDB("s1)
=> 1001

axlSetAllVarsDisabled(x_mainSDB 1)
=> t
```

axlSetDefaultVariables

```
axlSetDefaultVariables(
     l_variables
     [ t_libName ]
     )
     => t / nil
```

Description

Creates a set of default variables in the *Global Variables* tree on the Data View pane and in the Parameters and Variables assistant for an ADE XL session.

By using this function, you can define a distinct set of default variables for each library. You can also define a general set of default variables to be associated with all the libraries.

If you have set the default variables, when you open a new ADE XL setup, the program loads the set of default variables associated with the same library as the setup, if any exists. After that, it loads the generic set of default variables, if exist.

Arguments

l_variables	A list of default variables and their values.
t_libName	Library name. When you do not specify any library name, the variables are added to the list of default variables.

Value Returned

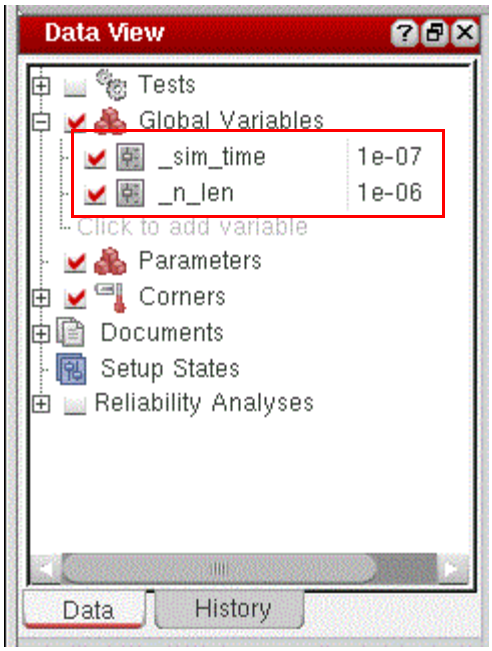
`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

If you add the following statement in .cdsinit, it adds two variables, `_n_len` and `_sim_time`, with their default values, to the `myDemoLib` library.

```
axlSetDefaultVariables( '(_n_len 1u _sim_time 100n) "myDemoLib" )
t
```

Next, when you launch Virtuoso and create a new adexl view, the default list of variables is added to the Global Variables list in the Data View snd the Variables and Parameters view, as shown in the figure below.



axlSetDesignVariablePerTest

```
axlSetDesignVariablePerTest(
     x_mainSDB
     t_varName
     t_testName
     [ ?enabled g_enableStatus ]
     )
     => t / nil
```

Description

Enables or disables a design variable for a specific test. In the ADE XL UI, it selects or clears the check box next to a design variable under a test.

Arguments

x_mainsSDB	Handle to the main setup database
t_varName	Name of the design variable
t_testName	Name of the test
?enabled *g_enableStatus*	Enable status to be set for the given design variable Default value: `t`

Value Returned

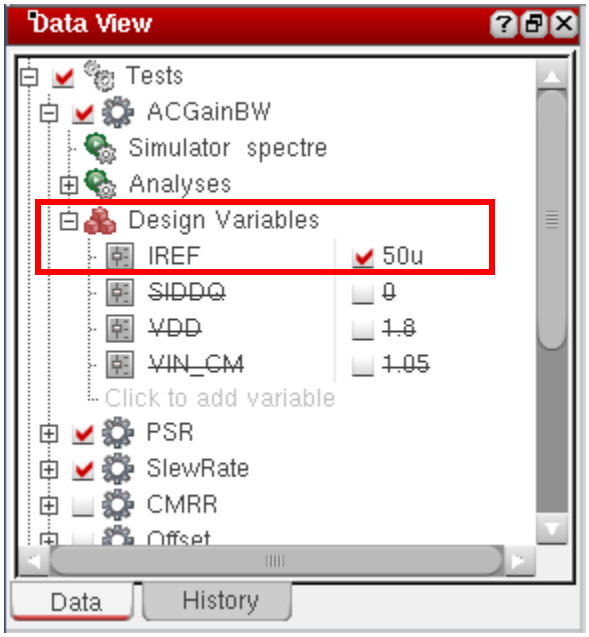
`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The example code given below shows how to enable a design variable, `IREF`, for the `ACGainBW` test.

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB( s1 )
=> 1001
axlSetDesignVariablePerTest(x_mainSDB "IREF" "ACGainBW" ?enabled t)
```

```
=> t
; The check box next to the IREF variable in the Design Variables list for the
ACGainBW test is selected.
```



Related Functions

axlGetEnabledGlobalVarPerTest

4

Parameters-related Functions

The SKILL APIs described in this chapter are helpful in working with the device parameters in the setup database.

Parameters-Related SKILL Functions

Function	Description
axlGetParameters	Returns a list of paths to all the device parameters found in the given database.
axlGetParameter	Returns the database handle to the given device parameter in the setup database.
axlGetParameterValue	Returns value of the specified device parameter in the setup database.
axlGetAllParametersDisabled	Returns the selection status of the option to include the device parameters in simulations. In the ADE XL GUI, this is the selection status of the Parameters check box in the Data View assistant pane.
axlRegisterCustomDeviceFilter	Adds a custom filter for device instance parameters on the Variables and Parameters assistant pane.
axlSetParameter	Sets a value for the specified device parameter.
axlSetAllParametersDisabled	Sets the status of the Parameters check box in the Data View assistant pane.

axlGetParameters

```
axlGetParameters(
     x_mainSDB
     )
     => l_parameterPaths / nil
```

Description

Returns a list of paths to all the device parameters found in the given database.

Argument

x_mainSDB	Handle to the setup database.

Value Returned

l_parameterPaths	List of paths to all the parameters found in the setup database.
nil	If no parameters are found.

Example

The following example code shows how to get the list of paths to all the parameters in the current session.

```
s1 = axlGetWindowSession()
=> "session0"

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

axlGetParameters(x_mainSDB)
("Two_Stage_Opamp/OpAmp/schematic/M10/l" "Two_Stage_Opamp/OpAmp/schematic/M10/m"
"Two_Stage_Opamp/OpAmp/schematic/M9/l"
)
```

axlGetParameter

```
axlGetParameter(
     x_mainSDB
     t_parameterPath
     )
     => x_parameterHandle / 0
```

Description

Returns the database handle to the given device parameter in the setup database.

Arguments

x_mainSDB	Handle to the setup database.
t_parameterPath	Complete path to the parameter *Library/Cell/View/Instance/Property*

Value Returned

x_parameterHandle	Handle to the parameter.
0	Unsuccessful, when the parameter is not found.

Example

The following example code shows how to get handle to the specified parameter and to use that handle to delete the parameter.

```
s1 = axlGetWindowSession()
=> "session0"

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

axlGetParameters(x_mainSDB)
("Two_Stage_Opamp/OpAmp/schematic/M10/l" "Two_Stage_Opamp/OpAmp/schematic/M10/m")

axlGetParameter(x_mainSDB "Two_Stage_Opamp/OpAmp/schematic/M10/l")
=> 2397

axlRemoveElement( 2397)
=> t
```

axlGetParameterValue

```
axlGetParameterValue(
     x_mainSDB
     t_parameterPath
     )
     => t_parameterValue / nil
```

Description

Returns value of the specified device parameter in the setup database.

Arguments

x_mainSDB	Handle to the setup database.
t_parameterPath	Complete path to the parameter *Library/Cell/View/Instance/Property*

Value Returned

t_parameterValue	Value of the given parameter.
nil	Setup database handle to the parameter not found.

Example

The following example code returns the value of the given parameter.

```
s1 = (axlGetWindowSession)
=> "session0"

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

axlGetParameters(x_mainSDB)
=>("Two_Stage_Opamp/OpAmp/schematic/M10/l"
"Two_Stage_Opamp/OpAmp/schematic/M10/m" "Two_Stage_Opamp/OpAmp/schematic/M6/fw")

axlGetParameterValue(x_mainSDB "Two_Stage_Opamp/OpAmp/schematic/M10/l")
=> "500n"
```

axlGetAllParametersDisabled

```
axlGetAllParametersDisabled(
     x_mainSDB
     )
     => t / nil
```

Description

Returns the selection status of the option to include the device parameters in simulations. In the ADE XL GUI, this is the selection status of the *Parameters* check box in the Data View assistant pane.

Argument

x_mainSDB	Setup database handle.

Value Returned

`t`	The *Parameters* check box in the Data View assistant pane is cleared.
`nil`	The *Parameters* check box in the Data View assistant pane is selected.

Example

The following example code returns the selection status of the *Parameters* check box in the Data View assistant pane.

```
s1 = (axlGetWindowSession)
=> "session0"

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

axlGetAllParametersDisabled(x_mainSDB)
=> nil
```

axlRegisterCustomDeviceFilter

```
axlRegisterCustomDeviceFilter(
     t_name
     s_function
     )
     => t / nil
```

Description

Adds a custom filter for device instance parameters on the Variables and Parameters assistant pane.

The default filters are *Default*, *CDF Parameters*, and *CDF Editable*. To add another filter in addition to these filters, use the axlRegisterCustomDeviceFilter function.

Arguments

t_name	Device instance parameter filter name. This is the name that appears in the *Filter* drop-down combo box on the Variables and Parameters assistant pane. Valid Values: Any string.
s_function	Symbol for a function that takes db:inst and *t_simulator* as arguments and returns a list of (property name, property value) lists.

Value Returned

t	Successful registration.
nil	Unsuccessful registration.

Example

The following example shows how to create and register a custom filter.

First, define a custom filter. For example:

```
(procedure (myCustomFilter inst simulator)
   (list
   (list "fw" (get inst "fw"))
```

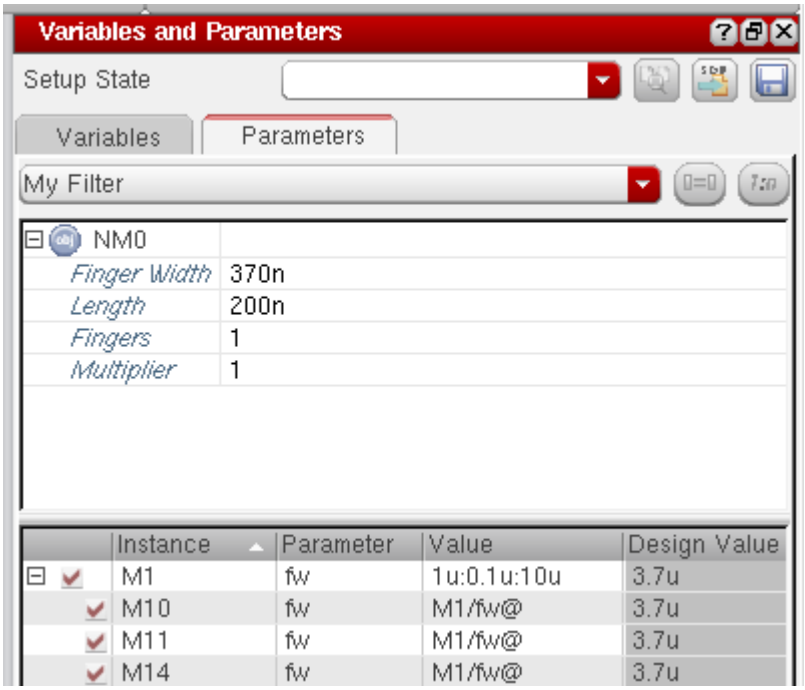
```
    (list "l" (get inst "l"))
    (list "fingers" (get inst "fingers"))
    (list "m" (get inst "m"))
    )
)
```

Then, call `axlRegisterCustomDeviceFilter` from the CIW as follows:

```
axlRegisterCustomDeviceFilter("My Filter" 'myCustomFilter)
```

The function returns `t` if the registration is successful; otherwise, `nil`.

On success, the above example adds a new filter, `My Filter`, to the Variables and Parameters assistant in the ADE (G)XL GUI. When you apply this filter, only the Finger Width, Length, Fingers, and Multiplier parameters are displayed for instances. All other parameters are filtered out. Note the filtered parameter list in the figure shown below.



axlSetParameter

```
axlSetParameter(
     x_mainSDB
     t_parameterPath
     t_value
     )
     => t / nil
```

Description

Sets a value for the specified device parameter.

Arguments

x_mainSDB	Handle to the setup database.
t_parameterPath	Complete path to the parameter *Library/Cell/View/Instance/Property*
t_value	Value to be set for the specified parameter.

Value Returned

t	The value of the device parameter is successfully set.
nil	Unsuccessful status.

Example

The following example code sets value for parameter `m` of device `M4` in the given database.

```
s1 = (axlGetWindowSession)
=> "session0"

x_mainSDB=axlGetMainSetupDB(s1 )
=> 1001

axlGetParameters(x_mainSDB)
=>("Two_Stage_Opamp/OpAmp/schematic/M10/l"
"Two_Stage_Opamp/OpAmp/schematic/M10/m" "Two_Stage_Opamp/OpAmp/schematic/M6/fw" )

axlSetParameter(x_mainSDB "Two_Stage_Opamp/OpAmp/schematic/M10/m" "2")
=> t
```

axlSetAllParametersDisabled

```
axlSetAllParametersDisabled(
     x_mainSDB
     g_enableStatus
     )
     => t / nil
```

Description

Sets the status of the *Parameters* check box in the Data View assistant pane.

Arguments

x_mainSDB	Setup database handle.
g_enableStatus	Option for setting the status of the *Parameters* check box. 0 selects the *Parameters* check box. 1 deselects the *Parameters* check box.

Value Returned

`t`	Successful select or deselect operation.
`nil`	Unsuccessful select or deselect operation.

Example

The following example code disables the usage of parameters in the ADE XL setup. This implies that it clears the *Parameters* check box in the Data View pane.

```
s1 = (axlGetWindowSession)
=> "session1"

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

axlSetAllParametersDisabled(x_mainSDB 0)
t
```

5

Model-Related Functions

The functions described in this chapter are used to work with the model files associated with the corners set in ADE XL. You can use SKILL functions to add a model file or model group to a corner, to modify their details such as their section list, test name, or block name, or to get these details for the existing model files.

Also see: Working with Model Files of ADE XL Tests

Model-Related SKILL Functions

Function	Description
axlAddModelPermissibleSectionLists	Creates a new list or adds new section names to the existing list of permissible section names for the given model imported from a PCF file. If a model file includes many sections out of which only a limited number of sections are of relevance to your testbench, you can create a permissible section list for that model file. If the LimitModelSections environment variable is set to `LimitedList`, while displaying the list of section names in the Corners Setup UI, ADE XL checks the permissible section list for a model file and shows only the relevant names. If you specify a section name that is not included in the permissible list, ADE XL shows appropriate errors.
axlGetModel	Returns a handle to the specified model file associated with the given corner.
axlGetModelBlock	Returns the block name with which the specified model is associated. By default, a model file is associated with a test and therefore, the block name is set to `Global`. However, in MTS mode, a model can be associated with a specific block in the design. Using this function, you can get the name of the block with which a model file is associated.
axlGetModelFile	Returns the model file path for the specified model.

Model-Related SKILL Functions, *continued*

Function	Description
axlGetModelGroup	Returns a handle to the specified model group in the setup database.
axlGetModelGroupName	Returns the name of the model group associated with the specified corner.
axlGetModelGroups	Returns a list of the model group names in a given setup database.
axlGetModelPermissible SectionLists	Returns a list of permissible section names for the given model extracted from a PCF file.
axlGetModelSection	Returns the model section being used for the corner with which the model file is associated.
axlGetModelSections	Returns a list of simulator sections in the specified model file.
axlGetModelTest	Returns the name of the test associated with the specified model. By default, a model file is associated with a test. However, in MTS mode, a model can be associated with a specific block in the design. Using this function, you can get the name of the test with which a model file is associated.
axlGetModels	Returns a list of model files associated with the givne corner.
axlPutModel	Adds a model file to the specified corner.
axlPutModelGroup	Adds a model group to the setup database or returns the handle to the model group if it already exists.
axlSetModelBlock	Sets the name of the block for the specified model. By default, a model is associated with all the design blocks and is set as `Global`. In the MTS mode, a model is associated with a specific MTS block, so you need to use this function to specify the name of that block.
axlSetModelFile	Sets the model file for the specified model.
axlSetModelGroupName	Sets or associates the given model group with the specified corner.

Model-Related SKILL Functions, *continued*

Function	Description
axlSetModelPermissible SectionLists	Sets a permissible section list for the given model extracted from a PCF file. If a model file includes many sections out of which only a limited number of sections are of relevance to your testbench, you can create a permissible section list for that model file. If the LimitModelSections environment variable is set to `LimitedList`, while displaying the list of section names in the Corners Setup UI, ADE XL checks the permissible section list for a model file and shows only the relevant names.
axlSetModelSection	Sets the section name for the specified model.
axlSetModelTest	Sets the name of the test to be associated with the specified model. By default, a model is associated with all the tests. In case of MTS mode, you need to associate the model with a specific test. Alternatively, you can use the axlPutModel function to specify the test name while adding a model.

axlAddModelPermissibleSectionLists

```
axlAddModelPermissibleSectionLists(
     x_handleModel
     l_sectionNames
     )
     => l_sectionHandles / nil
```

Description

Creates a new list or adds new section names to the existing list of permissible section names for the given model imported from a PCF file. If a model file includes many sections out of which only a limited number of sections are of relevance to your testbench, you can create a permissible section list for that model file. If the LimitModelSections environment variable is set to `LimitedList`, while displaying the list of section names in the Corners Setup UI, ADE XL checks the permissible section list for a model file and shows only the relevant names. If you specify a section name that is not included in the permissible list, ADE XL shows appropriate errors.

Also see: Importing Corners from Customization File in *Virtuoso Analog Design Environment XL User Guide*.

Arguments

x_handleModel	Handle to the model imported from a PCF file **Note:** This should be a model file in the main setup database.
l_sectionNames	List of section names to be added to the list of permissible sections

Value Returned

l_sectionHandles	List containing setup handles to all the permissible sections for the given model file.
nil	Unsuccessful operation.

Example

The following example code adds a new section name to the permissible section name list for the `gpdk090.scs` model imported from a PCF file:

```
; get the handle to the main setup database

sdb=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
; get the handle to the model imported from the PCF file.

;It is important to provide the handle to the main setup database in the ;function
call given below.

mh1=axlGetModel(sdb "gpdk090.scs")
=> 2323

; View the existing permissible sections list for the given model.

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ("TT_s1v"))
; the return value shows that the permissible sections list for the model
; includes only one section, TT_s1v.

axlAddModelPermissibleSectionLists(mh1 ( "FF_s1v" ))
=> (2603 2604)

; adding one more section name to the permissible sections list

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ( "TT_s1v" "FF_s1v" ))
; the return value shows that now the permissible sections list for the model
; includes two section names
```

Function References

axlGetCorner, axlGetCorners, `axlGetModel`, `axlSetModelPermissibleSectionLists`
`axlGetModelPermissibleSectionLists`

axlGetModel

```
axlGetModel(
     x_cornerHandle
     t_modelName
     )
     => x_modelFile / 0
```

Description

Returns a handle to the specified model file associated with the given corner.

Note: It is not essential for the model to be enabled for the corner.

Arguments

x_cornerHandle	Handle to a corner in the main setup database.
t_modelName	Model file name.

Value Returned

x_modelFile	Handle to the specified model file.
0	Unsuccessful operation.

Example

The following example code returns the handle to the `myModel.scs` model file associated with corner `C0`:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
cornerHandle=axlGetCorner(x_mainSDB "C0")
=>5497
modelHandle=axlGetModel(cornerHandle "myModel.scs")
=>5505
```

Function References

axlGetCorner, `axlPutModel`

axlGetModelBlock

```
axlGetModelBlock(
     x_modelHandle
     )
     => t_blockName / nil
```

Description

Returns the block name with which the specified model is associated. By default, a model file is associated with a test and therefore, the block name is set to `Global`. However, in MTS mode, a model can be associated with a specific block in the design. Using this function, you can get the name of the block with which a model file is associated.

Argument

x_model	Handle to a model.

Value Returned

t_blockName	Name of block associated with the specified model.
`nil`	Unsuccessful operation.

Example

The following example code shows how to get the block name for `mymodel.scs`:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
cornerHandle=axlGetCorner(x_mainSDB "C0")
=>5497
modelHandle=axlGetModel(cornerHandle "myModel.scs")
=>5505
axlGetModelBlock(modelHandle)
=>"Global"
```

```
; the return value indicates that the model is associated with the entire test and
not to a specific block
```

Function References

axlGetModel, axlSetModelBlock, axlGetCorner

axlGetModelFile

```
axlGetModelFile(
     x_modelHandle
     )
     => t_modelFile / nil
```

Description

Returns the model file path for the specified model.

Argument

x_model	Handle to a model.

Value Returned

t_modelFile	Model file name with full path. If the model file is imported from test, only the file name is returned.
nil	Unsuccessful operation.

Example

The following example code shows how to get the file details for a model file:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
cornerHandle=axlGetCorner(x_mainSDB "C0")
=> 1300
modelHandle=axlGetModel(cornerHandle "model1.scs")
=> 1311
axlGetModelFile(modelHandle)
=> "/hm/user/models/model1.scs"
```

Reference

axlGetModel, axlSetModelFile

axlGetModelGroup

```
axlGetModelGroup(
     x_mainSDB
     t_modelGroupName
     )
     => x_modelGroup / nil
```

Description

Returns a handle to the specified model group in the setup database.

Arguments

x_mainSDB	Handle to the setup database or a checkpoint history.
t_modelGroupName	Model group name.

Value Returned

x_modelGroup	Model group handle.
nil	Unsuccessful operation.

Example

The following example code returns a handle to the model group `mG1` in the current setup database.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
modelGrp=axlGetModelGroup(x_mainSDB "mG1")
=> 1311
```

Reference

axlPutModelGroup

axlGetModelGroupName

```
axlGetModelGroupName(
     x_cornerHandle
     )
     => t_modelGroupName / nil
```

Description

Returns the name of the model group associated with the specified corner.

Argument

x_cornerHandle	Corner handle.

Value Returned

t_modelGroupName	Names of model groups associated with the specified corner.
nil	Unsuccessful operation.

Example

The following example code shows how to get the names of model groups associated with the given corner:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
sdb=axlGetMainSetupDB(axlGetWindowSession())
cornerHandle=axlGetCorner(x_mainSDB "C1_VDD_2.0_Temp")
=> 6100
axlGetModelGroupName(cornerHandle)
"\"mG1\"  \"mG2\""
```

Reference

axlSetModelGroupName

axlGetModelGroups

```
axlGetModelGroups(
     x_mainSDB
     )
     => l_modelGroups / nil
```

Description

Returns a list of the model group names in a given setup database.

Argument

x_mainSDB	Handle to the setup database or a checkpoint history.

Value Returned

l_modelGroups	List of model group names in the setup database.
nil	Unsuccessful operation.

Example

The following example code shows how to get the list of model groups in the current setup database:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlGetModelGroups(x_mainSDB)
=>(5712 ("mG1" "mG2"))
```

Reference

axlPutModelGroup

axlGetModelPermissibleSectionLists

```
axlGetModelPermissibleSectionLists(
     x_handleModel
     )
     => l_sectionName / nil
```

Description

Returns a list of permissible section names for the given model extracted from a PCF file.

Argument

x_handleModel	Handle to the model imported from a PCF file **Note:** This should be a model file in the main setup database.

Value Returned

l_sectionNames	List containing setup handle and all the section names.
nil	Unsuccessful operation.

Example

The following example code gets the permissible section name list for `gpdk090.scs` model file:

```
; get the handle to the main setup database

sdb=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
; get the handle to the model imported from the PCF file.

;Note: It is important to provide the handle to the main setup database in the
;function call given below.

mh1=axlGetModel(sdb "gpdk090.scs")
=> 2323

; View the existing permissible sections list for the given model.

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ("TT_s1v"))
; the returned value shows that the permissible sections list for the model
; includes only one section, TT_s1v.
```

```
axlAddModelPermissibleSectionLists(mh1 ( "FF_s1v" ))
=> (2603 2604)

; adding one more section name to the permissible sections list

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ( "TT_s1v" "FF_s1v" ))
; the returned value shows that now the permissible sections list for the model
; includes two section names
```

Reference

axlSetModelPermissibleSectionLists, axlAddModelPermissibleSectionLists

axlGetModelSection

```
axlGetModelSection(
     x_modelHandle
     )
     => t_sectionName / nil
```

Description

Returns the model section being used for the corner with which the model file is associated.

Argument

x_modelHandle	Handle to the model

Value Returned

t_sectionName	Name of the model section
nil	Unsuccessful operation

Example

The following example code gets the model section for `gpdk.scs` model file:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
=> "C0"
cornerHandle=axlGetCorner(x_mainSDB "C0")
=> 1918
model = axlGetModel(cornerHandle "model1.scs")
=> 1311
axlGetModelSection(model)
=> "FF"
; the return value implies that corner C0 uses section FF of the model file,
model1.scs
```

Reference

axlGetModel, axlGetCorner, axlSetModelSection

axlGetModelSections

```
axlGetModelSection(
     t_fileName
     )
     => l_sectionNames / nil
```

Description

Returns a list of simulator sections in the specified model file.

Argument

t_fileName	Name of the model file

Value Returned

l_sectionNames	List of the model sections
nil	Unsuccessful operation

Example

The following command gets the list of model sections for `gpdk.scs` model file:

```
axlGetModelSections("./models/gpdk.scs")
=> ("FF" "SF" "FS" "SS" "TT")

axlGetModelSections("/install/cds/myModel.scs")
=> nil
; here nil is returned, which means that the command did not work. One of the reasons
for the unsuccessful operation may be that myModel.scs does not exist.
```

Reference

axlGetModel, axlGetCorner, axlSetModelSection

axlGetModelTest

```
axlGetModelTest(
     x_modelHandle
     )
     => t_testName / nil
```

Description

Returns the name of the test associated with the specified model. By default, a model file is associated with a test. However, in MTS mode, a model can be associated with a specific block in the design. Using this function, you can get the name of the test with which a model file is associated.

Argument

x_modelHandle	Handle to a model

Value Returned

t_testName	Name of the test associated with the specified model. In the default mode, the function returns `All`. In the MTS mode, it returns the name of the test of the block with which the model file is associated.
`nil`	Unsuccessful operation.

Example

The following example code shows how to get the name of the test associated with the model `model1.scs`:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
=> "C0"
cornerHandle=axlGetCorner(x_mainSDB "C0")
=> 1918
model = axlGetModel(cornerHandle "model1.scs")
=> 1311
axlGetModelTest(model)
"All"
; the return value indicates that the model file is associated with all the tests
in the adexl view.
```

Reference

axlGetModel, axlSetModelTest

axlGetModels

```
axlGetModels(
     x_cornerHandle
     )
     => l_modelFiles / nil
```

Description

Returns a list of model files associated with the givne corner.

Argument

x_cornerHandle	Corner handle.

Value Returned

l_modelFiles	List of model file names
nil	Unsuccessful operation

Example

The following example code gets the model files associated with corner `C0`:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
sdb=axlGetMainSetupDB(axlGetWindowSession())
cornerHandle=axlGetCorner(sdb "C0")
=> 1918
model = axlGetModels(cornerHandle)
(1310
   ("model1.scs" "model2.scs" "model3.scs")
)
```

axlPutModel

```
axlPutModel(
     x_cornerHandle
     t_modelFileName
     [ ?testName t_testName ]
     [ ?blockName t_blockName ]
     )
     => x_modelHandle / nil
```

Description

Adds a model file to the specified corner.

Arguments

x_cornerHandle	Handle to the corner for which the model file is to be added
t_modelFileName	Name of the model file **Note:** If the model file name already exists for another model file specified for a corner, you need to specify a unique name. For example, if `gpdk090.scs` is already present, you can specify `gpdk090.scs:1`.
`?testName` *t_testName*	Name of the test with which the model is to be associated Default value: `All` **Note:** By default, a model file is associated with all the tests. In the Multi-Technology Simulation (MTS) mode, use this argument to specify the test with which the model is to be associated.
`?blockName` *t_blockName*	Name of the design block Default value: `Global` **Note:** By default, a model file is used for the complete design. In the MTS mode, use this argument to specify the design block for which the model is to be used.

Value Returned

x_modelHandle	Handle to the model file
nil	Unsuccessful operation

Examples

Example 1

The following example adds two models from different model files for the `hightemp` corner. These models are applied to all the tests and blocks in the setup database:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
cornerHandle=axlGetCorner(x_mainSDB "C0")
; adding model
modelHandle=axlPutModel(cornerHandle "gpdk090.scs")

;adding file for the model
axlSetModelFile(modelHandle "../models/gpdk090.scs")

; specify a section to be used for the corner
axlSetModelSection(modelHandle "ff")

; enable the model for the corner
axlSetEnabled(modelHandle t)
```

Example 2

The following example shows how in the MTS mode, you can use the axlPutModel function to add a model `gpdk045.scs` for the `hightemp` corner, and apply it to the given test and block:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
cornerHandle=axlGetCorner(x_mainSDB "hightemp")
modelHandle=axlPutModel(cornerHandle "gpdk045" ?testName "MTS_test:testbench:1"
?blockName "design_45 inv")

=>1319
axlSetModelFile(modelHandle "../models/gpdk045.scs")
; specifies path to the model file
axlSetModelSection(modelHandle "ff")
axlSetEnabled(modelHandle t)
```

Reference

axlGetMainSetupDB, axlGetWindowSession, `axlSetModelFile`, `axlSetModelSection`

axlPutModelGroup

```
axlPutModelGroup(
     x_mainSDB
     t_modelGroupName
     )
     => x_modelGroup / nil
```

Description

Adds a model group to the setup database or returns the handle to the model group if it already exists.

Arguments

x_mainSDB	Handle to the setup database or checkpoint history
t_modelGroupName	Model group name.

Value Returned

x_modelGroup	Handle to the model group
nil	Unsuccessful operation

Example

The following example code adds a model group `mG1` to the current setup database.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
axlPutModelGroup(x_mainSDB "FF")
=> 1435
```

Reference

axlGetModelGroup

axlSetModelBlock

```
axlSetModelBlock(
     x_modelHandle
     t_blockName
     )
     => x_modelBlock / nil
```

Description

Sets the name of the block for the specified model. By default, a model is associated with all the design blocks and is set as `Global`. In the MTS mode, a model is associated with a specific MTS block, so you need to use this function to specify the name of that block.

Alternatively, you can use the axlPutModel function to set the block name while associating a model to a corner.

Arguments

x_modelHandle	Handle to a model.
t_blockName	Block name.

Value Returned

x_modelBlock	Handle to the model block element
`nil`	Unsuccessful operation

Examples

The following example code shows how to set a test name and a block name for a model to be associated with an MTS block:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>1001
sdb_corner=axlGetCorner(x_mainSDB "C2")
=>1098
modelHandle=axlPutModel(sdb_corner "fastModel")
=>1115
axlSetModelFile(modelHandle "../gpdk045/models/spectre/gpdk045.scs")
=>1116
axlSetModelSection(modelHandle "ff")
=>1119
;
```

```
;setting test name and block name for the design block, design_45 inv, for
;the test MTS_test:testbench:1
axlSetModelTest(modelHandle "MTS_test:testbench:1")
=>1117
axlSetModelBlock(modelHandle "design_45 inv")
=>1118
axlSetEnabled(modelHandle t)
```

Reference

axlGetModel, axlGetModelBlock, axlSetModelTest

axlSetModelFile

```
axlSetModelFile(
     x_modelHandle
     t_modelFile
     )
     => t_modelFile / nil
```

Description

Sets the model file for the specified model.

Arguments

x_modelHandle	Handle to a model
t_modelFile	Model file path and name

Value Returned

x_modelFile	Handle to the model file element
nil	Unsuccessful operation

Example

The following example code shows how to set the model file name for a model:

```
sdb=axlGetMainSetupDB(axlGetWindowSession())
=>1001
sdb_corner=axlGetCorner(sdb "C1")
=>1098
modelHandle=axlPutModel(sdb_corner "fastModel")
=>1115
axlSetModelFile(modelHandle
"/../adexl/MTS_testcase/gpdk045/models/spectre/gpdk045.scs")
=>1116
```

```
axlSetModelSection(modelHandle "fs")
=>1119
```

Reference

axlGetModel, axlGetModelFile

axlSetModelGroupName

```
axlSetModelGroupName(
     x_corner
     t_modelGroupName
     )
     => x_modelGroup / nil
```

Description

Sets or associates the given model group with the specified corner.

Arguments

x_corner	Corner handle.
t_modelGroupName	Model group name.

Value Returned

x_modelGroup	Handle to the model group name element.
nil	Unsuccessful operation.

Example

The following example code shows how to assign a model group to the given corner:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
sdb_corner=axlGetCorner(x_mainSDB "C0")
=> 7109
axlSetModelGroupName(sdb_corner "mG2")
=> 7192
```

Reference

axlGetModelGroupName

axlSetModelPermissibleSectionLists

```
axlSetModelPermissibleSectionLists(
     x_modelHandle
     l_sectionNames
     )
     => l_sectionHandles / nil
```

Description

Sets a permissible section list for the given model extracted from a PCF file. If a model file includes many sections out of which only a limited number of sections are of relevance to your testbench, you can create a permissible section list for that model file. If the LimitModelSections environment variable is set to `LimitedList`, while displaying the list of section names in the Corners Setup UI, ADE XL checks the permissible section list for a model file and shows only the relevant names.

Argument

x_modelHandle	Handle to the model imported from a PCF file **Note:** This should be a model file in the main setup database.
l_sectionNames	Names of all the sections in the given model file that are allowed to be selected.

Value Returned

l_sectionHandles	List containing handles to the permissible sections.
`nil`	Unsuccessful operation.

Example

The following example code sets a permissible section name list for the `gpdk090.scs` model imported from a PCF file:

```
; get the handle to the main setup database

sdb=axlGetMainSetupDB(axlGetWindowSession())
=> 1001
; get the handle to the model imported from the PCF file.

;Note: It is important to provide the handle to the main setup database in the
```

```
;function call given below.

mh1=axlGetModel(sdb "gpdk090.scs")
=> 2323

; View the existing permissible sections list for the given model.

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ("TT_s1v"))
; the return value shows that the permissible sections list for the model
; includes only one section, TT_s1v.

; creating a new permissible sections list for this model file

axlSetModelPermissibleSectionLists(mh1 ( "FS_s1v" "FF_s1v" ))
=> (2606 2607)
; the return value shows that now the permissible sections list for the model
; includes two section names

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ("FS_s1v" "FF_s1v"))
```

Reference

axlAddModelPermissibleSectionLists, axlGetModelPermissibleSectionLists

axlSetModelSection

```
axlSetModelSection(
     x_modelHandle
     t_sectionName
     )
     => x_modelSection / nil
```

Description

Sets the section name for the specified model.

Arguments

x_modelhandle	Model handle.
t_sectionName	Section name associated with the specified model.

Value Returned

x_modelSection	Handle to the model section element.
nil	Unsuccessful operation.

Example

The following example shows how to set the model section for a model file:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>1001
sdb_corner=axlGetCorner(x_mainSDB "C1")
=>1098
modelHandle=axlPutModel(sdb_corner "fastModel")
=>1115
axlSetModelFile(modelHandle "../gpdk045/models/spectre/gpdk045.scs")
=>1116
axlSetModelSection(modelHandle "fs")
=>1119
```

Reference

axlGetModel, axlGetModelSection

axlSetModelTest

```
axlSetModelTest(
     x_modelHandle
     t_testName
     )
     => x_modelTest / nil
```

Description

Sets the name of the test to be associated with the specified model. By default, a model is associated with all the tests. In case of MTS mode, you need to associate the model with a specific test. Alternatively, you can use the axlPutModel function to specify the test name while adding a model.

Arguments

x_modelHandle	Handle to a model.
t_testName	Name of the test associated with the specified model.

Value Returned

x_modelTest	Handle to the model test element.
nil	Unsuccessful operation.

Example

The following example code shows how to set a test name and a block name for a model to be associated with an MTS block:

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>1001
sdb_corner=axlGetCorner(x_mainSDB "C2")
=>1098
modelHandle=axlPutModel(sdb_corner "fastModel")
=>1115
axlSetModelFile(modelHandle "../gpdk045/models/spectre/gpdk045.scs")
=>1116
axlSetModelSection(modelHandle "ff")
=>1119
;
```

```
;setting test name and block name for an MTS block, design_45 inv
;
axlSetModelTest(modelHandle "MTS_test:testbench:1")
=>1117
axlSetModelBlock(modelHandle "design_45 inv")
=>1118
axlSetEnabled(modelHandle t)
```

Reference

axlGetModel, axlGetModelTest, axlSetModelBlock, axlGetCorner

Working with Model Files of ADE XL Tests

To work with the model files associated with a testbench in ADE XL, you can use the following SKILL functions:

- asiAddModelLibSelection
- asiGetModelLibSelectionList
- asiGetModelLibFile
- asiGetModelLibSection

For more details on these and related functions, refer to *Virtuoso Analog Design Environment L User Guide*.

A few examples that show how to add or view the model files associated with the ADE XL tests are given below.

Example 1

The following example code returns the model file name for the first model associated with an ADE XL test:

```
session = (axlGetWindowSession (hiGetCurrentWindow))
=> "session0"
x_mainSDB = (axlGetMainSetupDB session)
=> 1001

axlGetTests(x_mainSDB)
=> (1004
    ("opamp090:full_diff_opamp_AC:1" "opamp090:full_diff_opamp_TRAN:1")
)

testSession = axlGetToolSession(axlsession, "opamp090:full_diff_opamp_AC:1")
=> sevSession1

oSession = sevEnvironment(testSession)
=> stdobj@0x1c030668

modelList = asiGetModelLibSelectionList( oSession )
=> (("gpdk090.scs" "NN") ("fastmodel.scs" "FF"))

asiGetModelLibFile(car(modelList))
=> "gpdk090.scs"
```

Example 2

The following example code shows how to remove the existing model files for an ADE XL test, if any, and set a model file for it:

```
testName="AC"
=> "AC"

; get the handle to the ADE XL session
session=axlGetWindowSession()
=> "session0"

; get the handle to the ADE L or test session
testSession=axlGetToolSession(session testName)
=> sevSession1
oSession=sevEnvironment(testSession)
=> stdobj@0x1e213620

; remove any existing model files for the test
asiSetEnvOptionVal(oSession 'modelFiles (list (list "" "")))
=> t
(("" ""))

; add a model file to the test

asiAddModelLibSelection(oSession "testModelFile1.scs" "ss")
=> t

; you can use the asiGetModelLibSelectionList function to get the list of all
; the model files atached to the test

modelList = asiGetModelLibSelectionList( oSession )
printf("\tModel list = %L\n" modelList )
=>Model list= ("../models/spectre/gpdk045.scs" "mc") ("testModelFile1.scs" "ss") )
=> t
```

Example 3

The following example code shows how to get the details of model files associated with all the tests in ADE XL:

```
session = (axlGetWindowSession (hiGetCurrentWindow))
> "session0"

x_mainSDB = (axlGetMainSetupDB session)
> 1001

(foreach testName (cadr axlGetTests(x_mainSDB) )
    printf( "Test %s\n" testName)
    testSession = axlGetToolSession(session testName)
    oSession = sevEnvironment(testSession)
    modelList = asiGetModelLibSelectionList( oSession )
    printf("\tModel list  = %L\n" modelList )
)

>Test AC

    Model list  =
(("../adegxl/VAD_workshop_616/gpdk045_v_3_5/gpdk045/../models/spectre/gpdk045.scs
" "mc") ("testModelFile1.scs" "ss") )

Test TRAN

    Model list  = (("gpdk045.scs" "mc") ("ind.scs" "TT") ("cap.scs" "TT") )
```

6

Outputs-Related Functions

SKILL Functions for Outputs

Function	Description
ALIAS	Can be used in two scenarios- to give alias or alternate names to long net names or instance name paths, and to return the first value of a waveform result, for non-swept variables.
axlAddOutputs	Defines one or more output measures in an OCEAN script.
axlAddOutputsColumn	Defines one or more output measures in an OCEAN script.
axlAddOutputExpr	Adds an output expression to a test setup.
axlAddOutputSignal	Adds signal to a test setup.
axlDeleteOutput	Deletes output from a test setup.
axlDeleteOutputsColumn	Deletes a user-defined column from the Outputs table.
axlOutputResult	Specifies the value of an output in an OCEAN script file. This function only assigns a value to a measure, but does not add it to the script. You need to use the axlAddOutputs function to add or define output measures in an OCEAN script.
axlOutputsExportToFile	Exports outputs from the currently active setup to the specified CSV file.
axlOutputsImportFromFile	Imports outputs from the specified CSV file. Outputs can be exported to a CSV file by using axlOutputsExportToFile.
axlGetOutputUserDefinedData	Returns the value saved in a user-defined column for the given output and test name combination.
axlGetOutputNotation	Returns the notation style set for the specified output of the test.

SKILL Functions for Outputs, *continued*

Function	Description
axlGetOutputSignificantDigits	Returns the number of significant digits set for the specified output of the test.
axlGetOutputSuffix	Returns the suffix corresponding to the specified output of the test.
axlGetOutputUnits	Returns the output unit value used for displaying the results for the expression in the specified output of the test.
axlPutOutputNotation	Sets the notation style for the specified output of the test.
axlPutOutputSignificantDigits	Sets the number of significant digits corresponding to the specified output of the test.
axlPutOutputSuffix	Sets the suffix corresponding to the specified output of the test.
axlPutOutputUnits	Sets the output unit value for displaying the results for the expression in the specified output of the test.
axlGetUserDefinedOutputsColumns	Returns a list of names of the user-defined columns in the given setup database.
axlGetTemperatureForCurrentPointInRun	Within the OCEAN measurement script, this function allows to access the temperature of the current point in the run.
calcVal	Returns the value of an output of the same or another test. You can use the value returned by this function in another output expression.
axlRenameOutputsColumn	Changes the name of a user-defined column in the Outputs table.
axlSetOutputUserDefinedData	Sets value in the given user-defined column for the given test name and output.

ALIAS

```
ALIAS(
     t_globalVar
     )
     => t_globalVarValue / nil
```

Description

Can be used in two scenarios- to give alias or alternate names to long net names or instance name paths, and to return the first value of a waveform result, for non-swept variables.

For the second purpose, ALIAS provides a wrapper around the VAR function to output a scalar value by returning the first Y value from the VAR waveform. Therefore, you can replace VAR with ALIAS where VAR returns a waveform but you need a scalar value for an output or expression. ALIAS returns the first Y value of the waveform returned by VAR.

Important

You cannot use ALIAS function in the following cases:

- In pre-processing. It can be used only for the processed simulation results.
- To access sweep variables. It returns an incorrect value for sweep variables.

Argument

t_globalVar	Name of the global variable.

Value Returned

t_globalVarValue	Value of the global variable.
nil	Unsuccessful operation.

Examples

Example 1:

The following example demonstrates how to use a net/terminal name with an alias name:

```
mynet =”/I0/I1/M1/M2/M3/net2”
db(vh('hbac ALIAS("mynet") '((0 -1))))
```

Functions such as `VT`, `VF`, or `VH` expect a net/terminal name, and perform unsuccessfully with VAR. Instead, it is recommended to use ALIAS with such functions.

Example 2:

The following example demonstrates when to use ALIAS within a calculator function to provide a single scalar value for outputs that return a waveform:

```
X_Freq = 1M
value(db(vh('hbac "/X4" '((0 -1))))  ALIAS("X_Freq"))
```

Note: The value function expects a scalar value in the second argument. If we use VAR("`X_Freq`"), it returns a scalar value when evaluating at the leaf level, but a waveform at the root level, where all the values will be `1M`. Therefore, it is recommended to use ALIAS because it returns a scalar at the leaf level, and the first value from the waveform, when evaluating at the root level.

When to avoid using ALIAS?

In example2 shown above, if the `X_Freq` variable is swept with three different values, as shown below, ALIAS returns the value, `1M`:

```
X_Freq = 1M 2M 3M
value(db(vh('hbac "/X4" '((0 -1))))  ALIAS("X_Freq"))
```

Here, using ALIAS("`X_Freq`")would only provide the first value from the swept variable, which would be incorrect.

Therefore, it is strongly recommended not to use ALIAS with sweep variables, as it returns incorrect result values.

axlAddOutputs

```
axlAddOutputs(
     l_outputNames
     )
     => t / nil
```

Description

Defines one or more output measures in an OCEAN script.

Note: Prior to IC6.1.5 release, it was mandatory to specify this command on the first line of the script file. Starting IC6.1.5, this requirement has been removed. In addition:

- It is now optional to specify the `axlAddOutputs` command. ADE XL parses the script for axlOutputResult commands to extract derived measure names.
- You can specify this command anywhere in the script

Argument

`l_outputNames`	List of output names.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following command defines two outputs, `maxOfOut` and `minOfOut`. You can set the values of these outputs using axlOutputResult.

```
axlAddOutputs( '("maxOfOut" "minOfOut"))
t
```

Reference

axlOutputResult

axlAddOutputsColumn

```
axlAddOutputsColumn(
     x_mainSDB
     t_ColumnName
     )
     => t / nil
```

Description

Adds a new user-defined column to the ADE XL Outputs table.

Arguments

x_mainSDB	Handle to the main setup database
t_columnName	Name of the user-defined column to be added to the Outputs table

Value Returned

`t`	Successful addition of the specified column
`nil`	Unsuccessful operation

Example

The following example demonstrates how to add a user-defined column to the Outputs table.

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlAddOutputsColumn( x_mainSDB "Spec Description")
=>t
```

axlAddOutputExpr

```
axlAddOutputExpr(
     t_sessionName
     t_testName
     t_outputName
     [ ?expr t_expr ]
     [ ?evalType t_evalType ]
     [ ?exprDPLs l_exprDPLs ]
     [ ?plot g_plot ]
     [ ?save g_save ]
     )
     => t / t_error
```

Description

Adds an output expression to a test setup.

Arguments

t_sessionName	Name of session
t_testName	Name of test
t_outputName	Name to be assigned to the expression output
?expr *t_expr*	Expression to be used to calculate the output Default value: ""
?exprDPLs *l_exprDPLs*	A disembodied property list that provides details of expressions to be added in batch mode. Each output structure can be specified in the following format: `(nil 'outputName value 'expr value 'evalType value 'plot value 'save value)` Default value: `nil`
?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.
?plot *g_plot*	Specifies if the output is to be plotted Default value: `t`
?save *g_save*	Specifies if the output is to be saved

Value Returned

`t`	Successful addition of output to the test

t_error	If unsuccessful, returns an error message

Example

Example 1

The following example code shows how to use the axlAddOutputExpr function to add outputs for a test:

```
session = axlGetWindowSession()
=>"session0"
; returns handle to the current session

axlAddOutputExpr(session "AC" "output1" )
=> t
; the above statement adds an expression output, but the expression to be evaluated
; is not specified. The evaluation type of this output is 'point'

axlAddOutputExpr(session, "AC" "SRp" ?expr "ymax(deriv(VT(\"/OUT\")))" ?evalType
"corners" ?plot t ?save t)
; the above statement adds an expression output that evaluates the given expression
; across corners
```

Example 2

The following example code shows how to use the axlAddOutputExpr function to add outputs in batch mode:

```
session = axlGetWindowSession()
; gets a handle to the session

;; define a DPL for the first output to be added
dpl='(nil)
putprop(dpl "BW" 'outputName)
putprop(dpl "bandwidth(VT(\"/out\") 3 \"low\")" 'expr)
putprop(dpl t 'plot)
; append it to exprs
exprs=append(nil list(dpl))

; define a DPL for the second output to be added

dpl1='(nil)
putprop(dpl1 "Gain" 'outputName)
putprop(dpl1 "value(dB20(mag(VF(\"/OUT\))))" 'expr)
putprop(dpl1 t 'plot)
; append it to exprs

exprs=append(exprs list(dpl1))

; add the outputs
axlAddOutputExpr(session "opamp090:full_diff_opamp_AC:1" "" ?exprDPLs exprs)
;; this script adds two outputs, BW and Gain, to the ADE XL setup
```

axlAddOutputSignal

```
axlAddOutputSignal(
     t_sessionName
     t_testName
     t_signalName
     [ ?type t_outputType ]
     [ ?plot g_plot ]
     [ ?save g_save ]
     )
     => t / t_error
```

Description

Adds signal to a test setup.

Arguments

t_sessionName	Name of session.
t_testName	Name of test.
t_signalName	Name of the signal to be added to the outputs of the given test.
?type *t_outputType*	Type of the signal. Possible values: `terminal`, `net` Default value: `net`
?outputName *t_outputName*	Name to be assigned to the signal output. Default value: `""`
?plot *g_plot*	Specifies if the output is to be plotted. Default value: `t`
?save *g_save*	Specifies if the output is to be saved. Default value: `t`

Value Returned

`t`	Successful addition of signal output to the test.
`nil`	If unsuccessful, returns an error message.

Example

```
session = axlGetWindowSession()
testname = "voltage_divider:voltage_divider:1"
axlAddOutputSignal(session testname "/net1")
t
axlAddOutputSignal(session testname "V0/PLUS" ?type "terminal" ?plot t )
t
axlAddOutputSignal(session testname "/net2" ?outputName "Out1" ?type "net")
t
```

axlDeleteOutput

```
axlDeleteOutput(
     t_sessionName
     t_testName
     t_outputName
     [ ?type t_outputType ]
     )
     => t / t_error
```

Description

Deletes output from a test setup.

Arguments

t_sessionName	Name of session.
t_testName	Name of test.
t_outputName	Output to be deleted.
?type *t_outputType*	Type of the output.
	Possible values: `signal`, `expr`
	Default value: `""`

Value Returned

`t`	Successful deletion of an output from an ADE XL test.
t_error	If unsuccessful, returns an error message.

Example

```
session = axlGetWindowSession()
testname = "simLib1:sim_top1:1"
axlDeleteOutput(session testname "/net6")
axlDeleteOutput(session testname "V0/I/1" ?type "expr")
axlDeleteOutput(session testname "/net5" ?type "signal")
t
```

axlDeleteOutputsColumn

```
axlDeleteOutputsColumn(
     x_mainSDB
     t_columnName
     )
     => t / nil
```

Description

Deletes a user-defined column from the Outputs table.

Arguments

x_mainSDB	Handle to the main setup database
t_columnName	Name of the user-defined column to be deleted

Value Returned

`t`	Successful deletion of the specified column
`nil`	Unsuccessful operation

Example

The following example demonstrates how to delete a user-defined column, `Comments`.

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlGetUserDefinedOutputsColumns(x_mainSDB)
=>("Comments" "Spec Description")
axlDeleteOutputsColumn( x_mainSDB "Comments")
=>t
```

axlOutputResult

```
axlOutputResult(
     g_value
     [ t_outputName ]
     )
     => t / nil
```

Description

Specifies the value of an output in an OCEAN script file. This function only assigns a value to a measure, but does not add it to the script. You need to use the axlAddOutputs function to add or define output measures in an OCEAN script.

Note: It is recommended to initialize the output values before calling this function.

Arguments

g_value	Output value
t_outputName	Name of the output

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

The following OCEAN script sets the value of the `maxOfOut` output to 110 and `minOfOut` to 0.

```
$ cat myMeas.ocn
  aVar = 55
  axlOutputResult( aVar*2  "maxOfOut" )
  axlOutputResult( 0 "minOfOut" )
```

For details on how to load an OCEAN script file for a measurement, refer to Loading an OCEAN or a MATLAB Measurement.

Reference

axlAddOutputs

axlOutputsExportToFile

```
axlOutputsExportToFile(
     t_session
     t_fileName
     [ ?omitTestCol g_omitTestCol ]
     )
     => t / nil
```

Description

Exports outputs from the currently active setup to the specified CSV file.

Arguments

t_session	Name of the currently active session.
t_fileName	Name of the CSV file to which outputs are to be exported.
?omitTestCol *g_omitTestCol*	Specifies if the *Test* column is to be included in the exported output details. If the *Test* column is not included, the outputs can be reused for a different test. In the later case, you need to explicitly specify the test name while importing outputs.

Value Returned

t	Successful operation
nil	Unsuccessful operation

Example

The following example exports outputs from the active session to a CSV file `ACoutputs.csv`.

```
axlOutputsExportToFile(session0 "ACoutputs.csv" ?omitTestCol t)
```

Note: In this example, the test column is not exported.

axlOutputsImportFromFile

```
axlOutputsImportFromFile(
     t_session
     t_fileName
     [ ?operation operationType ]
     [ ?test testName ]
     )
     => t / nil
```

Description

Imports outputs from the specified CSV file. Outputs can be exported to a CSV file by using axlOutputsExportToFile.

Arguments

t_session Name of the currently active session.

t_fileName Name of the CSV file from which outputs are to be imported.

?operation *t_operationType*

Specifies how to use the imported outputs. This argument can take any one of the following three possible values:

- `overwrite`: Overwrites the existing set of outputs that are already defined with the same name as that of an output being imported.
- `retain`: Retains the existing set of outputs when the name of an existing output matches with that of an output being imported.
- `merge`: Keeps the existing set of outputs and merges them with the outputs imported from the specified CSV file.

Default value: `overwrite`

`?test` *`t_testName`*	Name of the test for which the imported outputs are to be used. **Note:** Specify this argument only when the test name is not saved in the CSV file being imported. If the CSV file contains the test column, each output is imported for the specified test. Default value: ""

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

The following example imports all the outputs from ACoutputs.csv and merges them with the existing set of outputs.

```
axlOutputsImportFromFile(session0 "ACoutputs.csv")
t
```

The following example imports all the outputs from outputs.csv for the ACGain test and overwrites the existing set of outputs.

```
axlOutputsImportFromFile(session0 "outputs.csv" ?operation "overwrite" ?test
"ACGain")
t
```

axlGetOutputUserDefinedData

```
axlGetOutputUserDefinedData(
     x_mainSDB
     t_testName
     t_OutputName
     t_ColumnName
     )
     => t_columnValue / nil
```

Description

Returns the value saved in a user-defined column for the given output and test name combination.

Arguments

x_mainSDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputName	Name of an output
t_columnName	Name of a user-defined column

Value Returned

t_columnValue	Value saved in the given column
`nil`	If there is no value saved in the column

Example

The following example demonstrates how to display value for output `UGF` in the `Spec Description` user-defined column.

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlGetOutputUserDefinedData(x_mainSDB "ACGainBW" "UGF" "Spec Description")
=>"UGF > 1.5M"
```

axlGetOutputNotation

```
axlGetOutputNotation(
     x_sdb
     t_testName
     t_outputID
     )
     => t_notation / " "
```

Description

Returns the notation style set for the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output

Value Returned

t_notation	Notation style of the specified output
`""`	Notation style is not set for the specified output

Example

The following example shows that `suffix` is the notation style for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlGetOutputNotation(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF")
=>"suffix"
```

axlGetOutputSignificantDigits

```
axlGetOutputSignificantDigits(
     x_sdb
     t_testName
     t_outputID
     )
     => t_digits / 0
```

Description

Returns the number of significant digits set for the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output

Value Returned

t_digits	Number of the significant digits of the specified output
0	Significant digits are not set for the specified output

Example

The following example shows that `9` is set as the significant digits for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlGetOutputSignificantDigits(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF")
=>"9"
```

axlGetOutputSuffix

```
axlGetOutputSuffix(
     x_sdb
     t_testName
     t_outputID
     )
     => t_suffix / " "
```

Description

Returns the suffix corresponding to the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output

Value Returned

t_suffix	Suffix of the specified output
`""`	Suffix is not set for the specified output

Example

The following example shows that `G` is the suffix of the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlGetOutputSuffix(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF")
=>"G"
```

axlGetOutputUnits

```
axlGetOutputUnits
     x_sdb
     t_testName
     t_outputID
     )
     => t_units / " "
```

Description

Returns the output unit value used for displaying the results for the expression in the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output

Value Returned

t_units	Unit of the specified output
`""`	Unit is not defined for the output

Example

The following example shows that the output unit is defined in `hz` for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlGetOutputUnits(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF")
=>"Hz"
```

axlPutOutputNotation

```
axlPutOutputNotation(
     x_sdb
     t_testName
     t_outputID
     t_value
     )
     => x_sdb / nil
```

Description

Sets the notation style for the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output
t_value	Notation style to be set for the specified output Valid values are: ■ `default`: Displays results using the default notation style specified in the Default Formatting Options form ■ `eng`: Displays results in the engineering notation ■ `sci`: Displays results in the scientific notation ■ `suffix`: Displays results in the suffix notation

Value Returned

x_sdb	Returns the handle to the output database when the notation style is successfully set
`nil`	Notation style was not set

Example

The following example shows that `suffix` is being set as the notation style for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlPutOutputNotation(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF" "suffix")
=>5141
```

axlPutOutputSignificantDigits

```
axlPutOutputSignificantDigits(
     x_sdb
     t_testName
     t_outputID
     x_value
     )
     => x_sdb / nil
```

Description

Sets the number of significant digits corresponding to the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output
x_value	The digits to be set for the specified output Valid values are `2` to `15`

Value Returned

x_sdb	Returns the handle to the output database when the number of significant digits is successfully set
`nil`	The significant digits is not set

Example

The following example shows that `9` is being set as the significant digit for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlPutOutputSignificantDigits(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF" 9)
=>5141
```

axlPutOutputSuffix

```
axlPutOutputSuffix(
     x_sdb
     t_testName
     t_outputID
     t_value
     )
     => x_sdb / nil
```

Description

Sets the suffix corresponding to the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output
t_value	Suffix to be set for the specified output

Value Returned

x_sdb	Returns the handle to the output database when the suffix is successfully set.
`nil`	Suffix is not set.

Example

The following example shows that `G` is being set as the suffix for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlPutOutputSuffix(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF"  "G")
=>5141
```

axlPutOutputUnits

```
axlPutOutputUnits(
     x_sdb
     t_testName
     t_outputID
     t_value
     )
     => x_sdb / nil
```

Description

Sets the output unit value for displaying the results for the expression in the specified output of the test.

Arguments

x_SDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputID	Name of an output
t_value	Unit value to be set for the specified output

Value Returned

x_sdb	Returns the handle to the output database when the output unit value is successfully set.
`nil`	Unit value is not set.

Example

The following example shows that `Hz` is being set as the unit for the `UGF` output of the `opamp:OpAmp_lab1AC_top:1` test.

```
axlPutOutputUnits(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF" "Hz")
=>5141
```

axlGetUserDefinedOutputsColumns

```
axlGetUserDefinedOutputsColumns(
     x_mainSDB
     )
     => l_columnNames / nil
```

Description

Returns a list of names of the user-defined columns in the given setup database.

Argument

x_mainSDB	Handle to the main setup database

Value Returned

l_columnNames	List of user-defined column names
nil	No user-defined columns found

Example

The following example demonstrates how to get a list of user-defined columns:

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlGetUserDefinedOutputsColumns(x_mainSDB)
=>("Comments" "Spec Description")
```

axlGetTemperatureForCurrentPointInRun

```
axlGetTemperatureForCurrentPointInRun(
     )
     => t_temperature / nil
```

Description

Within the OCEAN measurement script, this function allows to access the temperature of the current point in the run.

Arguments

None

Value Returned

t_temperature	Returns the temperature of the current point in the run.
nil	Returns nil otherwise.

Example

```
axlGetTemperatureForCurrentPointInRun()
-> "27"
```

calcVal

```
calcVal(
     t_outputName
     [ t_testName ]
     [ ?cornerName t_cornerName ]
     [ ?historyName t_historyName ]
     [ ?run t_runName ]
     [ ?result t_resultName ]
     [ ?getFirstSweepPoint t_getFirstSweepPoint ]
     [ ?matchParams g_matchParams ]
     [ ?defaultVal t_defaultVal ]
     )
     => g_output / nil
```

Description

Returns the value of an output of the same or another test. You can use the value returned by this function in another output expression.

Arguments

Important

The `?run` argument is used only for a cellview created using ADE Assembler.

t_outputName Name of the output to be used. The output can return a scalar value or waveform.

t_testName (Optional) Name of test name to which the given output belongs. When not specified, the name of the current test is used.

Note: You do not need to provide this argument when using the `calcVal` expression for global variables.

`?cornerName` *t_cornerName*

Name of the corner. When specified, the value of the given output for this corner is used.

Note: Name of a corner group cannot be used in this argument.

`?historyName` *t_historyName*

Name of the history from which the results are to be retrieved. When specified, the scalar or waveform result of the output from the given history is used. Otherwise, the value of the output from the current simulation run is used.

Note: The history must exist in order for `calcVal` to return the expected result. To retrieve the value of an output of type 'signal', the simulation waveform results must be saved.

`?run` *`t_runName`*

Name of the run plan from which the value of the given output is to be returned. This argument is useful only when the setup has a run plan.

Use this argument to get the value of an output from the results of one run (in the run plan) and use it in a variable, expression, or run condition for another run in the same run plan or history. It cannot be used to refer to the results of any other history. When both `?run` and `?historyName` are specified, `?historyName` is ignored by `calcVal`.

Also see: Using Results of One Run in Another

`?result` *`t_resultName`*

Name of an analysis from which the `signal` type output is to be to retrieved. When specified, returns a waveform for the corresponding output of that analysis. Use this argument only for the outputs of type `signal`.

Note: The specified analysis must exist in the test along with the simulation waveform results in order for `calcVal` to return the expected result.

`?getFirstSweepPoint` *`t_getFirstSweepPoint`*

Specifies that when sweep values are used, the result of only the first swept point is to be used for calculation.

If `?cornerName` is specified with this argument, the function retrieves the output value of the first swept point for the given corner. Otherwise, calcVal retrieves the results for all the corners set up for the test.

Default value: `nil`

`?matchParams`
`t_matchParams`

Specifies the requirements to look for points that have matched variables and parameters in the current run and the source run. The source run can be of any other test in the current history, or the same or another test in another history specified using `?historyName`.

Possible values:

- `"none"`: (Default value) Does not look for matched variables and parameters to copy the result value. Instead, it uses the result value for the current data point if all data points in the source and target tests are same. Otherwise, `calcVal` returns `nil`. This is equivalent to not setting `?matchParams`.
- `"all"`: Matches each variable or parameter value in the current test with the variables and parameters of the source test. If an identical point is found, the function returns the required output value for that point. If no point with the same number and values of variables and parameters is found, or more than one matching point are found, the function returns `nil`.
- A list containing a list of name-value pairs of the parameters to be matched: Compares the given set of variables and parameters. The list is given in the `list(list(<param1> <value1>) list(<param2> <value2>) ...)` format, where `<valueN>` can be a constant value or it can be returned by a user-defined function. If the same variables and parameters are found with the given values in both source and target point, the function returns the value of the given output. Otherwise, if no or more than one matching point are found, the function returns `nil`.

Important Points to Note

- This argument considers the corners and sweep parameters in the same way.
- If the `?getFirstSweepPoint` argument is set to `t`, only the first parameter is compared.
- Do not use `?matchParams` as `"none"` when the tests contain local sweeps.

`?defaultVal` *`t_defaultVal`*

The default value to be returned if `calcVal` fails for any reason.

This is helpful when `calcVal` is used in an expression for a dependent output measurement,or in the input value for a test variable.

Value Returned

`g_output`	Returns the value of the output.
`nil`	Returns `nil` if the function fails and the default value is not specified.

Examples

Example 1:

In the following example, `avg_vt` represents the expression `average(VT("/out"))` and `myTest` is the ADE XL test name:

```
calcVal("avg_vt" "myTest")
```

Example 2:

You can compute the value of an expression with reference to another expression from a different test, as shown below.

```
calcVal("output1" "test1")/calcVal("output2" "test3")
```

Example 3:

You can also use the output value for a corner, as shown below.

```
calcVal("Gain" "AC" ?cornerName "C2")
```

Example 4:

You can also use the output value from a history, as shown below.

```
calcVal("Gain" "AC" ?historyName "Interactive.0")
```

Example 5:

You can use the waveform from the result, as shown below.

```
calcVal("/OUT" "AC" ?result "ac")
=> srrWave:0x40421090
```

Example 6:

If you want to retrieve the value from only the first sweep point, you can use the `?getFirstSweepPoint` argument. The following example will always return the first sweep point value of `myCalib` in the test `calib_tb`. This is useful if you run a sweep on the test where the `calcVal` expression is used but the source of the value `myCalib` is not swept.

```
calcVal("myCalib" "calib_tb"  ?getFirstSweepPoint t)
```

Example 7:

You can specify a default value that can be used when the result for the specified output is not found.

```
calcVal("Gain" "AC" ?cornerName "C2" ?defaultVal "1000")
```

Example 8:

The default value can also be used to show an error message when the result for the corner is not available.

```
calcVal("Gain" "AC" ?cornerName "C2" ?defaultVal "ERROR: Gain for C2 is not
available")
```

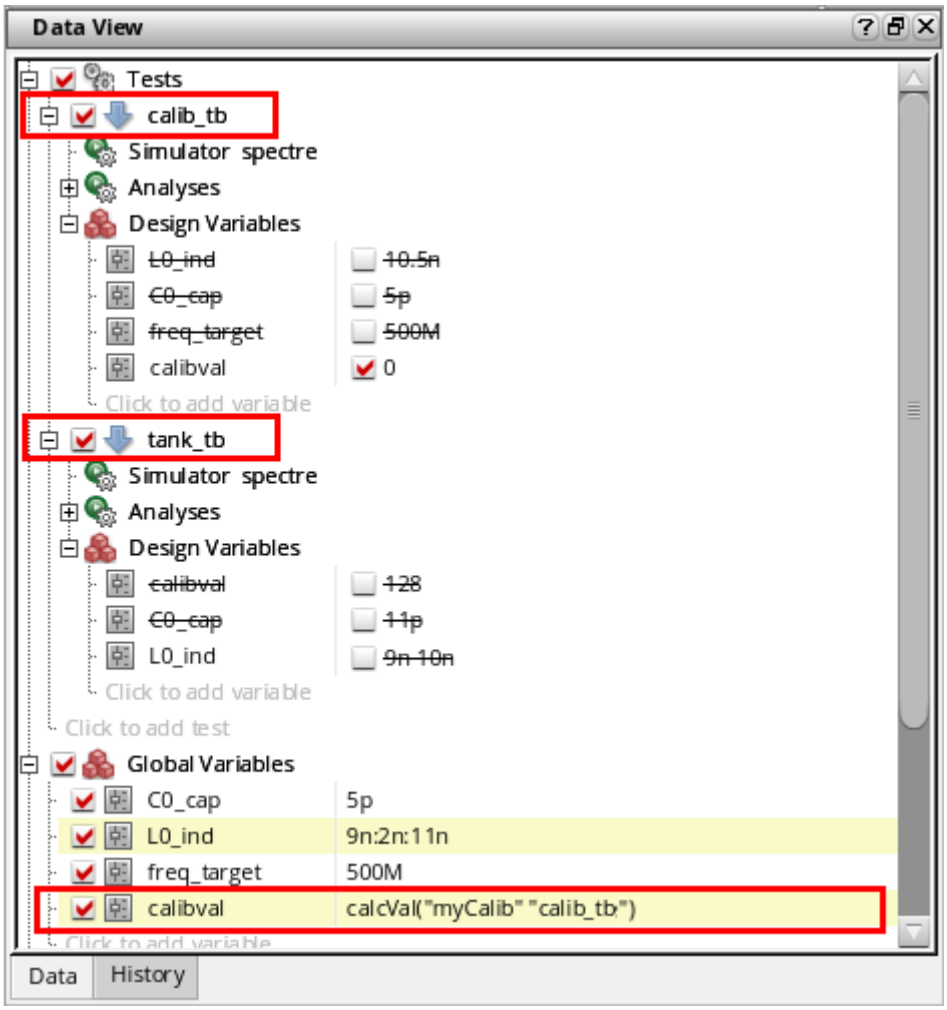
Example 9:

You can retrieve the results from a nominal run. The following example would give an output, which is the ratio of the measured frequency of the specified corner to the frequency in the nominal run:

```
calcVal("Freq")/calcVal("Freq" ?cornerName "Nominal")
```

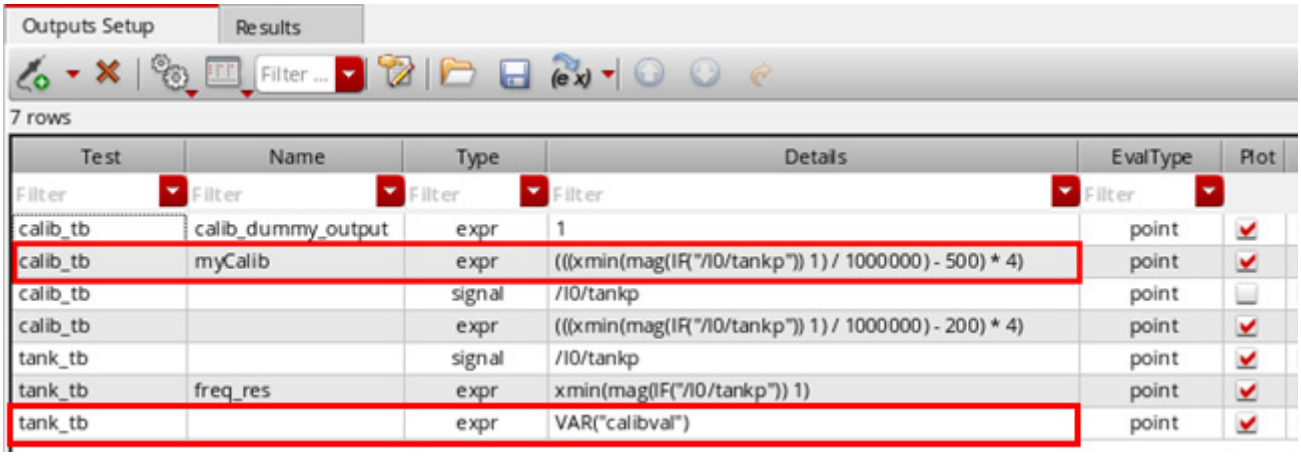
Example Using ?matchParams

The following examples to show how you can use the `?matchParams` argument of `calcVal`. The base setup used in these examples is shown below.



This setup contains two tests, `calib_tb` and `tank_tb`.

If not overriding, these tests use four global variables: `C0_cap`, `L0_ind`, `freq_target`, and `calibval`. `calibval` contains an expression that uses the value of the `myCalib` output defined for the test `calib_tb`.

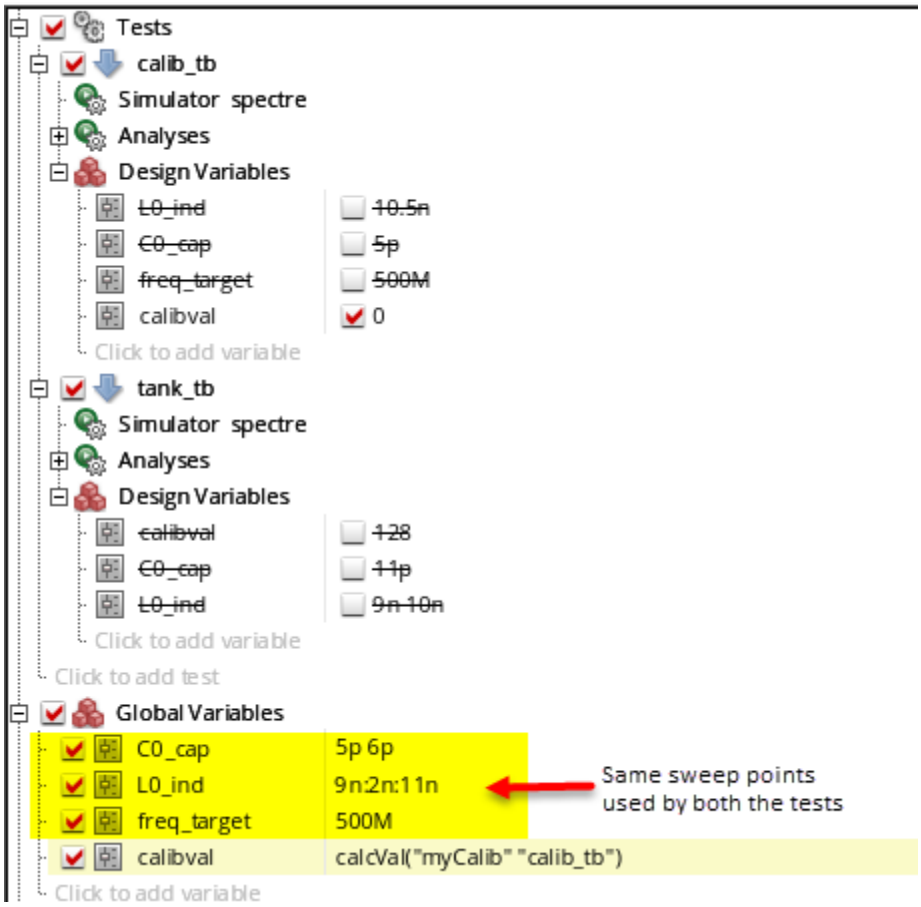


Test	Name	Type	Details	EvalType	Plot
calib_tb	calib_dummy_output	expr	1	point	✓
calib_tb	myCalib	expr	(((xmin(mag(IF("/I0/tankp")) 1) / 1000000) - 500) * 4)	point	✓
calib_tb		signal	/I0/tankp	point	
calib_tb		expr	(((xmin(mag(IF("/I0/tankp")) 1) / 1000000) - 200) * 4)	point	✓
tank_tb		signal	/I0/tankp	point	✓
tank_tb	freq_res	expr	xmin(mag(IF("/I0/tankp")) 1)	point	✓
tank_tb		expr	VAR("calibval")	point	✓

The `VAR("calibval")` output of test `tank_tb` returns the value of the `calibval` global variable, which saves the result of the `calcVal` function.

In the following example cases, you can observe the difference in the value of `VAR("calibval")` for different values of `?matachParam`, and varying variables and parameters:

Case I: `?matchParams` is set to the default value (`"none"`), the number of data points in the calibrated and target tests is same, and the swept variables and corners are identical

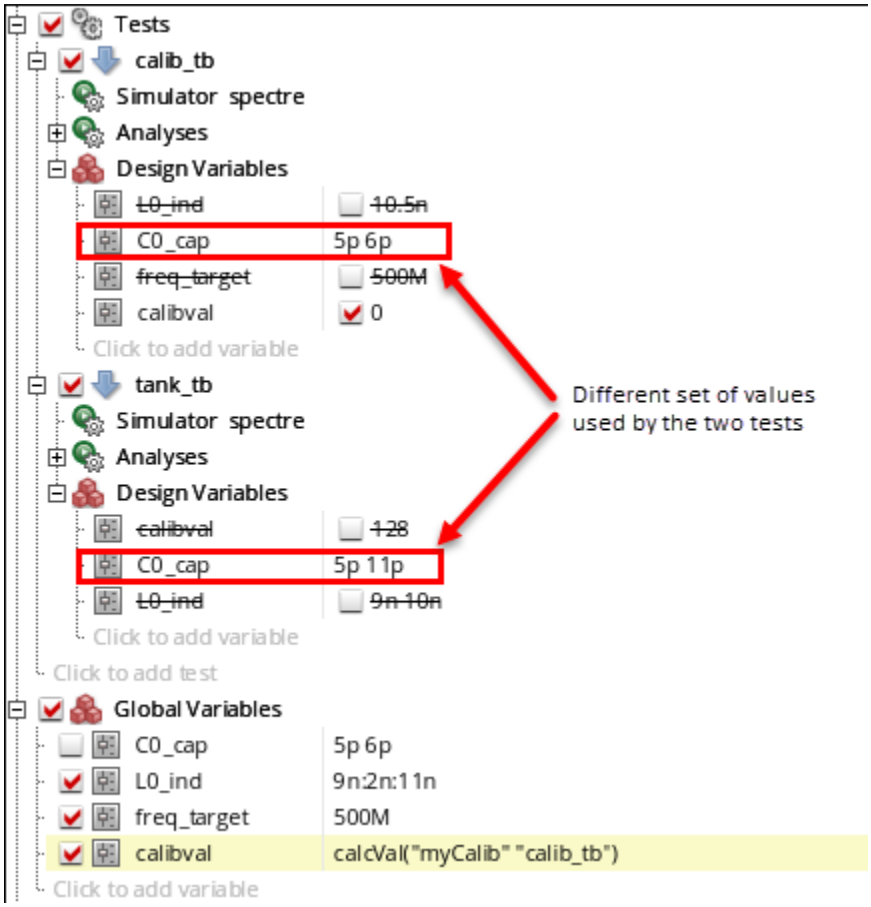


For each data point of tank_tb, there will be a valid value of the `calibval` global variable. Thus, `calcVal` returns a valid, non-nil value.

`calib_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
`myCalib`	198	4	99	-85
`tank_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
`calibval`	198	4	99	-85

Note: In the case described above, the return value would be same if `?matchParams` is set to `"all"` because the count and values of all variables and parameters match.

Case II: `?matchParams` is set to the default value (`"none"`), the number of data points in the calibrated and target tests is same, but the values are different



In this example, `?matchparams` is set to `"none"`, but `calcVal` does not look for any matching variable because the setup contains local sweeps. As a result, for each point of tank_tb, `calcVal` returns `nil`.

`calib_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
`myCalib`	198	4	99	-85
`tank_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=11p L0_ind=9n	C0_cap=11p L0_ind=11n
`calibval`	nil	nil	nil	nil

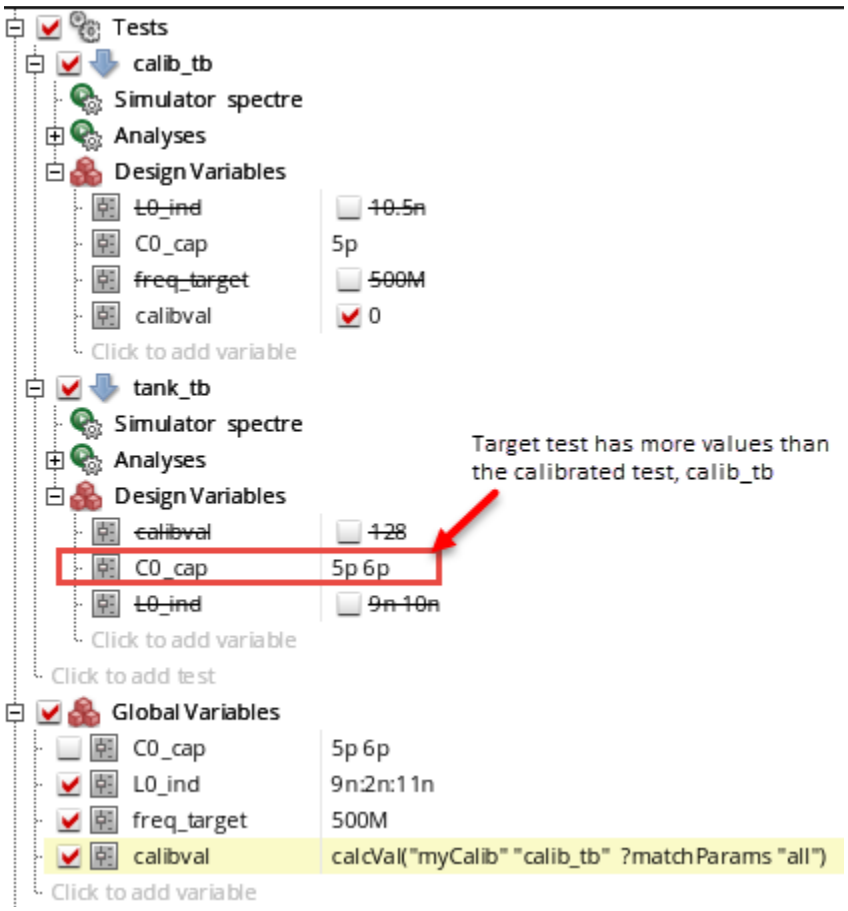
Important

In the case shown above, `calcVal` does not return valid results. Therefore, it is recommended to use `?matchParams` set to `"none"` only when the tests do not have local sweeps, the count of the enabled sweep points and corners in the source and target test is same and they are identical.

In case II described above, if `?matchparams` is set to `"all"` instead of `"none"`, `calcVal` looks for matching variables and parameters in other points, and the results are different, as shown below.

`calib_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
`myCalib`	198	4	99	-85
`tank_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=11p L0_ind=9n	C0_cap=11p L0_ind=11n
`calibval`	198	4	nil	nil

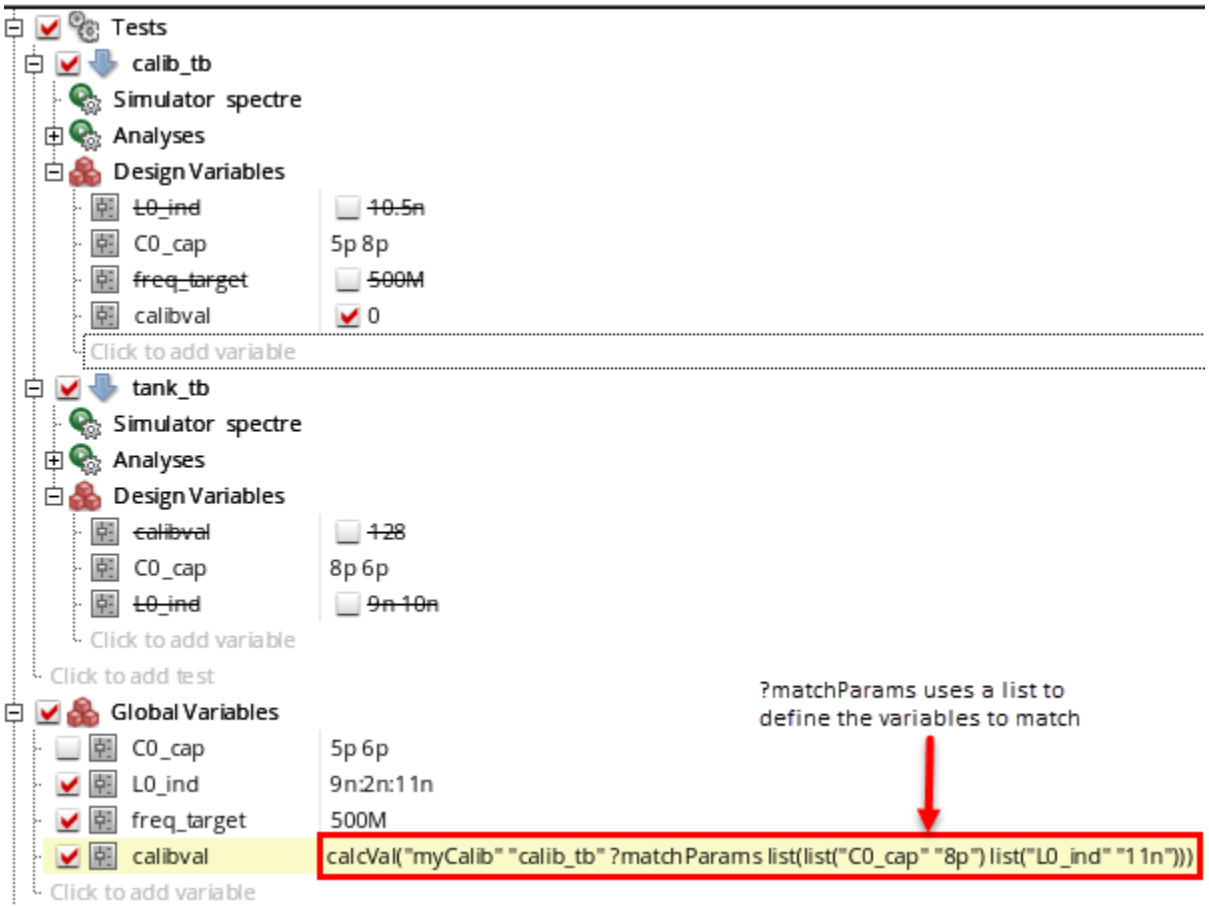
Case III: `?matchParams` is set to `"all"` and the target test has more data points than the calibrated test



`calcVal` does not find matching data points for some parameters. For those points, it returns `nil`, as shown below.

`calib_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n		
`myCalib`	198	4		
`tank_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
`calibval`	198	4	nil	nil

Case IV: `?matchParams` is set to `list(list("L0_ind" "11n") list("C0_cap" "8p"))`



In this case, `calcVal` looks for data points in the calibrated test that have parameters `L0_ind=11n` and `C0_cap=5p`. If found, the target test uses the value of the `myCalib` expression to get the value for `calibval` for all the points in the target test, as shown below.

`calib_tb`	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=8p L0_ind=9n	C0_cap=8p L0_ind=11n
`myCalib`	198	4	99	-85
`tank_tb`	C0_cap=8p L0_ind=9n	C0_cap=8p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
`calibval`	-85	-85	-85	-85

Recommendations for Using ?matchParams in calcVal

Depending on the variable and corner settings in the target and source (calibrated) test, you can specify an appropriate value for the `?matchParams` argument of `calcVal` to obtain correct results. You may find the following recommendations useful:

- Use the default value, `"none"`, when the following conditions are met:
  - The source and target tests do not use local sweep variables
  - The count of sweep values of variables and corners in the source and target tests is same
  - The values of all enabled corners and swept variables are identical

  In this case, `calcVal` considers only the count, and setup of variables and corners. It does not look for matching parameters.
- Set `?matchParams` to `"all"` when the tests have local sweeps or corners with varying values. In this case, `calcVal` finds points with matching sweeps and corners in the current or given history or run of the source test.
- Set `?matchParams` to *`<a-list-of-specific-design-points>`* or a user-defined function that returns a list containing name-value pairs of variables when the tests have local sweeps or corners with varying values, and you want `calcVal` to explicitly look for the given matching design points in the source and target tests. See the example in case IV above.
- To use the results of a non-identical corner from the source test, use the default value of `?matchParams` and specify the corner name of the source test by using the `?cornerName` argument.
- To use the results from another history or run by using the `?history` or `?run` arguments of `calcVal`, ensure that the design points are identical. These arguments can be used with any value of `?matchParams`.
- Do not specify `?matchParams` when `?getFirstSweepPoint` is set to `t`, which indicates that the result of the first sweep point in the source test is to be used for all design points in the target test. `?getFirstSweepPoint` works with the default value of `?matchParams` and ignores any other value set for it.

axlRenameOutputsColumn

```
axlRenameOutputsColumn(
     x_mainSDB
     t_columnName
     t_newColumnName
     )
     => t / nil
```

Description

Changes the name of a user-defined column in the Outputs table.

Arguments

x_mainSDB	Handle to the main setup database
t_columnName	Name of the user-defined column to be renamed
t_newColumnName	New name to be assigned to the user-defined column

Value Returned

`t`	When the column is successfully renamed
`nil`	Unsuccessful operation

Example

The following example demonstrates how to rename a user-defined column in the Outputs table:

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlRenameOutputsColumn( x_mainSDB "SpecDescr" "Spec Description")
=>t
```

axlSetOutputUserDefinedData

```
axlSetOutputUserDefinedData(
     x_mainSDB
     t_testName
     t_outputName
     t_columnName
     t_columnValue
     )
     => t / nil
```

Description

Sets value in the given user-defined column for the given test name and output.

Arguments

x_mainSDB	Handle to the main setup database
t_testName	Name of a test in the setup database
t_OutputName	Name of an output
t_columnName	Name of a user-defined column
t_Value	Value to be set in the column

Value Returned

`t`	Returns t when the value is successfully set in the given column
`nil`	Unsuccessful operation

Example

The following example demonstrates how to display value for output `Gain` in the `Spec Description` user-defined column.

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlSetOutputUserDefinedData(x_mainSDB "ACGainBW" "UGF" "Spec Description" "UGF >
1.5M")
t
```

7

Test-Related Functions

Test-Related SKILL Functions

Function	Description
axlGetCornersForATest	Returns a list of corners enabled for the given test.
axlGetEnabledGlobalVarPerTest	Returns the status of a global variable for the given test. When a global variable is overridden for a test, this function returns `nil`, which implies that the value of the global variable will not be considered for the test. Instead, the test will use a local value set for that variable.
axlGetEnabledTests	Returns a list of tests enabled in the given ADE XL setup database.
axlGetOrigTestToolArgs	Returns an associative list of original tool options set for the test before you ran the simulation after modifying the test setup.
axlGetTest	Finds a test in the setup database and returns its handle.
axlGetTests	Returns a list containing a handle to all tests in the setup database and a list of all test names.
axlGetTestToolArgs	Returns an associative list of tool option names and values for a test.
axlSaveResValue	Adds a name and value to the results database for the current point. You can use this function to specify an OCEAN measurement that you want to appear on the Outputs assistant pane.
axlSetTestToolArgs	Sets the tool options for the test.

Test-Related SKILL Functions, *continued*

Function	Description
axlToolSetOriginalSetupOptions	Sets options to their original values for the tool instance associated with the specified session and test.
axlToolSetSetupOptions	Sets the option values for the tool instance associated with the specified session and test.
axlWriteOceanScriptLCV	Writes an OCEAN script for the given adexl view in the specified file. If `axlWriteOceanScriptLCV()` is used in an existing session, it will write the current in-memory values and not the values saved on the disk.

Example Scripts

axlGetCornersForATest

```
axlGetCornersForATest(
     x_session
     t_test
     )
     => l_corners / nil
```

Description

Returns a list of corners enabled for the given test.

Arguments

x_session	Handle to the session
t_test	Test name

Value Returned

l_corners	List containing the names of corners associated with the given test and a list of corner variables and their value pairs.
`nil`	Unsuccessful operation

Example

The following example returns the list of all the enabled corners assiciated with test `Test1`:

```
data_session = axlGetWindowSession(hiGetCurrentWindow())

axlGetCornersForATest(data_session "Test1")

(("C1_VDD_2.2_Temp_1"
    (("VDD" "2.2")
        ("temperature" "75")
        ("corModelSpec" "File=All#Global#gpdk045.scs Section=\"tt\";")
    )
    )
    ("C1_VDD_2.2_Temp_0"
    (("VDD" "2.2")
        ("temperature" "-25")
```

```
        ("corModelSpec" "File=All#Global#gpdk045.scs Section=\"tt\";")
    )
    )
("" nil)
)
```

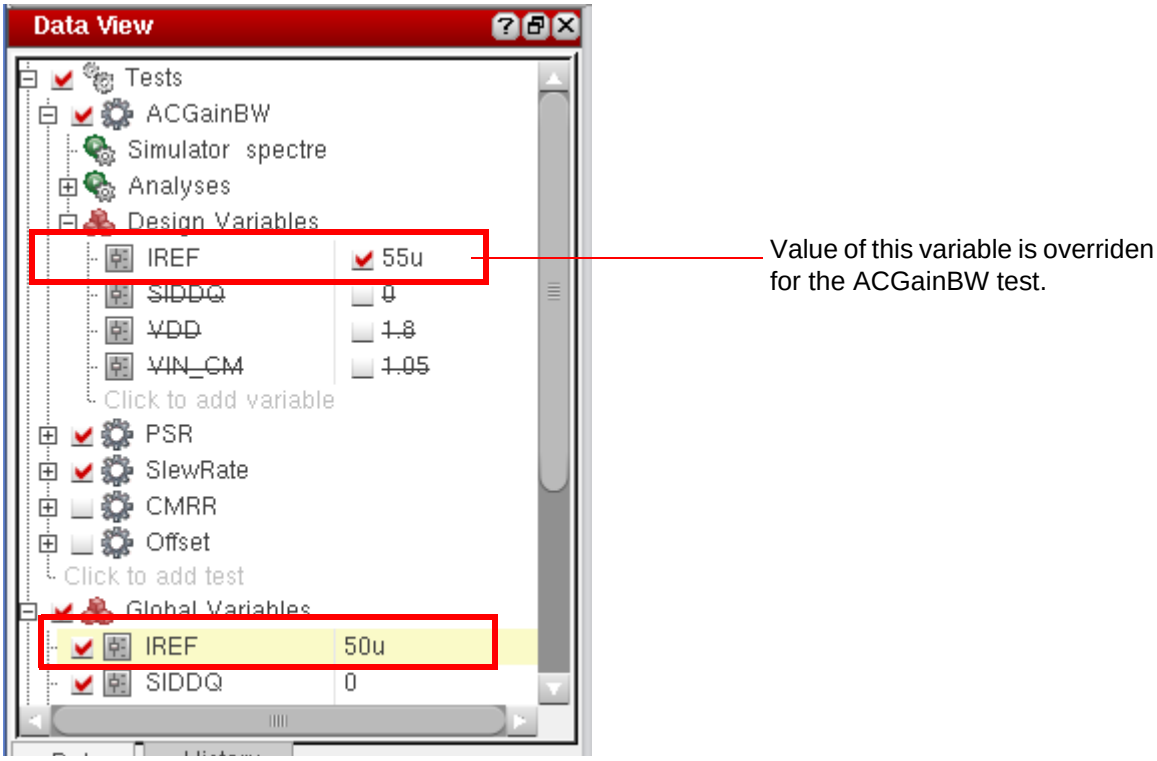
axlGetEnabledGlobalVarPerTest

```
axlGetEnabledGlobalVarPerTest(
     x_hsdb
     t_varName
     t_test
     )
     => t / nil
```

Description

Returns the status of a global variable for the given test. When a global variable is overridden for a test, this function returns `nil`, which implies that the value of the global variable will not be considered for the test. Instead, the test will use a local value set for that variable.

For example, in the figure shown below, the function will return `nil` for `IREF` because the global variable is not enabled or used for the test.



The function does not determine whether point sweeps or variables are enabled for simulation runs. Use the following functions for such cases:

- Use axlGetAllVarsDisabled to find if the variables are enabled for the simulation run. This function returns the status of the *Global Variables* check box in the *Data View* pane.

- Use axlGetAllSweepsEnabled to find if the sweep points are enabled for the simulation run. This function returns the status of the *Point Sweep* check box in the *Run Summary* pane.
- Use axlGetEnabled to find if a specific variable is enabled.

Arguments

x_hsdb	Handle to the main setup database
t_varName	Name of the global variable
t_test	Name of the test

Value Returned

`t`	Returns `t` if the global variable is enabled for the given test
`nil`	Unsuccessful operation

Example

The following code returns the status of the global variables `VDD` and `IREF` for the test `ACGainBW` when the setup is as shown in the example figure shown above:

```
s1 = axlGetWindowSession(hiGetCurrentWindow())
x_mainSDB=axlGetMainSetupDB(s1)
=> 1001
axlGetEnabledGlobalVarPerTest(x_mainSDB "VDD" "ACGainBW")
=>t
axlGetEnabledGlobalVarPerTest(x_mainSDB "IREF" "ACGainBW")
=>nil
```

Related Functions

axlSetDesignVariablePerTest

axlGetEnabledTests

```
axlGetEnabledTests(
     x_mainSDB
     )
     => l_tests / nil
```

Description

Returns a list of tests enabled in the given ADE XL setup database.

Arguments

x_mainSDB	Handle to the setup database

Value Returned

l_tests	A list containing the handles and names of the tests enabled in the given ADE XL setup
nil	Unsuccessful operation

Example

The following example shows how to get the list of enabled tests:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB = axlGetMainSetupDB(session)
=>1001
axlGetEnabledTests(x_mainSDB)
=>((1005 "AC") (1013 "TEST"))
```

axlGetOrigTestToolArgs

```
axlGetOrigTestToolArgs(
     x_hsdb
     )
     => l_toolOptions / nil
```

Description

Returns an associative list of original tool options set for the test before you ran the simulation after modifying the test setup.

Argument

x_hsdb	Handle to the test.

Value Returned

l_toolOptions	Associative list of original tool options set for the test before you ran a simulation after modifying the test setup.
nil	Unsuccessful operation.

Example

```
data_session = axlGetWindowSession(hiGetCurrentWindow())
data_sdb=axlGetMainSetupDB(data_session)

;; Get test args
testh= axlGetTest( data_sdb "OpAmp1" )
axlGetOrigTestToolArgs( testh )
'(("lib"   "opamplib")
  ("cell"  "ampTest")
  ("view"  "schematic")
  ("sim"   "spectre")
  ("path"  "./aState")
  ("state" "tran_state")
 )
```

Reference

axlGetTestToolArgs

axlGetTest

```
axlGetTest(
     x_hsdb
     t_test
     )
     => x_test / nil
```

Description

Finds a test in the setup database and returns its handle.

Arguments

x_hsdb	Setup database handle.
t_test	Test name.

Value Returned

x_test	Test handle.
nil	Unsuccessful operation.

Example

```
data_session = axlGetWindowSession(hiGetCurrentWindow())
data_sdb=axlGetMainSetupDB(data_session)
;; Enable a test
data_dead_band = axlGetTest( data_sdb "data_dead_band" )
axlSetEnabled( data_dead_band t )

;; Disable tests
foreach( test cadr( axlGetTests( data_sdb ) )
        axlSetEnabled( axlGetTest( data_sdb test ) nil )
3
```

Reference

axlCreateSession, axlSetMainSetupDB, axlGetTests, axlSetEnabled

axlGetTests

```
axlGetTests(
     x_hsdb
     )
     => l_tests / nil
```

Description

Returns a list containing a handle to all tests in the setup database and a list of all test names.

Argument

x_hsdb	Setup database handle.

Value Returned

l_tests	List containing a handle to all tests in the setup database and a list of all test names.
nil	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
;; Disable tests
foreach( test cadr( axlGetTests( data_sdb ) )
        axlSetEnabled( axlGetTest( data_sdb test ) nil )
'((1011 "Trans_12u")
  (1021 "dc_ac10G_2")
 )
```

Reference

axlSetMainSetupDB, axlSetEnabled, axlGetTest

axlGetTestToolArgs

```
axlGetTestToolArgs(
     x_hsdb
     )
     => l_toolOptions / nil
```

Description

Returns an associative list of tool option names and values for a test.

Argument

x_hsdb	Handle to the test.

Value Returned

l_toolOptions	Associative list of tool option names and values for the test. Valid Values when the tool is ADE: `lib` *t_libName* — Library name. `cell` *t_cellName* — Cell name. `view` *t_viewName* — View name. `sim` *t_simulator* — Simulator name. `state` *t_stateName* — State name. `path` *t_path* — Path to ADE state.
`nil`	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())

;; Get test args
testh= axlGetTest( data_sdb "OpAmp1" )
axlGetTestToolArgs( testh )
'(("lib"   "opamplib")
  ("cell"  "ampTest")
  ("view"  "schematic")
  ("sim"   "spectre")
  ("path"  "./artist_state")
  ("state" "tran_state")
 )
```

Reference

axlGetOrigTestToolArgs, axlToolSetSetupOptions

axlSaveResValue

```
axlSaveResValue(
     t_resultName
     g_resultValue
     )
     => t / nil
```

Description

Adds a name and value to the results database for the current point. You can use this function to specify an OCEAN measurement that you want to appear on the Outputs assistant pane.

Arguments

t_resultName	Result name.
g_resultValue	Result value (number, list, or waveform).

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Example

```
axlSaveResValue( "PhaseMargin" result )
t
```

axlSetTestToolArgs

```
axlSetTestToolArgs(
     x_hsdb
     l_toolOptions
     )
     => x_hsdb / nil
```

Description

Sets the tool options for the test.

Arguments

x_hsdb	Handle to the test for which you need to set options.
l_toolOptions	Associative list of tool option names and values for the test. Valid Values when the tool is ADE:

`lib` *t_libName*

Library name.

`cell` *t_cellName*

Cell name.

`view` *t_viewName*

View name.

`sim` *t_simulator*

Simulator name.

`state` *t_stateName*

State name.

`path` *t_path* Path to ADE state.

Value Returned

x_hsdb	Setup database handle.
`nil`	Unsuccessful operation.

Example

The following example code shows how to load a state saved from ADE L for a test into ADE XL:

```
session = axlGetWindowSession()
sdb = axlGetMainSetupDB(session)
libName = axlGetSessionLibName(session)
cellName = axlGetSessionCellName(session)

handleToTest = axlPutTest(sdb "state_test" "ADE")

axlSetTestToolArgs(handleToTest list( list("lib" libName) list("cell" cellName)
list("view" "schematic")  list("sim" "spectre")))

testSession=asiGetSession(handleToTest)
```

```
asiLoadState(testSession ?name "spectre_state1" ?lib libName ?cell cellName
?simulator "spectre")
```

axlToolSetOriginalSetupOptions

```
axlToolSetOriginalSetupOptions(
     t_session
     t_test
     l_toolOptions
     [ ?history x_history ]
     )
     => t / nil
```

Description

Sets options to their original values for the tool instance associated with the specified session and test.

Arguments

t_session	Session name.
t_test	Test name.
l_toolOptions	Associative list of original option names and values for the tool instance. Valid Values for tool instance ADE: `lib` *t_libName* Library name. `cell` *t_cellName* Cell name. `view` *t_viewName* View name. `sim` *t_simulator* Simulator name. `state` *t_stateName* State name. `path` *t_path* Path to ADE state.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlToolSetOriginalSetupOptions( "session0" "delayTest" axlGetOrigTestToolArgs(
1031 ) )
t
```

Reference

axlGetOrigTestToolArgs, axlToolSetSetupOptions

axlToolSetSetupOptions

```
axlToolSetSetupOptions(
     t_session
     t_test
     l_toolOptions
     )
     => t / nil
```

Description

Sets the option values for the tool instance associated with the specified session and test.

Arguments

t_session	Session name.
t_test	Test name.
l_toolOptions	Associative list of original option values for the tool instance. Valid Values for tool instance ADE:

`lib` *t_libName*

Library name.

`cell` *t_cellName*

Cell name.

`view` *t_viewName*

View name.

`sim` *t_simulator*

Simulator name.

`state` *t_stateName*

State name.

`path` *t_path* Path to ADE state.

Value Returnedz

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlToolSetSetupOptions( "session0" "delayTest" axlGetTestToolArgs( 1031 ) )
t
```

Reference

axlGetTestToolArgs, axlToolSetOriginalSetupOptions

axlWriteOceanScriptLCV

```
axlWriteOceanScriptLCV(
     t_fileName
     t_libraryName
     t_cellName
     t_viewName
     => t / nil
```

Description

Writes an OCEAN script for the given adexl view in the specified file. If `axlWriteOceanScriptLCV()` is used in an existing session, it will write the current in-memory values and not the values saved on the disk.

Note: If a file already exists with the same name, it is overwritten with the new one.

Arguments

t_fileName	Name of the OCEAN file in which you need to save the OCEAN script.
t_libraryName	Library name in the adexl view
t_cellName	Cell name in the adexl view
t_viewName	Name of the adexl view

Value Returned

`t`	Returns `t` when the function sucessfully writes an OCEAN script for the given view.
`nil`	Returns `nil` when the function fails to write an OCEAN script.

Example

The following example demonstrates how to save an OCEAN script for the `opamplib:ampTest:adexl` cellview in the `oceanScript.ocn` file:

```
axlWriteOceanScriptLCV("oceanScript.ocn" "opamplib" "ampTest" "adexl")
=> t
```

Example Scripts

Example 1

The following example script shows how to modify analyses for multiple ADE XL tests at one time.

```
LJNaxlUpdateAnalysisFieldValAllTests (analysis field value)

;    Update all tests that contain the specified analysis type.
;    e.g.  (LJNaxlUpdateAnalysisFieldValAllTests 'tran 'stop 100n)

LJNaxlUpdateTranStopTimeAllTests (stopTime)


;    Shortcut function to update tran stop time.
;    e.g.  LJNaxlUpdateTranStopTimeAllTests 100n
;     An ADE XL view must be opened.  Make sure the ADE XL view is in focus before
; entering the command in the CIW. All tests that contain the specified analysis
type will be updated.

;load this file in the .cdsinit:
(when !(axlIsICRPProcess)
    load("<this filename>")
)

;Example: Add $HOME/skill directory to SKILL path:
    setSkillPath(cons("~/skill" getSkillPath()))

KPNS: The GUI does not update automatically with the new analysis value although
the value has been updated. If you close the view and open it again the value is
updated or if you open the 'Choosing Analyses' form and select OK the value (e.g.
tran stop time) is updated.


(defun LJNaxlUpdateAnalysisFieldValAllTests (analysisType field value)
    (let (axl tests testSession oSession analysisName analysisList)
        axl = (LJNaxlGetCurrentWindowSessionAndSetupDB)
        tests = (cadr (axlGetTests axl->sdbh) )
    ; Special case to convert tran stop time to string if necessary (handle 1e-9 or
    ; 1n without double-quotes)

     (when (and (analysisType == 'tran) (field == 'stop) (not (stringp value)))
            value = (aelSuffixNotation value)
     )

         (foreach testName tests
            (printf "Test %s\n" testName)
            testSession = (axlGetToolSession axl->sess testName) ;sev session
            oSession = (sevEnvironment testSession) ;oasis session
            analysisName = (asiGetAnalysisName (asiGetAnalysis oSession
analysisType))
            analysisList = (asiGetEnabledAnalysisList oSession)
            (foreach analysis analysisList
                (printf "\tAnalysis name = %L\n" analysis->name)
            (when analysis->name == analysisName
                (printf "\t\tSetting %A = %A\n" field value)
                (asiSetAnalysisFieldVal analysis field value)
                (printf "\tNew val = %A\n" (asiGetAnalysisFieldVal analysis field
```

```
)))
        )
        )
    )
)
(defun LJNaxlUpdateTranStopTimeAllTests (stopTime)
    (let ()
        (LJNaxlUpdateAnalysisFieldValAllTests 'tran 'stop stopTime)
    )
)

;Utility functions

(defun LJNaxlGetCurrentWindowSessionAndSetupDB ()
    (let (sess sdbh ret)
        sess = (LJNaxlGetCurrentWindowSession)
         (unless sdbh
            (error "Cannot find ADE XL setupDB handle.\n")
        )
        ret=(ncons nil)
        ret->sess=sess
        ret->sdbh=sdbh
        ret
       )
)

(defun LJNaxlGetCurrentWindowSetupDB ()
       (let (sess sdbh)
       sess = (LJNaxlGetCurrentWindowSession)
       sdbh = (axlGetMainSetupDB sess)
       (unless sdbh
              (error "Cannot find ADE XL setupDB handle.\n")
       )
       sdbh
       )
)

(defun LJNaxlGetCurrentWindowSession ()
(       let (sess)
              sess = (axlGetWindowSession (hiGetCurrentWindow))
       (unless sess
              (error "Cannot find ADE XL session from current window. Select the
ADE XL window to set the current window.\n")
       )
       sess
       )
)

;To get a list of available analysis fields for an analysis:
;e.g:
;(LJNdisplayAnalysisField 'tran)
;(LJNdisplayAnalysisField 'dc)

(defun LJNdisplayAnalysisField (analysisType)
       (let (session analysis)
              session = asiGetCurrentSession()
              analysis = asiGetAnalysis(session analysisType)
              asiDisplayAnalysisField(analysis)
       )
)
```

8

Specification-Related Functions

Specification-Related SKILL Functions

Function	Description
axlAddSpecToOutput	Adds a specification to an output defined for a test. You can also use this function to modify an existing specification for an output.
axlGetSpecs	Returns a list containing a handle to all specifications in the setup database and a list of all specification names.
axlGetSpec	Finds the named specification in the setup database and returns its handle.
axlGetSpecData	Returns the specification for the given result, test and corner combination in the given setup database.
axlGetSpecWeight	Returns the weight value for a specification.

Important

In the earlier releases, `axlPutSpec` function was used to add a specification for an output and the `axlSetSpec*` functions, such as `axlSetSpecMax` or `axlSetSpecRange`, were used to add specification values and other details. It is now recommended to use the axlAddSpecToOutput function instead of the `axlPutSpec` or `axlSetSpec*` functions to add a specification for an output.

axlAddSpecToOutput

```
axlAddSpecToOutput(
     x_hsdb
     t_testName
     t_resultName
     [ ?min g_minValue ]
     [ ?max g_maxValue ]
     [ ?gt g_greaterThanValue ]
     [ ?lt g_lessThanValue ]
     [ ?range g_rangeValues ]
     [ ?tol g_toleranceValue ]
     [ ?info g_info ]
     [ ?weight g_weightingFactor ]
     [ ?corner g_cornerName ]
     )
     => t / t_error
```

Description

Adds a specification to an output defined for a test. You can also use this function to modify an existing specification for an output.

Arguments

x_hsdb	Setup database handle.
t_testName	Name of the test.
t_resultName	Name of the result.
?min *g_minValue*	Value for the min spec.
?max *g_maxValue*	Value for the max spec.
?gt *g_greaterThanValue*	Value for the greater than spec.
?lt *g_lessThanValue*	Value for the less than spec.
?range *g_rangeValues*	A range of values for the range spec.
?tol *g_toleranceValue*	Value for the tolerance spec.
?info *g_info*	Any information string for info spec.
?weight *g_weightingFactor*	A weighting factor for this spec.
?corner *g_cornerName*	Name of the corner for which the spec is to be enabled. This argument helps to override a specification for a particular corner. By default, a specification defined for a measurement applies to all the corners enabled for the test. To change the specification for a particular corner, specify the name of that corner in this argument. In the ADE XL GUI, you can view the overridden corner name in the Override Specifications form. For more details on overriding a specification for a corner, refer to Overriding the Measurement Specification for a Corner in the *Virtuoso Analog Design Environment XL User Guide*.

Value Returned

t	Successful addition of specification to an output.
t_error	If unsuccessful, returns an error message.

Example

The following example shows how to add different types of specification to an exisitng output for a test:

```
session = (axlGetWindowSession)
sdb = (axlGetMainSetupDB session)

axlAddSpecToOutput(sdb "TRAN" "" ?gt "1")
t

axlAddSpecToOutput(sdb "TRAN" "" ?lt "1" ?weight "1" ?corner "CC_1")
t

axlAddSpecToOutput(sdb "TRAN" ""  ?range 1:3 )
t
; The above statement removes the existing lt spec from the  output and
; assigns the range spec.
axlAddSpecToOutput(sdb "TRAN" ""  ?info "info spec")
t
axlAddSpecToOutput(sdb "TRAN" "gain"  ?tol 1:2 ?corner "CC_2")
t
```

axlGetSpecs

```
axlGetSpecs(
     x_hsdb
     )
     => l_list / nil
```

Description

Returns a list containing a handle to all specifications in the setup database and a list of all specification names.

Argument

x_hsdb	Setup database handle.

Value Returned

l_list	List containing a handle to all specifications in the setup database and a list of all specification names.
nil	Unsuccessful operation.

Example

The following example shows how to get all the exisitng specifications from the given setup database:

```
session = axlGetWindowSession()
x_mainSDB = axlGetMainSetupDB(session)
axlGetSpecs( x_mainSDB )
(1002 ( "opamplib:ampTest:1.gain" "opamplib:ampTest:1.bandwidth") )
```

axlGetSpec

```
axlGetSpec(
     x_hsdb
     t_specName
     )
     => x_spec / 0
```

Description

Finds the named specification in the setup database and returns its handle.

Arguments

x_hsdb	Setup database handle.
t_specName	Specification name.

Value Returned

x_spec	Specification handle.
0	Unsuccessful operation.

Example

The following example shows how to get the handle to an exisitng specification from the given setup database:

```
session = (axlGetWindowSession)
sdb = (axlGetMainSetupDB session)
axlGetSpec(sdb "opamplib:ampTest:1.gain")
1002
```

For more examples, see axlGetSpecData.

axlGetSpecData

```
axlGetSpecData(
     x_hsdb
     t_resultName
     t_testName
     [ t_cornerName ]
     )
     => l_specDetails / nil
```

Description

Returns the specification for the given result, test and corner combination in the given setup database.

Arguments

x_hsdb	Handle to the setup database.
t_resultName	Name of the measure.
t_testName	Name of the test.
t_cornerName	Name of the corner.

Value Returned

l_specDetails	Details of specification.
nil	If no specification exists for the given result.

Examples

The following examples show how you can get the specification details for a given result:

Example 1

```
session = (axlGetWindowSession)
x_mainSDB = (axlGetMainSetupDB session)
axlGetSpecData(x_mainSDB "pw" "opamplib:ampTest:1" "C0")
=>("gt" "10")
```

Example 2

```
axlGetSpecData(x_mainSDB "pw" "opamplib:ampTest:2")
=>("range" "10" "20")
```

Example 3

```
axlGetSpecData(x_mainSDB "pw" "opamplib:ampTest:3")
=>nil
```

Example 4

The following example displays specification details for all the results in the active setup database.

```
session = (axlGetWindowSession)
x_mainSDB = (axlGetMainSetupDB session)
(
foreach spec (cadr (axlGetSpecs x_mainSDB))
    specname=parseString(spec ".")
    test=car(specname)
    result=cadr(specname)
    specval=axlGetSpecData(1001 result test)
    printf("test=%s, result=%s, specValue=%L \n" test result specval)
)
```

The above code displays all the specification details, as shown below.

```
test=Two_Stage_Opamp:OpAmp_AC_top:1, result=Current, specValue=("lt" "1m")
test=Two_Stage_Opamp:OpAmp_AC_top:1, result=Gain, specValue=("max" "45")
test=Two_Stage_Opamp:OpAmp_AC_top:1, result=UGF, specValue=("gt" "250M")
test=Two_Stage_Opamp:OpAmp_TRAN_top:1, result=SettlingTime, specValue=("lt" "9n")
test=Two_Stage_Opamp:OpAmp_TRAN_top:1, result=Swing, specValue=("gt" "1")
test=Two_Stage_Opamp:OpAmp_AC_top:1:1, result=Current, specValue=("lt" "1e-3")
test=Two_Stage_Opamp:OpAmp_AC_top:1:1, result=Gain, specValue=("max" "45.0")
test=Two_Stage_Opamp:OpAmp_AC_top:1:1, result=UGF, specValue=("gt" "250e6")
```

Reference

axlGetSpec, axlGetSpecs

axlGetSpecWeight

```
axlGetSpecWeight(
     x_spec
     )
     => t_weight / nil
```

Description

Returns the weight value for a specification.

Argument

x_spec	Specification handle.

Value Returned

t_weight	Weight value.
nil	Unsuccessful operation.

Example

```
session = (axlGetWindowSession)
x_mainSDB = (axlGetMainSetupDB session)
spec = axlGetSpec(x_mainSDB "gain" )
axlGetSpecWeight( spec )
=>1
```

Reference

axlGetSpec

9

Corners-Related Functions

This chapter describes the public SKILL functions that can be used to work with corners in an ADE XL setup.

Corners-Related SKILL Functions

Function	Description
axlGetAllCornersEnabled	
	Returns the selection status of the Corners check box in the Run Summary pane.
axlCorners	Opens the Corners Setup form.
axlGetCorner	Finds a corner by name and returns a handle to that corner.
axlGetCorners	Returns a list containing a handle to all corners and a list of names of corners and corner groups in the setup database.
axlGetCornerDisabledTests	
	Returns a list containing a handle to the disabled tests and the names of tests that are disabled for the given corner.
axlGetCornerCountForName	
	Returns the count of individual corners contained in the specified corner group.
axlGetCornerNameForCurrentPointInRun	
	Returns the name of the corner for the current point being simulated. This function is useful for debugging in OCEAN script based measures.
axlGetNominalCornerEnabled	
	Returns `t` if the nominal corner is enabled in the specified setup database. This is same as the status of the Nominal check box on the Run Summary assistant.

Corners-Related SKILL Functions, *continued*

Function	Description
axlGetNominalCornerTestEnabled	Returns the status of the check box corresponding to the given test in the Nominal column on the Corners Setup form.
axlGetStatVars	Returns a list of the names of statistical variables that use the `statistical::mismatch` or `statistical::global` prefix for the given statistical corner.
axlLoadCorners	Loads a set of corners from the specified XML file in which the corners were saved earlier.
axlLoadCornersFromPcfToSetupDB	Imports a set of predefined corners from a process customization file into the corners setup for the given ADE XL session.
axlPlotAcrossDesignPoints	Plots an output across all the design points for a particular corner.
axlPutCorner	Adds a new corner by the given name and returns a handle to that corner. If a corner already exists with the same name, the function returns the handle to that corner.
axlPutDisabledCorner	Adds a new corner by the given name and returns a handle to that corner. If a corner already exists with the same name, the function returns the handle to that corner. In addition, the corner is disabled for the specified test name, but enabled for other tests in the ADE XL session.
axlSetDefaultCornerEnabled	Enables or disables the default (nominal) corner for the specified test. The program creates a nominal corner when you create a test. This corner represents the absence of corner-specific information.
axlSetAllCornersEnabled	Enables or disables all the corners for simulation. This changes the selection status of the Corners check box in the Run Summary assistant.
axlSetCornerName	Sets or updates the name of the given corner.

Corners-Related SKILL Functions, *continued*

Function	Description
axlSetCornerTestEnabled	Enables or disables simulation of a test for the given corner. This function changes the status of the check box corresponding to a test in the column for the given corner.
axlSetNominalCornerEnabled	Enables or disables the nominal corner in the specified setup database.
axlSetNominalCornerTestEnabled	Sets the status of the check box corresponding to the given test in the Nominal column on the Corners Setup form.
axlSetWCCTime	Sets the time information for the given specfication handle of a worst case corner.
axlGetWCCCorner	Gets the corner name for the given specification handle of a worst case corner.
axlGetWCCHistory	Returns name of the history for a specification of a worst case corner.
axlGetWCCResult	Returns result of a specification of a worst case corner.
axlGetWCCSpec	Returns result of a specification of a worst case corner.
axlGetWCCSpecs	Returns a list of specifications for the given worst case corner.
axlGetWCCTest	Returns name of the test of the given specification.
axlGetWCCTime	Returns the generating time information for the given specfication handle of a worst case corner.
axlGetWCCRangeBound	Returns an integer value that specifies whether the worst case corner corresponds to the minimum or the maximum value of the spec.
axlGetWCCVarMonotonicity	Gets the monotonicity of a specific variable or parameter of the worst case corner.
axlGetWCCVars	Returns a list containing a handle to all variables and a list of all variable names.

axlGetAllCornersEnabled

```
axlGetAllCornersEnabled(
     x_mainSDB
     )
     => t / nil
```

Description

Returns the selection status of the *Corners* check box in the Run Summary pane.

Argument

x_mainSDB	Setup database handle.

Value Returned

`t`	Returns `t`, if the *Corners* check box is selected in the Run Summary pane.
`nil`	Returns `nil`, if the *Corners* check box is deselected in the Run Summary pane.

Example

The following example gets the status of the Corners check box in the Run Summary pane and displays it:

```
axlsession=axlGetWindowSession( hiGetCurrentWindow())
=> "session0"
x_mainSDB=axlGetMainSetupDB(axlsession)
=> 1001
axlGetAllCornersEnabled(x_mainSDB)
=> t
```

Reference

axlGetWindowSession, axlGetMainSetupDB

axlCorners

```
axlCorners(
     t_session
     [ g_refresh ]
     )
     => t / nil
```

Description

Opens the Corners Setup form.

If Corners Setup form is not already open, ADE XL opens the form and brings it in focus. If the form is already opened, it is brought in focus. In the later case, the second argument specifies if any changes related to corners and tests are to be reflected in the Corners Setup form.

Arguments

t_session	String representing the ADE (G) XL session name.
g_refresh	Specifies if the changes related to corners and tests in the setup database are automatically reflected in the Corners Setup form while it is already open. Default value: `nil` If this argument is set to `nil`, you need to close and re-open the form to view these updates. When this variable is set to `t`, the updates are automatically reflected in the form. **Note:** This is helpful only when the Corners Setup form is already open.

Value Returned

t	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example opens the Corners Setup form for the current ADE XL session:

```
session_name = axlGetWindowSession()
=>"session0"

;; Load Corner Setup User Inteface Form
axlCorners(session_name t)
=> t
```

Reference

axlGetWindowSession, axlGetMainSetupDB

axlGetCorner

```
axlGetCorner(
     x_mainSDB
     t_cornerName
     )
     => x_corner / nil
```

Description

Finds a corner by name and returns a handle to that corner.

Arguments

x_mainSDB	Setup database handle.
t_cornerName	Corner name.

Value Returned

x_corner	Handle to a corner.
nil	Unsuccessful operation.

Examples

Example 1

The following example shows how to find a corner with name VDD_C0:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
cHandle = axlGetCorner(x_mainSDB "VDD_C0")
=> 1340
; you can further use this corner handle to modify or remove the corner
axlRemoveElement(cHandle)
```

Example 2

The following example code disables all the corners in the current ADE XL session:

```
session = axlGetWindowSession()
=>"session0"
```

```
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
;; Disable corners
foreach(corner cadr( axlGetCorners(x_mainSDB) )
    axlSetEnabled(axlGetCorner(x_mainSDB corner) nil))
=> ("C0" "C1" "C2")
```

Reference

axlGetWindowSession, axlGetMainSetupDB, axlSetEnabled, axlGetCorners

axlGetCorners

```
axlGetCorners(
     x_mainSDB
     )
     => l_corners / nil
```

Description

Returns a list containing a handle to all corners and a list of names of corners and corner groups in the setup database.

Argument

x_mainSDB	Setup database handle.

Value Returned

l_corners	List containing a handle to the corners and a list of names of corners and corner groups in the setup database.
`nil`	Unsuccessful operation.

Examples

Example 1

The following example code gets a list of all the corners and corner groups in the current ADE XL session:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
axlGetCorners(x_mainSDB)
=> (1003 ("C0" "C1" "C2_0_0" "C2_0_1" "C2_0_2")
)
```

Example 2

The following example code shows how to remove all the corners from the setup database of the current ADE XL session:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
axlGetCorners(x_mainSDB)
=> (1003
("C0" "C1" "C2_0_0" "C2_0_1" "C2_0_2")
)
axlRemoveElement(1003)
=> t
; this code removes all the existing corners from the setup database
)
```

Reference

axlGetWindowSession, axlGetMainSetupDB, axlRemoveElement, `axlGetCorner`

axlGetCornerDisabledTests

```
axlGetCornerDisabledTests(
     x_cornerHandle
     )
     => l_testNames / nil
```

Description

Returns a list containing a handle to the disabled tests and the names of tests that are disabled for the given corner.

Argument

x_cornerHandle	Handle to the corner for which you need to find the disabled tests.

Value Returned

l_testNames	List containing a handle to the disabled tests and the names of tests that are disabled for the given corner.
nil	Unsuccessful operation.

Example

The following example code gets a list of all the test names disabled for corner C1:

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
x_cHandle= axlGetCorner(x_mainSDB "C1") )
=> 1065
axlGetCornerDisabledTests(x_cHandle)
=> ( (3978 ("AC" "TRAN") )
```

axlGetCornerCountForName

```
axlGetCornerCountForName(
     x_mainSDB
     t_cornerGroup
     )
     => x_cornerCount / -1
```

Description

Returns the count of individual corners contained in the specified corner group.

Argument

x_mainSDB	Setup database handle.
t_cornerGroup	Name of the corner group.

Value Returned

x_cornerCount	Number of corners found in the corner group.
-1	If *x_mainSDB* or *t_cornerGroup* are not found.

Example

The following example code shows how to find the number of corners present in a particular corner group:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
axlGetCorners(x_mainSDB)
=> (1003
("C0" "C1" "C2_0_0" "C2_0_1" "C2_0_2"))
axlGetCornerCountForName 1003 "C1"
=> 1
axlGetCornerCountForName 1003 "C2_0_0"
=> 6
```

axlGetCornerNameForCurrentPointInRun

```
axlGetCornerNameForCurrentPointInRun(
     )
     => t_cornerName
```

Description

Returns the name of the corner for the current point being simulated. This function is useful for customized processing or debugging in the OCEAN script based measures.

Arguments

None

Value Returned

t_cornerName	Name of the corner for the current point being simulated. For nominal corner, the text " " is returned.

Example

```
axlGetCornerNameForCurrentPointInRun()
=> "cor_77"
```

where, `cor_77` is the current corner being run.

Reference

Loading an OCEAN or a MATLAB Measurement

axlGetNominalCornerEnabled

```
axlGetNominalCornerEnabled(
     x_mainSDB
     )
     => t / nil
```

Description

Returns `t` if the nominal corner is enabled in the specified setup database. This is same as the status of the *Nominal* check box on the Run Summary assistant.

Argument

x_mainSDB	Setup database handle.

Value Returned

`t`	Nominal corner is enabled.
`nil`	Nominal corner is disabled.

Example

The following example code returns the status of the nominal corner:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
axlGetNominalCornerEnabled(x_mainSDB)
=>t
```

axlGetNominalCornerTestEnabled

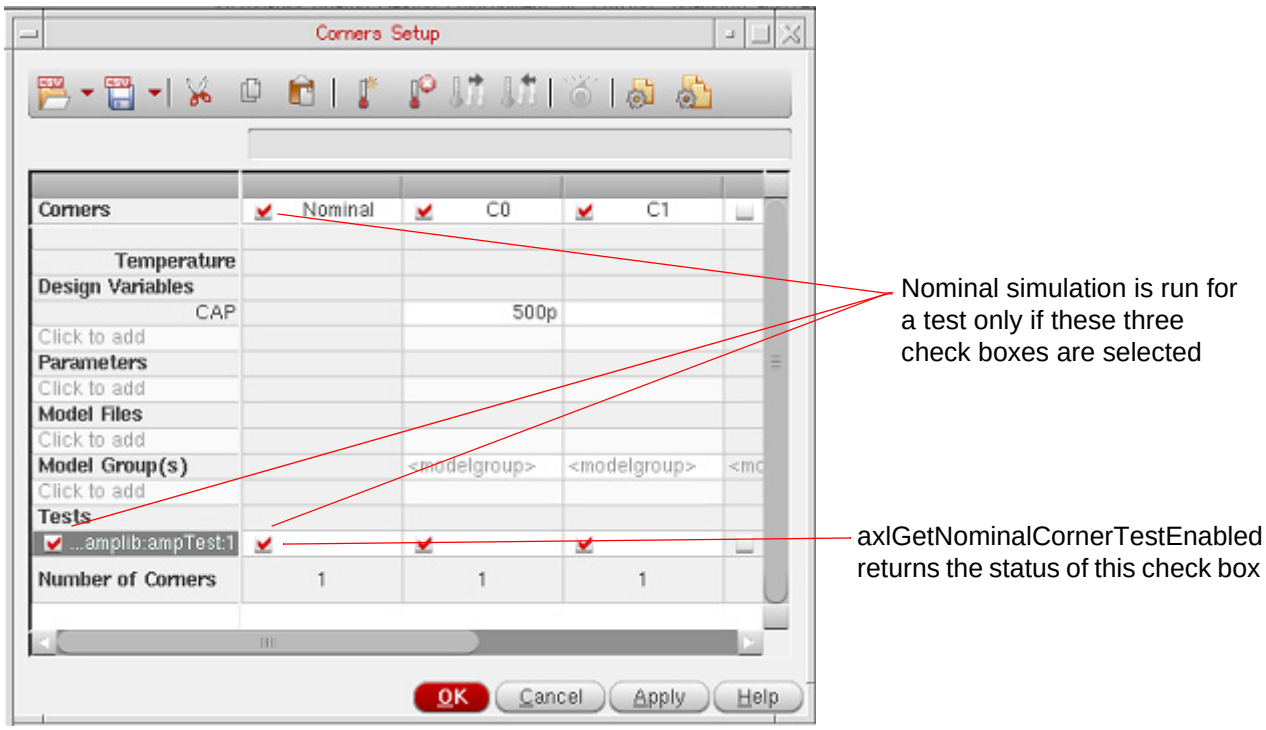
```
axlGetNominalCornerTestEnabled(
     t_testHandle
     )
     => t / nil
```

Description

Returns the status of the check box corresponding to the given test in the *Nominal* column on the Corners Setup form.

Nominal simulation for a test is run only if the following three conditions are met:

- The check box before the column heading in the *Nominal* column on the Corners Setup form is selected
- The check box before the test name in the row header for the tests is selected
- The check box in the cell corresponding to the given test in the *Nominal* corner is enabled



Argument

t_testHandle	Handle to the test for which you need to check if nominal simulation would be run

Value Returned

`t`	Returns `t`, if successful.
`nil`	Returns `nil`, if not successful.

Example

The following example code returns the status of the nominal corner:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
x_testHandle = axlGetTest(x_mainSDB "AC")
axlGetNominalCornerTestEnabled(x_testHandle)
=> t
```

Related Functions

axlGetNominalCornerEnabled, axlGetEnabled

axlGetStatVars

```
axlGetStatVars(
     x_mainSDB
     x_cornerId
     )
     => l_vars / nil
```

Description

Returns a list of the names of statistical variables that use the `statistical::mismatch` or `statistical::global` prefix for the given statistical corner.

Arguments

x_mainSDB	Handle to the active setup database.
x_cornerID	ID of the statistical corner.

Values Returned

l_vars	A list of statistical variables.
`nil`	Returns `nil`, if unsuccessful.

Example

```
x_mainSDB=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))
=>1002

cornerName=axlGetCorner(x_mainSDB "Stat_seq_1_params")
=>44567

statvarsList=axlGetStatVars(cornerName)
=> (("statistical:mismatch:I0.M7:parl2" "1.546454185")
    ("statistical:mismatch:I0.M7:plo_dxw" "1.412371304")
    ("statistical:mismatch:I0.M5:plo_dxl" "-0.8702216304")
    ("statistical:mismatch:I0.M2:plo_dxw" "-1.253658311")
    ("statistical:mismatch:I0.M9:parl1" "-0.878480435")
    ("statistical:mismatch:I0.M1:plo_tox" "0.8711784603")
    ("statistical:mismatch:I0.M1:parl1" "0.689194273")
    ("statistical:mismatch:I0.M6:plo_dxw" "-1.139652443")
    ("statistical:mismatch:I0.M5:plo_dxw" "0.3336112529")
    ("statistical:mismatch:I0.M3:plo_dxl" "-1.544827915")
    ("statistical:mismatch:I0.M2:plo_tox" "0.01353515771"))
```

axlLoadCorners

```
axlLoadCorners(
     x_mainSDB
     t_SDBfileName
     )
     => x_cornersHandle / 0
```

Description

Loads a set of corners from the specified XML file in which the corners were saved earlier.

Note: This function removes all the existing corners from the ADE XL setup before creating the corners loaded from the specified setup database (XML) file.

Arguments

x_mainSDB	Setup database handle.
t_SDBfileName	Path to the setup database file from which corner details are to be loaded.

Value Returned

x_cornersHandle	Returns t when successful.
0	Otherwise returns nil.

Example

The following example code loads the corners from the co.sdb file into the current ADE XL session:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
```

```
axlLoadCorners(x_mainSDB "/home/user1/co.sdb")
1003
```

axlLoadCornersFromPcfToSetupDB

```
axlLoadCornersFromPcfToSetupDB (
     t_session
     t_fileName
     t_testNameList
     g_overwriteExistingCorners
     )
     => t / nil
```

Description

Imports a set of predefined corners from a process customization file into the corners setup for the given ADE XL session.

Arguments

t_session	Name of session in which you want to import corners.
t_fileName	Path to the PCF file
t_testNameList	List of tests for which the imported corners should be enabled. Separate the test names in the list using a blank space.
g_overwriteExistingCorners	Flag to specify how corners should be imported when an existing corner and a corner defined in the PCF file have the same name. Valid values: `t` — If an existing corner and a corner defined in the PCF file have the same name, overwrite the existing corner with the corner defined in the PCF file. `nil` — If an existing corner and a corner defined in the PCF file have the same name, import the corner defined in the PCF file with a different name. For example, if there is an existing corner named `C1` and the PCF file has a corner named `C1`, the corner in the PCF file will be imported as `C1_0`.

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

The following example code loads corners from a PCF file to the setup database:

```
session = axlGetWindowSession()
=>"session0"
axlLoadCornersFromPcfToSetupDB("session0" "./myCorners.pcf" "\"test1\" \"test2\""
"t")
```

axlPlotAcrossDesignPoints

```
axlPlotAcrossDesignPoints(
     t_session
     t_testName
     t_historyName
     t_outputName
     t_cornerName
     )
     => x_corner / nil
```

Description

Plots an output across all the design points for a particular corner.

Arguments

t_session	Name of the ADE XL session
t_testName	Name of the test
t_historyName	Name of the history from which results are to be used
t_outputName	Name of the output to be plotted across corners
t_cornerName	Name of the corner for which the results are to be plotted

Value Returned

`t`	The results are successfully plotted
`nil`	Unsuccessful operation

Example

The following example plots the output `OPT_V` across all the design points for corner `C0_0`:

```
session = axlGetWindowSession()
=>"session0"
axlPlotAcrossDesignPoints("session" "Tran_sim" "Interactive.0" "OPT_v" "C0_0")
=>t
```

axlPutCorner

```
axlPutCorner(
     x_mainSDB
     t_cornerName
     )
     => x_corner / nil
```

Description

Adds a new corner by the given name and returns a handle to that corner. If a corner already exists with the same name, the function returns the handle to that corner.

Arguments

x_mainSDB	Setup database handle.
t_cornerName	Corner name.

Value Returned

x_corner	Handle to a corner.
nil	Unsuccessful operation.

Example

The following example code adds a new corner, testC, to the existing database and also assigns value to the VDD variable for that corner:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
axlPutCorner(x_mainSDB "testC")
=>2080
axlPutVar(2080 "VDD" "2.0")
=>2082
```

Reference

axlCreateSession, axlSetMainSetupDB

axlPutDisabledCorner

```
axlPutDisabledCorner(
     x_testHandle
     t_cornerName
     )
     => x_disabledcorner / nil
```

Description

Adds a new corner by the given name and returns a handle to that corner. If a corner already exists with the same name, the function returns the handle to that corner. In addition, the corner is disabled for the specified test name, but enabled for other tests in the ADE XL session.

You can also use this function to disable a specific corner for a particular test.

Arguments

x_testHandle	Test handle.
t_cornerName	Corner name.

Value Returned

x_disabledcorner	Handle to a disabled corner.
nil	Unsuccessful operation.

Example

Example 1:

The following example gets the list of all the corners in the setup database. Next, it disables corner `C2` for the `opamplib:ampTest:2` test :

```
s1 = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=>1001
axlGetCorners(x_mainSDB)
=> (1003
("C0" "C1" "C2" "C2_0_1" "C2_0_2")
)
```

```
x_testHandle2 = axlGetTest( x_mainSDB "opamplib:ampTest:2")
=>2028
axlPutDisabledCorner(x_testHandle2 "C2")
=>2186
```

Example 2:

The following example code adds a new corner, `testC2`, and disables it for the `opamplib:ampTest:1` test :

```
s1 = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=>1001
x_testHandle = axlGetTest( x_mainSDB "opamplib:ampTest:1")
=>1015
axlPutDisabledCorner(1015 "testC2")
=>2186
```

axlSetDefaultCornerEnabled

```
axlSetDefaultCornerEnabled(
     x_testHandle
     g_enable
     )
     => 1 / 0
```

Description

Enables or disables the default (nominal) corner for the specified test. The program creates a nominal corner when you create a test. This corner represents the absence of corner-specific information.

Arguments

x_testHandle	Test handle.	
g_enable	Enable flag. Valid Values:	
	`nil`	Disables the default corner for the test.
	any other value	Enables the default corner for the test.

Value Returned

`1`	Successful operation.
`0`	Unsuccessful operation.

Example

The following example code shows how to disable the nominal corner for a specific test:

```
s1 = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=>1001
testHandle = axlGetTest( x_mainSDB "test1" )
=>1005
axlSetDefaultCornerEnabled(1005 nil)
=>1
```

You can use the `axlSetDefaultCornerEnabled` function to enable or disable the nominal corner for a test in a SKILL trigger code.

Reference

axlCreateSession, axlGetTest, axlSetMainSetupDB

axlSetAllCornersEnabled

```
axlSetAllCornersEnabled(
     x_mainSDB
     g_enable
     )
     => t / nil
```

Description

Enables or disables all the corners for simulation. This changes the selection status of the *Corners* check box in the Run Summary assistant.

Arguments

x_mainSDB	Setup database handle.
g_enable	Specifies if the corners are to be enabled or disabled. Set the value as 1 to enable all corners, otherwise set it as `0`.

Value Returned

`t`	Returns `t`, if successful.
`nil`	Returns nil, if not successful.

Example

The following example code disables all the corners in the current ADE XL session:

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlSetAllCornersEnabled(x_mainSDB nil)
=> t
```

axlSetCornerName

```
axlSetCornerName(
     x_cornerHandle
     t_cornerName
     )
     => t / nil
```

Description

Sets or updates the name of the given corner.

Arguments

x_cornerHandle	Handle to the corner for which the name is to be changed.
t_cornerName	New name to be set for the corner

Value Returned

`t`	Returns `t`, if successful.
`nil`	Returns nil, if not successful.

Example

The following example code shows how to update the name for a corner:`session =`

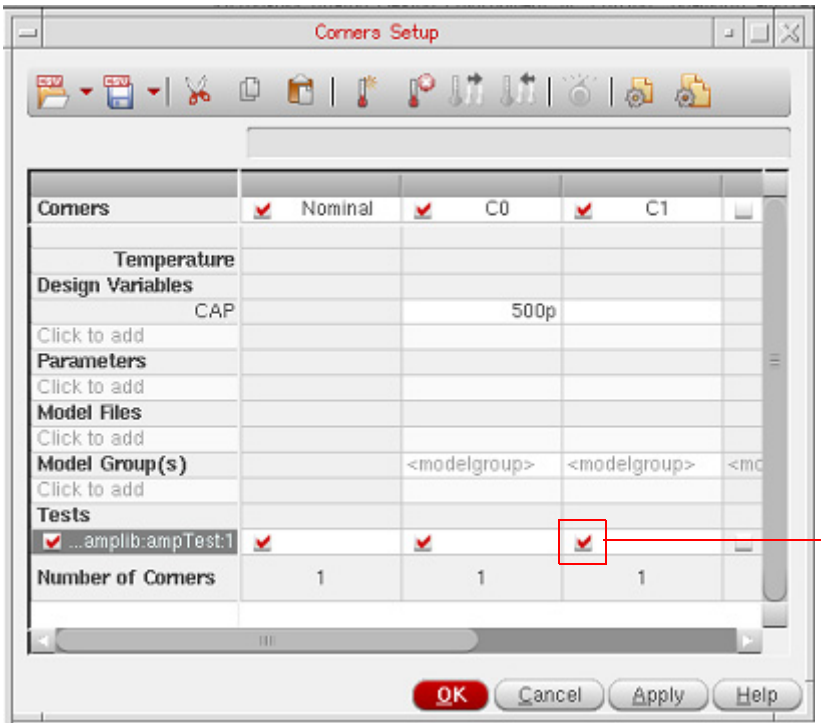
```
axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001

;; list pre-existing corners:
cadr(axlGetCorners(x_mainSDB))
=> ("C0" "C1")

;; retrieve the SDB handle for a specific corner C0
cornerHandle = axlGetCorner(x_mainSDB "C0")
=> 1234   ;; a non-zero integral handle

;; update the corner name
axlSetCornerName(cornerHandle "newName")
=> t   ;; successful modification
;; retrieve corners again to validate the name change
cadr(axlGetCorners(x_mainSDB))
=> ("newName "C1")
```

axlSetCornerTestEnabled

```
axlSetCornerTestEnabled(
     x_cornerHandle
     t_testName
     g_enableFlag
     )
     => x_num / nil
```

Description

Enables or disables simulation of a test for the given corner. This function changes the status of the check box corresponding to a test in the column for the given corner.



Changes the status of the check box corresponding to the given test and the given corner.

Arguments

x_cornerHandle	Handle to the corner
t_testName	Name of the test
g_enableFlag	Enable or disable status

Value Returned

x_num	Returns an integer value, if successful.
nil	Returns nil, if not successful.

Example

The following example code shows how to enable the test AC for corner C0:

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
x_cornerHandle = axlGetCorner(x_mainSDB "C0")
=> 3573
axlSetCornerTestEnabled(x_cornerHandle "AC" t)
=> 3979
```

axlSetNominalCornerEnabled

```
axlSetNominalCornerEnabled(
     x_mainSDB
     g_enable
     )
     => t / nil
```

Description

Enables or disables the nominal corner in the specified setup database.

Arguments

x_hsdb	Setup database handle.
g_enable	Specifies if the nominal corner is to be enabled or disabled. Set the value as 1 to enable the nominal corner, otherwise set it as `0`.

Value Returned

`t`	Returns `t`, if successful.
`nil`	Returns nil, if not successful.

Example

The following example code enables the nominal corner in the current ADE XL session:

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
axlSetNominalCornerEnabled(x_mainSDB 1)
=>t
```

axlSetNominalCornerTestEnabled

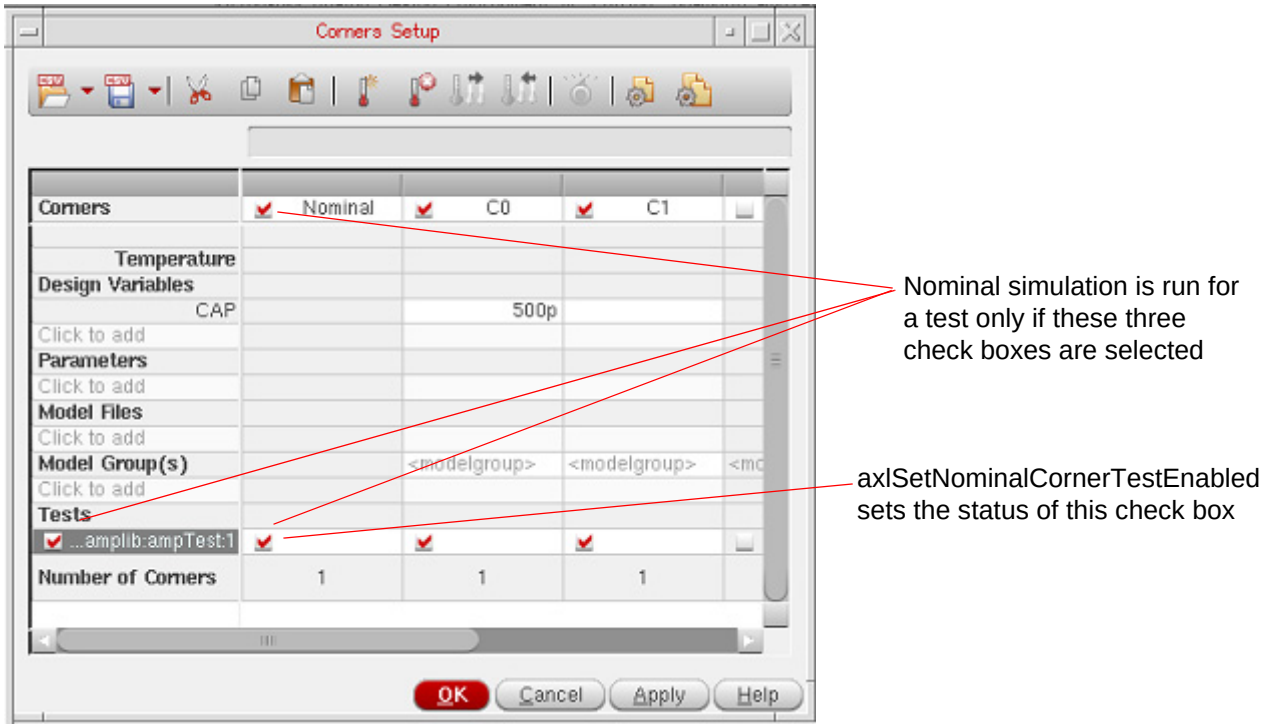
```
axlSetNominalCornerTestEnabled(
     x_testHandle
     g_enableFlag
     )
     => t / nil
```

Description

Sets the status of the check box corresponding to the given test in the *Nominal* column on the Corners Setup form.

Nominal simulation for a test is run only if the following three conditions are met:

- The check box before the column heading in the *Nominal* column on the Corners Setup form is selected
- The check box before the test name in the row header for the tests is selected
- The check box in the cell corresponding to the given test in the *Nominal* corner is enabled



Argument

t_testHandle	Handle to the test for which you need to specify if the nominal simulation would be run
g_enableFlag	`t` or `nil` status to be set for the check box corresponding to the given test

Value Returned

`t`	Returns `t`, if successful.
`nil`	Returns `nil`, if not successful.

Example

The following example code selects the status of the nominal corner:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
x_testHandle = axlGetTest(x_mainSDB "AC")
axlSetNominalCornerTestEnabled( x_testHandle t)
=> t
```

Related Functions

axlSetNominalCornerEnabled, axlSetEnabled

axlSetWCCTime

```
axlSetWCCTime(
     x_specID
     t_time
     )
     => t / nil
```

Description

Sets the time information for the given specfication handle of a worst case corner.

Note: This function does not require any particular format for specifying the input time and also no validity checks are required for the time string.

Arguments

x_specID	Handle to the specfication of the worst case corner.
t_time	Time information to be set.

Value Returned

`t`	Returns t if the time information is successfully set.
`nil`	Returns nil, if unsuccessful.

Example

```
spech = axlGetWCCSpec(1054, "Gain")
axlSetWCCTime( spech, "Mar  3 20:52:47 2014")
```

Here, `1054` is the worst case corner sdb handle.

This example sets the time information of the spech (specification handle) to "Mar 3 20:52:47 2014".

axlGetWCCCorner

```
axlGetWCCCorner(
     x_specHandle
     )
     => t_value / nil
```

Description

Gets the corner name for the given specification handle of a worst case corner.

Arguments

x_specHandle	Setup database handle to the specification of a worst case corner.

Value Returned

t_value	Returns name of the corner.
nil	Returns nil if no worst case corner is found.

Example

```
corner1 = axlGetWCCCorner(1005)
"_default"
```

axlGetWCCHistory

```
axlGetWCCHistory(
     x_specHandle
     )
     => t_historyName / nil
```

Description

Returns name of the history for a specification of a worst case corner.

Arguments

x_specHandle	Database handle to the specification of a worst case corner.

Value Returned

t_historyName	Returns name of the history.
nil	Returns nil, if not successful.

Example

```
t_history = axlGetWCCHistory(1005)
"History.1"
```

axlGetWCCResult

```
axlGetWCCResult(
     x_specHandle
     )
     => t_result / nil
```

Description

Returns result of a specification of a worst case corner.

Arguments

x_specHandle	Database handle to the specification of a worst case corner.

Value Returned

t_result	Returns result of the specification.
nil	Returns nil, if not successful.

Example

```
axlGetWCCResult(1005)
"avg_vt"
```

axlGetWCCSpec

```
axlGetWCCSpec(
     x_cornerHandle
     t_specName
     )
     => x_spec / nil
```

Description

Returns handle to a specification for the specified worst case corner.

Arguments

x_cornerHandle	Handle to a worst case corner for which you want to get specification.
t_specName	Name of the spec for which you want to get the handle.

Value Returned

x_spec	Returns handle to a specification.
nil	Returns nil, if not successful.

Example

```
axlGetWCCSpec(1005 "test1.result1")
1987
```

axlGetWCCSpecs

```
axlGetWCCSpecs(
     x_wccHandle
     )
     => l_specs / nil
```

Description

Returns a list of specifications for the given worst case corner.

Arguments

x_cornerHandle	Handle to the worst case corner.

Value Returned

l_specs	Returns list of specifications for the given worst case corner.
nil	Returns nil, if not successful.

Example

```
l_specs = axlGetWCCSpecs(1005)
=> (1940 ("solutions:ampTest:1.avg_vt" "solutions:ampTest:2.avg_vt1")
```

axlGetWCCTest

```
axlGetWCCTest(
     x_wccHandle
     )
     => t_testName / nil
```

Description

Returns name of the test of the given specification.

Arguments

x_wccHandle	Handle to the spec of a worst case corner..

Value Returned

t_testName	Returns name of the test of the given spec.
nil	Returns nil, if not successful.

Example

```
sess = axlGetWCCTest(x_)
"solutions:ampTest:1"
```

axlGetWCCTime

```
axlGetWCCTime(
     x_specId
     )
     => t_time / nil
```

Description

Returns the generated time information for the given specfication handle of a worst case corner.

Arguments

x_specId	Handle to the specification of the worst case corner.

Values Returned

t_time	Returns the time string value.
nil	Returns nil, if unsuccessful.

Example

```
specHandle = axlGetWCCSpec(1054, "Gain")
axlGetWCCTime(specHandle)
```

Here, `1054` is the worst case corner sdb handle. Prints the time information that was set for the specification handle, `specHandle`.

axlGetWCCRangeBound

```
axlGetWCCRangeBound(
     x_hsdb
     )
     => t_rangeBound
```

Description

Returns an integer value that specifies whether the worst case corner corresponds to the minimum or the maximum value of the spec.

Arguments

x_hsdb	Handle to the worst case corner

Value Returned

`t_rangeBound`	Returns an integer value that specifies whether the worst case corner corresponds to the minimum or the maximum value of the spec. Return values: ■ `0`: Indicates that the corner is created for the lower boundary of the spec range. ■ `1`: Indicates that the corner is created for the upper boundary of the spec range.

Example

```
corner = axlGetCorner(1001 "WCC_C2")
1934
axlGetWCCRangeBound(1934)
1
```

axlGetWCCVar

```
axlGetWCCVar(
     x_hscr
     t_name
     )
     => x_handle / nil
```

Description

Finds the specified variable by name and returns a handle to it.

Arguments

x_hscr	Handle to the worst case corner.
t_name	Name of the variable for which you want to get the handle.

Value Returned

x_handle	Returns handle to the variable.
nil	Returns nil, if the specified variable is not found.

Example

```
x_handleToVar = axlGetWCCVar(1005 "CAP")
1005
```

axlGetWCCVarMonotonicity

```
axlGetWCCVarMonotonicity(
     x_hsdb
     )
     => t_monotonnicity / nil
```

Description

Gets the monotonicity of a specific variable or parameter of the worst case corner.

Arguments

x_hsdb	Setup database handle to the variable or parameter for which you want to get the monotonicity.

Value Returned

`t_monotonicity`	Returns the monotonicity value.
`nil`	Returns `nil`, if unsuccessful.

Example

```
m_var1 = axlGetWCCVarMonotonicity(1005)
"+1"
```

axlGetWCCVars

```
axlGetWCCVars(
     x_hsdb
     )
     => l_vars / nil
```

Description

Returns a list containing a handle to all variables and a list of all variable names.

Arguments

x_hsdb	Setup database handle.

Values Returned

`l_vars`	Returns list of variables for the given worst case corner handle.
`nil`	Returns `nil`, if unsuccessful.

Example

```
axlGetWCCVars(1005)
Returns a list of variable names for the specified corner handle, 1005.
```

10

Optimization-Related Functions

SKILL Functions for Optimization

Function	Description
axlGetWYCSigmaTarget Limit	Retrieves the sigma-to-target limit for Improve Yield flow using worst yield corners.
axlSetWYCSigmaTarget Limit	Sets the sigma-to-target limit for Improve Yield flow using worst yield corners.

axlGetWYCSigmaTargetLimit

```
axlGetWYCSigmaTargetLimit(
     )
     => n_sigma_limit
```

Description

Gets the sigma-to-target limit for for Improve Yield flow using worst yield corners. If this value is not set, then the flow internally sets it to 100.

Argument

None.

Value Returned

n_sigma_limit	The sigma-to-target limit for the Improved Yield flow.
nil	Unsuccessful operation.

Example

```
axlGetWYCSigmaTargetLimit()
```

axlSetWYCSigmaTargetLimit

```
axlSetWYCSigmaTargetLimit(
     n_sigma_limit
     )
     => t / nil
```

Description

Sets the sigma-to-target limit for Improve Yield flow using worst yield corners.

Arguments

n_sigma_limit	The sigma-to-target limit.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlSetWYCSigmaTargetLimit(120)
```

11

Run-Related Functions

Run-Related SKILL Functions

Function	Description
axlExportOutputView	Exports the results view to the specified .csv or .html file.
axlGetAllSweepsEnabled	Returns the selection status of the check box associated with the Sweep option in the Run Summary widget. If the check box is not checked or deselected for Sweeps option , then this function will return `nil`.
axlGetCurrentRunMode	Returns the current simulation run mode of a given ADE (G)XL session.
axlGetParasiticRunMode	Gets the parasitic run mode name from the active setup or history checkpoint.
axlGetParasiticParaLCV	Gets the name of the parasitic cellview attached to the parasitic run mode in the active setup or history checkpoint.
axlGetParasiticSchLCV	Gets the name of the schematic cellview attached to the parasitic run mode in the active setup or history checkpoint.
axlGetPreRunScript	Returns the current simulation run mode of a given ADE (G)XL session.
axlGetRunDistributeOptions	Returns the current run option settings for the given setup database.
axlGetRunData	Returns the handle to the history obtained after running a simulation in the given ADE XL session. You can use this handle to get access to the history results or to get setup details by using the history checkpoint.
axlGetRunMode	Returns a handle to the named run mode in the specified setup database.

Run-Related SKILL Functions, *continued*

Function	Description
axlGetRunModes	Returns a list of available run modes from the specified setup database.
axlGetRunOption	Returns a handle to the named run option (`t_runoptName`) in the setup database for the specified run mode (`t_mode`).
axlGetRunOptionName	Returns the run option name.
axlGetRunOptions	Returns a list containing a handle to all run options in the setup database and a list of all run option names for the specified run mode.
axlGetRunOptionValue	Returns the value associated with the provided run option.
axlGetRunStatus	Returns the completion status in terms of the number of points, tests, or corners completed for all histories running in the given ADE XL session or for the specified history.
axlPutRunOption	Adds a run option to the setup database or edits an existing one and returns the handle to the option. The list of valid option names (`t_runoptName`) depends on the run mode (`t_mode`).
axlIsSimUsingStatParams	Returns `t`, if statistical variables are being set or varied for a particular simulation run. For example, statistical parameters in Monte Carlo run or a statistical corner for Improve Yield. Returns `nil` otherwise.
axlRunAllTests	Starts an ADE XL run of all enabled tests.
axlRunAllTestsWithCallback	Starts an ADE XL run of all enabled tests and specifies a SKILL expression to call upon their completion.
axlRunSimulation	Starts an ADE XL run of all enabled tests and specifies a SKILL expression to call upon completion.
axlSetCurrentRunMode	Sets the current simulation run mode for the given database.
axlImportPreRunScript	Imports and attaches the given script to the specified test. In the ADE XL GUI, you can right-click the test name and choose `Add/Edit Pre Run Script` to view or edit the pre-run script attached to the test.
axlSetParasiticRunMode	Sets the parasitic run mode for the given ADE XL setup.
axlSetPreRunScript	Sets or adds a pre-run script for an ADE XL test.

Run-Related SKILL Functions, *continued*

Function	Description
axlSetPreRunScriptEnabled	Enables or disables execution of pre-run scripts before running simulations.
axlSetRunDistributeOptions	Sets the specified run option settings for the given setup database. These settings are also visible in the Run Options form.
axlSetRunOptionName	Sets the run option name.
axlStop	Stops a run based on id..
axlStopAll	Stops all runs currently evaluating in the ADE XL session.
axlReadHistoryResDB	Returns a handle to the ADE XL results database saved with the specified history.
axlReadResDB	Returns a handle to the specified ADE XL results database.
axlSetRunOptionValue	Sets a value for the given run option.

axlExportOutputView

```
axlExportOutputView(
     t_sessionName
     t_fileName
     t_viewType
     [ ?history g_historyName ]
     [ ?testName g_testName ]
     [ ?filterName g_filterName ]
     [ ?clearAllFilters g_clearAllFilters]
     )
     => t / nil
```

Description

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

Arguments

t_sessionName	Name of the ADE Assembler or ADE Explorer session. Default value: current session
t_fileName	Path and name of the file to which results are to be exported.
t_viewType	Name of the output view to be exported. Valid values: ■ "Detail" ■ "Detail - Transpose" ■ "Status" ■ "Summary" ■ "Yield" ■ "Checks/Asserts" ■ "Fault" ■ "Current" Default value: "Current"
?history *g_historyName*	Name of the history for which outputs are to be exported. Default value: " "
?testName *g_testName*	Name of the test for which outputs are to be exported. This argument is useful when you are exporting results from a multi-test cellview. **Note:** This argument is supported only for the Checks/Asserts and Fault result views.

Value Returned

t	The output view is successfully exported to the specified file
nil	Unsuccessful operation

Examples

```
axlExportOutputView(axlGetWindowSession() "./abc.csv" "Yield")

axlExportOutputView(axlGetWindowSession() "./abc.html" "Detail-Transpose")

axlExportOutputView(axlGetWindowSession() "./abcd.html" "Yield" ?history
"Interactive.20")
```

axlGetAllSweepsEnabled

```
axlGetAllSweepsEnabled(
     x_hsdb
     )
     => t / nil
```

Description

Returns the selection status of the check box associated with the Sweep option in the Run Summary widget. If the check box is not checked or deselected for Sweeps option , then this function will return `nil`.

Argument

x_hsdb	Setup database handle.

Value Returned

`t`	Returns `t`, if the Sweep is enabled.
`nil`	Returns `nil`, if the Sweep is disabled.

Example

```
axlGetAllSweepsEnabled 1003
t
```

axlGetCurrentRunMode

```
axlGetCurrentRunMode(
     x_hsdb
     )
     => t_mode / nil
```

Description

Returns the current simulation run mode of a given ADE (G)XL session.

Argument

x_hsdb	SetupDB handle.

Value Returned

t_mode	Valid Values:
	`Single Run, Sweeps and Corners`
	`Monte Carlo Sampling`
	`Global Optimization`
	`Local Optimization`
	`Improve Yield`
	`Sensitivity Analysis`
nil	`Unsuccessful operation`

Examples

Example 1:

The following example returns the run mode set in the current ADE XL session:

```
sdb = axlGetMainSetupDB(axlGetWindowSession())
runMode = axlGetCurrentRunMode(sdb)
"Single Run, Sweeps and Corners"
```

Example 2:

The following example finds the run mode of the given history name:

```
(defun CCRaxlGetRunModeFromHistoryName (sdbh histName)
(let (checkPoint)
checkPoint = (axlGetHistoryCheckpoint (axlGetHistoryEntry sdbh histName))
(axlGetCurrentRunMode checkPoint)
)
)


histName = "MonteCarlo.0"
sess = axlGetWindowSession()
sdbh = (axlGetMainSetupDB sess)
runMode = (CCRaxlGetRunModeFromHistoryName sdbh histName) (printf "Run mode for
history %s = \"%s\"\n" histName runMode)
```

axlGetParasiticRunMode

```
axlGetParasiticRunMode(
     x_mainSDB
     )
     => t_runMode
```

Description

Gets the parasitic run mode name from the active setup or history checkpoint.

Argument

x_mainSDB	Handle to the active setup database or a checkpoint history

Value Returned

t_runMode	Name of the parasitic run mode

Example

The following example returns the name of the parasitic run mode set in the active session:

```
s1 = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001
axlGetParasiticRunMode(1001)
=> "Extracted (Parasitics/LDE)"
```

Related Functions

axlGetMainSetupDB, axlGetWindowSession

axlGetParasiticParaLCV

```
axlGetParasiticParaLCV(
     t_sessionName
     t_paraRunMode
     )
     => t_cellViewName
```

Description

Gets the name of the parasitic cellview attached to the parasitic run mode in the active setup or history checkpoint.

Arguments

t_sessionName	Name of the ADE XL session or checkpoint history. Alternatively, you can provide a handle to the session or checkpoint history.
t_parasiticRunMode	Name of the parasitic run mode.

Value Returned

`t_cellViewName`	Name of the cellview

Example

The following example returns the name of the parasitic cellview attached to the `Extracted` parasitic run mode in the active session:

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001
axlGetParasiticRunMode(x_mainSDB)
=> "Extracted (Parasitics/LDE)"
axlGetParasiticParaLCV(1001 "Extracted (Parasitics/LDE)")
=> "opAmp_test:osc:av_extracted_Ronly"
```

axlGetParasiticSchLCV

```
axlGetParasiticSchLCV(
     t_sessionName
     t_paraRunMode
     )
     => t_cellViewName
```

Description

Gets the name of the schematic cellview attached to the parasitic run mode in the active setup or history checkpoint.

Arguments

t_sessionName	Name of the ADE XL session or checkpoint history. Alternatively, you can provide a handle to the session or checkpoint history.
t_parasiticRunMode	Name of the parasitic run mode.

Value Returned

`t_cellViewName`	Name of the schematic cellview

Example

The following example returns the name of the schematic cellview attached to the `Extracted` parasitic run mode in the active session:

```
s1 = axlGetWindowSession()
=> "session0"
x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001
axlGetParasiticRunMode(x_mainSDB)
=> "Extracted (Parasitics/LDE)"
axlGetParasiticParaLCV(1001 "Extracted (Parasitics/LDE)")
=> "opAmp_test:osc:av_extracted_Ronly"
axlGetParasiticSchLCV(1001 "Extracted (Parasitics/LDE)")
=> "opAmp_test:osc:schematic"
```

axlGetPreRunScript

```
axlGetPreRunScript(
     t_sessionName
     t_testName
     )
     => t_filePath
```

Description

Returns the path to the pre-run script file attached to the given ADE XL test.

Arguments

t_session	Name of the ADE XL session or a handle to it.
t_testName	Name of the test for which you want to get the path to the pre-run script.

Value Returned

t_filePath	Path to the script file. If no script is set for the given test, returns a blank string.

Example

The following example script returns the details of the pre-run script for the `AmpTest1` test:

```
axlsession=axlGetWindowSession( hiGetCurrentWindow())
"session0"
axlGetPreRunScript(axlsession, "AmpTest1")
"./myScript"
```

Related APIs

axlSetPreRunScript, axlSetPreRunScriptEnabled, axlImportPreRunScript

axlGetRunDistributeOptions

```
axlGetRunDistributeOptions(
     x_hsdb
     )
     => r_runOptions / nil
```

Description

Returns the current run option settings for the given setup database.

Argument

x_hsdb	Handle to the setup database.

Value Returned

r_runOptions	Struct of run options specified for the given setup database. This struct contains the following three elements: 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

```
sdb = axlGetMainSetupDB(axlGetWindowSession())
runOpt = axlGetRunDistributeOptions(sdb)
runOpt~>??
(JobLimit 4 DivideJobs Specify RunIn Parallel
)
```

Reference

axlGetRunData

```
axlGetRunData(
     t_sessionName
     x_runID
     )
     => x_historyHandle / nil
```

Description

Returns the handle to the history obtained after running a simulation in the given ADE XL session. You can use this handle to get access to the history results or to get setup details by using the history checkpoint.

Arguments

t_sessionName	Name of the ADE XL session or a handle to it
x_runID	Unique run ID obtained after running a simulation

Value Returned

x_historyHandle	Handle to the history of the simulation run
nil	Unsuccessful operation

Example

The following example code shows how to get the history handle and use it to work with the results:

```
session=axlGetWindowSession()
=> "session0"

runid=axlRunSimulation(?session session)
=> 0
; here, 0 is the run ID

x_historyHandle=axlGetRunData(session 0)
=> 3234

; you can use this history handle to get the details of history or to get results

t_historyName=axlGetHistoryName(x_historyHandle)
=> "Interactive.20"
```

```
; the following function call loads the results for the history in the ADE XL GUI

axlViewHistoryResults(session x_historyHandle)

; the history name can be used to access the results by using SKILL code, as shown
below.

rdbObj=axlReadHistoryResDB(t_historyName ?session session)
=>axlrdb@0x1949a418

; This results data object can be used to access different elements from the results
database for that history. For more details, see the example for axlReadResDB.
```

axlGetRunMode

```
axlGetRunMode(
     x_hsdb
     t_mode
     )
     => x_mode / nil
```

Description

Returns a handle to the named run mode in the specified setup database.

Arguments

x_hsdb	Setup database handle.
t_mode	A valid run mode name.
	You can get the list of valid run mode names by using the axlGetRunModes function.

Value Returned

x_mode	Run mode handle.
nil	Unsuccessful operation.

Example

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>1001
axlGetRunMode( x_mainSDB "Global Optimization" )
1058
```

Reference

axlGetRunModes, axlRunAllTests, axlRunAllTestsWithCallback

axlGetRunModes

```
axlGetRunModes(
     x_mainSDB
     )
     => l_modes / nil
```

Description

Returns a list of available run modes from the specified setup database.

Argument

x_mainSDB	Setup database handle.

Value Returned

l_modes	List of handle to run modes and the run mode names.
nil	Unsuccessful operation.

Example

The following example shows how axlGetRunModes can be used to get the list of all the available run modes:

```
axlGetRunModes( x_mainSDB )
=>(1182
("Local Optimization" "Global Optimization" "Monte Carlo Sampling" "Improve Yield"
"High Yield Estimation" "Sensitivity Analysis" "Create Worst Case Corners" "Single
Run, Sweeps and Corners" "Size Over Corners" )
```

axlGetRunOption

```
axlGetRunOption(
     x_hsdb
     t_mode
     t_runoptName
     )
     => x_runOption / nil
```

Description

Returns a handle to the named run option (*t_runoptName*) in the setup database for the specified run mode (*t_mode*).

Arguments

x_hsdb Setup database handle.

t_mode Run mode.
Valid Values:

`Sampling`

`Global Optimization`

`Local Optimization`

`Monte Carlo Sampling`

t_runoptName Run option name.
Valid Values depend on *t_mode* as follows:

For `Sampling`:

`points` Number of sampling points

For `Global Optimization`:

`tillsatisfied`

Optimization stops when all goals are met

`timelimit` Optimization stops when the program reaches the time limit (in minutes)

`numpoints` Optimization stops when the program reaches the number of points

`ptswithnoimprovement`

Optimization stops when there is no improvement for the number of points

For `Local Optimization`:

`effort` Optimization effort

`tillsatisfied`

Optimization stops when all goals are met

`timelimit` Optimization stops when the program reaches the time limit (in minutes)

`numpoints` Optimization stops when the program reaches the number of points

`ptswithnoimprovement`

Optimization stops when there is no improvement for the number of points

For `Monte Carlo Sampling`:

`mcmethod` Monte Carlo Sampling method

`mcnumpoints`

Number of Monte Carlo sampling points

Note: Typically, this number should be at least the number of statistical variables.

Value Returned

`x_runOption`	Handle to named run option.
`nil`	Unsuccessful operation.

Example

```
runopth = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "savemismatch")
axlGetRunOptionValue(runopth)
```

Reference

`axlGetRunOptionName`, `axlSetRunOptionName`

axlGetRunOptionName

```
axlGetRunOptionName(
     x_runOption
     )
     => t_runoptName / nil
```

Description

Returns the run option name.

Argument

x_runOption	Run option handle.

Value Returned

t_runoptName	Run option name.
nil	Unsuccessful operation.

Example

```
runopth = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "savemismatch")
=> 1035
axlGetRunOptionName( runopth )
=> "savemismatch"
```

Reference

axlGetRunOption, axlPutRunOption

axlGetRunOptions

```
axlGetRunOptions(
     x_hsdb
     t_runModeName
     )
     => l_list / nil
```

Description

Returns a list containing a handle to all run options in the setup database and a list of all run option names for the specified run mode.

Arguments

x_hsdb	Setup database handle.
t_runModeName	Name of the run mode for which you want to get run options. The name of the run mode must be same as it is displayed in the *Run Mode* list in ADE XL GUI.

Value Returned

l_list	List containing a handle to all the run options in the setup database and a list of all run option names.
nil	Unsuccessful operation.

Example

The following example code shows how to get the names of run options for the Monte Carlo run mode:

```
session = axlGetWindowSession()
x_mainSDB = axlGetMainSetupDB(session)
axlGetRunOptions( x_mainSDB "Monte Carlo Sampling")

(1452 ("dutsummary" "ignoreflag" "mcmethod" "mcnumpoints" "mcnumbins"
    "mcStopEarly" "mcStopMethod" "samplingmode" "saveprocess" "savemismatch"
    "mcreferencepoint" "donominal" "saveallplots" "montecarloseed"
    "mcstartingrunnumber" "mcYieldTarget" "mcYieldAlphaLimit")
)
; you can further get the value of one of the run options, as shown below.
x_runOption = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "mcnumpoints")
```

```
axlGetRunOptionValue(x_runOption)
"200"
```

axlGetRunOptionValue

```
axlGetRunOptionValue(
     x_runOption
     )
     => t_runoptValue / nil
```

Description

Returns the value associated with the provided run option.

Argument

x_runOption	Run option handle.

Value Returned

t_runoptValue	Value of the given run option.
nil	Unsuccessful operation.

Example

The following example code shows how you can view the value for a Monte Carlo run option:

```
session = axlGetWindowSession()
x_mainSDB = axlGetMainSetupDB(session)
axlGetRunOptions( x_mainSDB "Monte Carlo Sampling")

(1452
    ("dutsummary" "ignoreflag" "mcmethod" "mcnumpoints" "mcnumbins"
    "mcStopEarly" "mcStopMethod" "samplingmode" "saveprocess" "savemismatch"
    "mcreferencepoint" "donominal" "saveallplots" "montecarloseed"
    "mcstartingrunnumber" "mcYieldTarget" "mcYieldAlphaLimit"
    )
)
x_runOption = axlGetRunOption(1001 "Monte Carlo Sampling" "mcnumpoints")
axlGetRunOptionValue(x_runOption)
"200"
```

Reference

axlGetRunOption, axlPutRunOption

axlGetRunStatus

```
axlGetRunStatus(
     t_sessionName
     [ ?optionName t_optionName ]
     [ ?historyName t_historyName ]
     )
     => l_statusValues
```

Description

Returns the completion status in terms of the number of points, tests, or corners completed for all histories running in the given ADE XL session or for the specified history.

Arguments

t_sessionName Name of the session

?optionName *t_optionName*

A value that specifies the elements for which the status value is to be returned.

Valid values:

`all`: Returns the number of points for which simulation is complete and the total number of points.

`Tests`: Returns the number of tests for which simulation is complete and the total number of tests.

`Corners`: Returns number of corners for which simulation is complete and the total number of corners.

Default value: `all`

?historyName *t_historyName*

Name of the history item for which the status is to be returned.

Default value: `""`

If this argument is not provided, ADE XL returns the completion status for the given ADE XL session. In this case, the total number of points, tests, or corners in the returned value is the total number of points, tests, or corners run across all the histories running in the given ADE XL session.

Value Returned

l_statusValues The returned list contains the following two values:

- The number of points, tests, or corners completed
- The total number of points, tests, or corners to be run in the given ADE XL session

Example

The following example code shows how the completion status for an ADE XL run is returned:

```
axlSession=axlGetWindowSession( hiGetCurrentWindow())
=> "session0"
axlGetRunStatus("session0")
=> (4 14)
; The returned value suggests that out of the total 14 points to be run in the
session session0, simulations are complete for 4 points.
;
axlGetRunStatus("session0" ?historyName "Interactive.10" ?optionName "tests")
=> (1 2)
; The returned value suggests that out of the total two tests in the Interactive.10
history of session0, simulations are complete for one test.
;
axlGetRunStatus("session0" ?historyName "Interactive.10" ?optionName "corners")
=> (4 8)
; The returned value suggests that out of the total eight corners to be run in the
Interactive.10 history of session0, simulations are complete for four corners.
```

axlIsSimUsingStatParams

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

Description

Returns `t`, if statistical variables are being set or varied for a particular simulation run. For example, statistical parameters in Monte Carlo run or a statistical corner for Improve Yield. Returns `nil` otherwise.

Argument

None.

Value Returned

t	Successful operation
`nil`	Unsuccessful operation.

axlPutRunOption

```
axlPutRunOption(
     x_hsdb
     t_mode
     t_runoptName
     )
     => x_runOption / nil
```

Description

Adds a run option to the setup database or edits an existing one and returns the handle to the option. The list of valid option names (*t_runoptName*) depends on the run mode (*t_mode*).

Note: Any unsupported option names you specify will have unspecified effects on the behavior of the run mode. There are no run options for the `Single Run, Sweeps and Corners` run mode.

Arguments

x_hsdb Setup database handle.

t_mode Run mode.
Valid Values:

`Sampling`

`Global Optimization`

`Local Optimization`

`Monte Carlo Sampling`

t_runoptName Run option name.
Valid Values depend on *t_mode* as follows:

For `Sampling`:

`points` Number of sampling points

For `Global Optimization`:

`tillsatisfied`

Optimization stops when all goals are met

`timelimit` Optimization stops when the program reaches the time limit (in minutes)

`numpoints` Optimization stops when the program reaches the number of points

`ptswithnoimprovement`

Optimization stops when there is no improvement for the number of points

For `Local Optimization`:

`effort` Optimization effort

`tillsatisfied`

Optimization stops when all goals are met

`timelimit` Optimization stops when the program reaches the time limit (in minutes)

`numpoints` Optimization stops when the program reaches the number of points

`ptswithnoimprovement`

Optimization stops when there is no improvement for the number of points

For `Monte Carlo Sampling`:

`mcmethod` Monte Carlo Sampling method

`mcnumpoints`

Number of Monte Carlo sampling points

Note: Typically, this number should be at least the number of statistical variables.

Value Returned

`x_runOption`	Handle to run option.
`nil`	Unsuccessful operation.

Example

```
axlPutRunOption( 1004 "Monte Carlo Sampling" "points")
1048
```

Reference

`axlGetRunOption`, `axlGetRunOptions`, `axlRunAllTests`, `axlRunAllTestsWithCallback`

axlRunAllTests

```
axlRunAllTests(
     t_session
     t_mode
     )
     => x_runid / nil
```

Description

Starts an ADE XL run of all enabled tests.

Arguments

t_session	Session name.

t_mode	Run mode.
	Valid Values:
	`Single Run, Sweeps and Corners`
	`Sampling`
	`Global Optimization`
	`Local Optimization`
	`Monte Carlo Sampling`

Value Returned

x_runid	Run ID.
`nil`	Unsuccessful operation.

Example

```
axlRunAllTests( "session0" "Single Run, Sweeps and Corners" )
1
```

axlRunAllTestsWithCallback

```
axlRunAllTestsWithCallback(
     t_session
     t_mode
     t_callback
     )
     => x_runid / nil
```

Description

Starts an ADE XL run of all enabled tests and specifies a SKILL expression to call upon their completion.

Arguments

t_session	Session name.

t_mode	Run mode.
	Valid Values:
	`Single Run, Sweeps and Corners`
	`Sampling`
	`Global Optimization`
	`Local Optimization`
	`Monte Carlo Sampling`

t_callback	SKILL expression for callback.

Value Returned

x_runid	Run ID.
`nil`	Unsuccessful operation.

Example

```
axlRunAllTestsWithCallback( ( axlCreateSession "data_session" ) "Single Run,
Sweeps and Corners" "( "printf(\"run complete\")" )
=> 1001
```

Reference

axlCreateSession

axlRunSimulation

```
axlRunSimulation(
     [ ?session t_session ]
     [ ?callback t_callback ]
     )
     => x_runid / nil
```

Description

Starts an ADE XL run of all enabled tests and specifies a SKILL expression to call upon completion.

Arguments

`?session` *`t_session`*	Session name.
`?callback` *`t_callback`*	SKILL expression for callback.

Value Returned

`x_runid`	Run ID.
`nil`	Unsuccessful operation.

Example

Example 1:

In the following example, ADE XL runs simulation for all enabled tests in the current session.

```
axlRunSimulation()
```

Example 2:

In the following example, ADE XL runs simulation for all enabled tests in the current session. After the simulation is complete, the function will also execute the callback script provided by the callback argument.

```
axlRunSimulation( ?session "session0" ?callback "printf(\"Run Complete\")")
```

axlSetCurrentRunMode

```
axlSetCurrentRunMode(
     x_hsdb
     t_mode
     )
     => t / nil
```

Description

Sets the current simulation run mode for the given database.

Arguments

x_hsdb Setup database handle.

t_mode Run mode.

Valid Values for ADE Explorer:

```
"Single Run, Sweeps and Corners"
```

Valid Values for ADE Assembler:

```
"Single Run, Sweeps and Corners"
"Monte Carlo Sampling"
"Global Optimization"
"Local Optimization"
"Improve Yield"
"High Yield Estimation"
"Sensitivity Analysis"
"Feasibility Analysis"
"Worst Case Corners"
"Manual Tuning"
"Size Over Corners"
"Run Plan"
"Fault Simulation"
```

Valid Values for ADE XL and ADE GXL:

```
"Single Run, Sweeps and Corners"
"Monte Carlo Sampling"
"Global Optimization"
"Local Optimization"
"Improve Yield"
"High Yield Estimation"
"Sensitivity Analysis"
"Feasibility Analysis"
"Worst Case Corners"
"Manual Tuning"
"Size Over Corners"
"Manual Optimization
"Conjugate Gradient Optimization"
"Feasibility Analysis"
"Worst Case Corners"
"Manual Tuning"
"Size Over Corners"
```

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
sdb = axlGetMainSetupDB(axlGetWindowSession())
runMode = "Monte Carlo Sampling"
axlSetCurrentRunMode(sdb runMode)
```

axlImportPreRunScript

```
axlImportPreRunScript(
     t_sessionName
     t_testName
     t_preRunScriptName
     )
     => t_preRunScriptName / nil
```

Description

Imports and attaches the given script to the specified test. In the ADE XL GUI, you can right-click the test name and choose *Add/Edit Pre Run Script* to view or edit the pre-run script attached to the test.

Note: If you make any changes to the test script in the *Add/Edit Pre Run Script* form, the original script is not modified. Instead, the changes are saved in a new script, which is then attached to the test.

Arguments

t_session	Name of the ADE XL session.
t_testName	Name of the test to which the imported script is attached.
t_preRunScriptName	Location path and name of the script to be imported.

Value Returned

t_preRunScriptName	Name of the pre-run script imported for the given test.
nil	Unsuccessful operation.

Example

```
axlsession=axlGetWindowSession( hiGetCurrentWindow())
=> "session0"
axlImportPreRunScript("session0", "AmpTest1", "./myscript")
=> "myscript"
```

Related Function

axlSetPreRunScriptEnabled

axlSetParasiticRunMode

```
axlSetParasiticRunMode(
     x_mainSDB
     t_runModeName

     )
     => t / nil
```

Description

Sets the parasitic run mode for the given ADE XL setup.

Arguments

x_mainSDB	Handle to the setup database

t_runModeName	Run mode to be set
	Possible values:
	`"No Parasitics/LDE"`
	`"Schematic Estimates (Parasitics)"`
	`"Extracted (Parasitics/LDE)"`
	`"Layout(Parasitics/LDE)"`

Value Returned

`t`	Successful operation
`nil`	Unsuccessful operation

Example

The following example returns the name of the parasitic run mode set in the active session:

```
axlSession=axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( axlSession )
=> 1001
axlGetParasiticRunMode(x_mainSDB "Extracted (Parasitics/LDE)")
```

Related Functions

axlGetMainSetupDB, axlGetWindowSession, axlGetParasiticRunMode

axlSetPreRunScript

```
axlSetPreRunScript(
     t_sessionName
     t_testName
     g_scriptName
     )
     => t_filePath / nil
```

Description

Sets or adds a pre-run script for an ADE XL test.

Arguments

t_session	Name of the ADE XL session.
t_testName	Name of the test for which you want to set or add a pre-run script.
t_scriptName	Path to the file that contains the script.

Value Returned

t_filePath	Path to the script file.
nil	Unsuccessful operation.

Example

The following example script adds a pre-run script for the `AmpTest1` test:

```
axlsession=axlGetWindowSession( hiGetCurrentWindow())
=> "session0"
axlSetPreRunScript(axlsession, "AmpTest1", "./myScript")
=> "./myScript"
axlSetPreRunScriptEnabled(axlsession, "AmpTest1", t)
=> t
```

Related APIs

axlGetPreRunScript, axlSetPreRunScriptEnabled

axlSetPreRunScriptEnabled

```
axlSetPreRunScriptEnabled(
     t_sessionName
     t_testName
     g_enabled
     )
     => t / nil
```

Description

Enables or disables execution of pre-run scripts before running simulations.

Arguments

t_session	Name of the ADE XL session.
t_testName	Name of the test for which you want to enable or disable execution of a pre-run script.
g_enabled	Specifies if the execution of a pre-run script is to be enabled or disabled.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example script enables the execution or pre-run script for the `AmpTest1` test:

```
axlsession=axlGetWindowSession( hiGetCurrentWindow())
"session0"
axlSetPreRunScriptEnabled(axlsession, "AmpTest1", t)
t
```

Related APIs

axlGetPreRunScript, axlSetPreRunScriptEnabled, axlImportPreRunScript

axlSetRunDistributeOptions

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

Description

Sets the specified run option settings for the given setup database. These settings are also visible in the Run Options form.

Arguments

x_hsdb	Handle to the setup database.
?RunIn *t_runIn*	Describes how multiple simulations need to run. Valid values: `Parallel`, `Serial`.
?DivideJobs *t_divideJobs*	Specifies how the ICRPs can be 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 run options are set successfully
`nil`	Unsuccessful operation

Example

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

```
sdb = axlGetMainSetupDB(axlGetWindowSession())
axlSetRunDistributeOptions(sdb ?RunIn 'Parallel ?DivideJobs 'Specify ?JobLimit 3)
=> t
```

Reference

axlGetRunDistributeOptions

axlSetRunOptionName

```
axlSetRunOptionName(
     x_runOption
     t_runoptName
     )
     => t / nil
```

Description

Sets the run option name.

Arguments

x_runOption	Run option handle.
t_runoptName	Run option name.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
runopth = axlGetRunOption( x_mainSDB "Monte Carlo Sampling" "savemismatch")
=> 1048
axlSetRunOptionName( runopth "points" )
t
```

Reference

axlGetRunOption, axlGetRunOptionName, axlGetRunOptions

axlStop

```
axlStop(
     t_session
     x_runid
     )
     => t / nil
```

Description

Stops a run based on id..

Arguments

t_session	Session Name.
x_runid	Run Id.

Value Returned

t	Successful Operation.
nil	Unsuccessful Operation.

Example

```
sess=axlGetWindowSession(window(3))
=> "session0"
id = (axlRunAllTests sess "Global Optimization")
=> 0
axlStop(sess id)
=> t
```

axlStopAll

```
axlStopAll(
     t_session
     )
     => t / nil
```

Description

Stops all runs currently evaluating in the ADE XL session.

Argument

t_session	ADE XL session name.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
sess=axlGetWindowSession(window(3))
=> "session0"
axlStopAll( sess)
t
```

axlReadHistoryResDB

```
axlReadHistoryResDB(
     t_historyName
     [ ?session t_sessionName ]
     )
     => h_ResultsDBObj / nil
```

Description

Returns a handle to the ADE XL results database saved with the specified history.

This handle is similar to the handle that is returned by the axlReadResDB function, You can use this handle to access various database objects in the result. For more details, refer to axlReadResDB.

Arguments

t_historyName	Name of the history.
?session *t_session-Name*	Name of the session.

Values Returned

h_ResultsDBObj	Handle to the results database.
nil	Returns nil otherwise.

Example

The following code returns a handle rdbHandle to the results database for a history named CornerResults:

```
sess=axlGetWindowSession(window(3))
rdbHandle=axlReadHistoryResDB("CornerResults" ?session sess)
=>axlrdb@0x1949a418
```

axlReadResDB

```
axlReadResDB(
     t_ResultsDBFileName
     )
     => h_ResultsDBObj / nil
```

Description

Returns a handle to the specified ADE XL results database.

This handle provides read-only access to the results database that contains objects of the following five types:

- point - a design point
- corner - a corner defined for a particular design point
- test - a test defined for a particular corner
- param - a parameter defined for a particular corner
- output - an output defined for a particular test

There is a hierarchical relationship between the instances of these objects. For example, a point is associated with one or more corners. Each corner is associated with one or more tests. Each test is associated with zero or more outputs, and so on. As a result of this relationship, for an object, you can access the properties of the object itself and other objects related to it.

Each object has:

- Three properties: `name`, which returns the name of the object; `value`, which returns its value; and a property that returns ID of the parent object. For example, an object of type corner has a property `pointID` that returns the ID of the parent point object.
- A set of member functions that return the instances of that object type and other related types. For example, using an instance of type output, you can get the value of an output object and its parent test instance. Using the functions given for a test instance, you can get a corner instance. For a corner instance, you can get the parameters which were used to generate the output, as well as the point and its ID.

In addition, the result database provides a function `help()` for each object of the above mentioned types to displays the list and description of functions that can be called using that particular object. For example, function call `help('corner)` displays a list of all the

functions that can be called using an object of type corner. `help('all)` displays help for all the object types.

Note: Alternatively, you can also call the axlReadHistoryResDB function to return a handle to the specified ADE XL results database. For more details, see .

Argument

t_ResultsDBFileName	Name of the results database file.

Values Returned

h_ResultsDBObj	Handle to the results database.
`nil`	Returns `nil` otherwise.

Example 1

The following code returns a handle `rdb` to the results database for a history named `CornerResults`:

```
historyname="Interactive.1"
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>
x_history=axlGetHistoryEntry(x_mainSDB historyname)
=>
rdbPath=axlGetHistoryResults(x_history)
=>
rdb=axlReadResDB(rdbPath)
=> axlrdb@0x1949a438
; the returned value is a handle to the results database
```

Example 2

The following code opens the results database and displays built-in help:

```
rdb->help()
```

The help is displayed in the CIW, as shown in the following figure:

```
Toplevel Help:

Functions:
  corner(t_cornerName x_pointID) => o_cornerInst
    Returns the corner indicated by name and point ID.
  corners([?name t_cornerName] [?point x_pointID] [?sortBy 'name|'point]) => l_cornerInst
    Returns list of corner instances, which may be narrowed by supplying corner name or point ID.
  help(['point|'corner|'test|'output|'param|'all]) => t
    Displays help for a particular instance type ("all" for all types)
  output(t_outputName t_testName t_cornerName x_pointID) => o_outputInst
    Returns the output instance for a given test, point, and corner.
  outputs([?type 'expr|'signal|'devCheck] [?sortBy 'name|'point|'corner|'test|'value|'type]) => l_outputInst
    Returns list of output instances, which may be narrowed by supplying the output type.
  param(t_paramName  t_cornerName x_pointID) => o_paramInst
    Returns a parameter instance for a given point and corner.
  params([?name t_name] [?corner t_cornerName] [?point x_pointID] [?type 'fixed|'design|'corner] [?sortBy 'name|'pc
    Returns the list of parameter instances, which may optionally be filtered by type.
  point(x_pointID) => o_pointInst
    Returns a specific point instance for a given point ID.
  points([?point x_pointID]||?limit x_numBestPoints]||?sortBy 'id|'best]) => l_pointInst
    Returns a list of all point instances, which may optionally be limited to a subset of best points.
  test(t_name t_cornerName x_pointID) => o_testInst
    Returns a specific test instance.
  tests([?point x_pointID] [?corner t_cornerName] [?name t_name] [?sortBy 'name|'point|'corner]) => l_testInst
    Returns list of test instances, which may be narrowed by supplying the point ID, corner name, or test name.

t
```

Example 3

The example code given below shows how to use the handle to the results database to explore the result values. The handle to the first point in the results database is obtained to display the outputs of type expression with their values. The results are sorted by corner.

```
historyname="Interactive.1"
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001

; returns handle to the setup database of the current ADE XL session

x_history=axlGetHistoryEntry(x_mainSDB historyname)
=>

; returns handle to the given history
rdbPath=axlGetHistoryResults(x_history)
=>
rdb=axlReadResDB(rdbPath)
=> axlrdb@0x1949a438

; The following statement 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)

)
```

Example 4

The following functions calls describe how to access test objects in a result database:

```
; The following function lists all test instances and their properties.
rdb->tests()

; The following function returns a test instance with the given test name, from
; the given corner for point 1.
tst = rdb->test(<t_testName> <t_cornerName> 1)


; The following statement returns the execution host for the test instance.
tst->host

; The following statement returns the run status for the test
tst->status
; The following statement returns the start time and stop time the test in the
; default format, such as 'Mon Sep  9 22:25:01 EDT 2013'

Test->startTime()
Test->stopTime()

; You can change the format in which the start time and stop time is displayed. For
; this, specify the format as shown below. You can change the fromat, as
appropriate.
tst->startTime("h:m:s ap")
; Time would be displayed in this format: : '10:25:1 pm'
tst->stopTime("h:m:ss:zzz ap")
; Time would be displayed in this format: : '12:25:01:035 pm'

; The following code gets the parent corner for the test and returns the parameters
;(sorted by name) for that corner and their values.


foreach(mapcar p tst->corner()->params(?sortBy 'name)
    list( p->name p->value)
)
```

axlSetRunOptionValue

```
axlSetRunOptionValue(
     x_runOptionHandle
     t_runOptionValue
     )
     => t / nil
```

Description

Sets a value for the given run option.

Arguments

x_runOptionHandle	Handle to the run option for which you need to set a value. To find the valid names of run options for a run mode, use the axlGetRunOptions function.
t_runOptionValue	Value of the given run option.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

Example 1

The following example code shows how you can view and change the value for a Monte Carlo run option:

```
session = axlGetWindowSession()
x_mainSDB = axlGetMainSetupDB(session)
axlGetRunOptions( x_mainSDB "Monte Carlo Sampling")

=> (1452
    ("dutsummary" "ignoreflag" "mcmethod" "mcnumpoints" "mcnumbins"
    "mcStopEarly" "mcStopMethod" "samplingmode" "saveprocess" "savemismatch"
    "mcreferencepoint" "donominal" "saveallplots" "montecarloseed"
    "mcstartingrunnumber" "mcYieldTarget" "mcYieldAlphaLimit"
    )
)
```

```
x_runOption = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "mcnumpoints")
axlGetRunOptionValue(x_runOption)
=> "200"
;; changing the value of mcnumpoints to 400

axlSetRunOptionValue(x_runOption "400")
=> t
```

Example 2

```
;; setting a value for the savemismatch option for Monte Carlo
(axlSetRunOptionValue (axlPutRunOption (axlGetActiveSetup (axlGetMainSetupDB (
axlGetWindowSession ))) "Monte Carlo Sampling" "savemismatch") "1")
```

Reference

axlGetRunOptions

12

History-Related Functions

History-Related SKILL Functions

Function	Description
axlGetCurrentHistory	Returns the internal integer value representing the current history entry in active use.
axlGetHistory	Returns a list containing a handle to all history entries in the setup database and a list of all the history entries.
axlGetHistoryCheckpoint	Returns a handle to the checkpoint of a history entry.
axlGetHistoryEntry	Finds a history entry in the setup database and returns a handle to that entry.
axlGetHistoryGroup	Returns a handle to the named history group in the setup database.
axlGetHistoryLock	Returns the lock status of the given history. When a history item is locked, the corresponding setup details and results cannot be deleted.
axlGetHistoryName	Returns the name of the history item that holds the data for the latest simulation run.
axlGetHistoryPrefix	Returns the current history prefix value from the given ADE XL session. The prefix value depends on the run mode selected in the session.
axlGetHistoryResults	Gets the results database from a history entry. This function calls `axlGetResultsLocation` to get the results location.
axlGetOverwriteHistory	Returns a boolean value specifying the status of the Overwrite History option for the active setup.
axlGetOverwriteHistoryName	Returns the name of the history that is set to be overwritten for the active setup.

History-Related SKILL Functions, *continued*

Function	Description
axlLoadHistory	Copies the setup database branch and returns the handle to the copy.
axlSetHistoryLock	Locks the specified checkpoint history. After it is locked, you cannot delete the history or the simulation data saved for it.
axlSetHistoryName	Sets a new name for the specified history.
axlSetHistoryPrefixInPreRunTrigger	Locks the specified checkpoint history. After it is locked, you cannot delete the history or the simulation data saved for it.
axlSetOverwriteHistory	Sets the Overwrite History option for the active setup.
axlSetOverwriteHistoryName	Sets the name of the history to be overwritten for the specified active setup.
axlOpenResDB	Opens the database file specified by `t_fileName`. If the file does not exist, it is created.
axlPutHistoryEntry	Inserts or finds a history entry in the setup database and returns a handle to that entry.
axlReEvaluateHistory	Re-evaluates the history in the specified mode with respect to the active setup for a given session.
axlRemoveSimulationResults	Removes the simulation results data for the given history. The function removes only the results saved by the simulator. The ADE XL results database and the history item is not removed.
axlRestoreHistory	Sets the given history as active setup in the given ADE XL session.
axlViewHistoryResults	Display the results for the specified history item on the Results tab of the given ADE XL session.
axlWriteMonteCarloResultsCSV	Writes the results of the given Monte Carlo run history in CSV format. ADE XL saves the results for each corner in a separate .csv file.

axlGetCurrentHistory

```
axlGetCurrentHistory(
     t_sessionName
     )
     => x_historyHandle / nil
```

Description

Returns the internal integer value representing the current history entry in active use.

Argument

t_sessionName	ADE XL session name.

Value Returned

x_historyHandle	Integer value representing the handle to the currently active history entry.
nil	Unsuccessful operation.

Example

```
sess=axlGetWindowSession()
"session0"
axlGetCurrentHistory( "session0" )
1002
```

axlGetHistory

```
axlGetHistory(
     x_hsdb
     )
     => l_history / nil
```

Description

Returns a list containing a handle to all history entries in the setup database and a list of all the history entries.

Argument

x_hsdb	Setup database handle.

Value Returned

l_history	List containing a handle to all history entries in the setup database and a list of all the history entries.
nil	Unsuccessful operation.

Example

```
sess=axlGetWindowSession()
"session0"
sdb=axlGetMainSetupDB(sess)
1001
axlGetHistory(sdb)
(1045 ("Interactive.0" "Interactive.1"))
```

axlGetHistoryCheckpoint

```
axlGetHistoryCheckpoint(
     x_history
     )
     => x_checkpoint / nil
```

Description

Returns a handle to the checkpoint of a history entry.

Argument

x_history	Handle to a history entry.

Value Returned

x_checkpoint	Handle to a checkpoint.
nil	Unsuccessful operation.

Example

Example 1:

The following example checks for the existence of a history named data_design_verification and then loads the history:

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlLoadHistory( data_sdb )
=>1203
```

Example 2:

The following example finds the run mode of the given history name:

```
(defun CCRaxlGetRunModeFromHistoryName (sdbh histName)
(let (checkPoint)
checkPoint = (axlGetHistoryCheckpoint (axlGetHistoryEntry sdbh histName))
(axlGetCurrentRunMode checkPoint)
)
)


histName = "MonteCarlo.0"
sess = (axlGetWindowSession)
```

```
sdbh = (axlGetMainSetupDB sess)
runMode = (CCRaxlGetRunModeFromHistoryName sdbh histName) (printf "Run mode for
history %s = \"%s\"\n" histName runMode)
```

Reference

axlCreateSession, axlSetMainSetupDB, axlLoadHistory, axlGetHistoryEntry

axlGetHistoryEntry

```
axlGetHistoryEntry(
     x_hsdb
     t_historyName
     )
     => x_history / nil
```

Description

Finds a history entry in the setup database and returns a handle to that entry.

Arguments

x_hsdb	Setup database handle.
t_historyName	History entry name.

Value Returned

x_history	Handle to a history entry.
nil	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
if( axlGetHistoryEntry( data_sdb "data_design_verification" )==0
error( "Failed to get history item named 'data_design_verification'" ) )
1004
```

Reference

axlSetMainSetupDB, axlCreateSession

axlGetHistoryGroup

```
axlGetHistoryGroup(
     x_hsdb
     t_histgrpName
     )
     => x_history / nil
```

Description

Returns a handle to the named history group in the setup database.

Arguments

x_hsdb	Setup database handle.
t_histgrpName	History group name.

Value Returned

x_history	Handle to history group.
nil	Unsuccessful operation.

Example

```
axlGetHistoryGroup(1048 "ImproveYield.1")
2096
```

axlGetHistoryLock

```
axlGetHistoryLock(
     x_historyHandle
     )
     => t / nil
```

Description

Returns the lock status of the given history. When a history item is locked, the corresponding setup details and results cannot be deleted.

Arguments

x_historyHandle	Handle to a history in the ADE XL setup database

Value Returned

`t`	Specifies that the given history is locked in the database
`nil`	Specifies that the given history is not locked in the database

Example

The following example shows how to get the lock status of the current history:

```
session=axlGetWindowSession()
=> "session1"
history1=axlGetCurrentHistory(session)
=> 1067
axlGetHistoryLock(history1)
=> t
```

The following example shows how to get the lock status of the given checkpoint history:

```
session=axlGetWindowSession()
=> "session1"
x_mainSDB=axlGetMainSetupDB(session)
=> 2468
handleHistory=axlGetHistoryCheckpoint( axlGetHistoryEntry(x_mainSDB
"Interactive.1"))
=> 1067
axlGetHistoryLock(handleHistory)
=> nil
```

```
; The returned value nil shows that the Interactive.1 history is currently unlocked
; in the database.
```

Related Functions

axlGetCurrentHistory, axlGetHistoryCheckpoint, axlSetHistoryLock

axlGetHistoryName

```
axlGetHistoryName(
     x_historyEntry
     )
     => t_historyName / nil
```

Description

Returns the name of the history item that holds the data for the latest simulation run.

Argument

x_historyEntry	Integer argument representing the history entry.

Value Returned

t_historyName	String value representing the name of the history item.
nil	Unsuccessful operation.

Example

```
axlGetHistoryName( axlGetCurrentHistory( "session0" ) )
"Interactive.0"
```

axlGetHistoryPrefix

```
axlGetHistoryPrefix(
     x_sessionName
     )
     => t_historyPrefix / nil
```

Description

Returns the current history prefix value from the given ADE XL session. The prefix value depends on the run mode selected in the session.

Argument

t_sessionName	Name of the ADE XL session

Value Returned

t_historyPrefix	The current history prefix value in the ADE XL session
nil	Unsuccessful operation.

Example

The following example shows how to get the current history prefix from the ADE XL session:

```
session=axlGetWindowSession()
=> "session1"
axlGetHistoryPrefix(session)
=> "Interactive"
```

Related Functions

axlGetWindowSession

axlGetHistoryResults

```
axlGetHistoryResults(
     x_history
     )
     => t_results / nil
```

Description

Gets the results database from a history entry. This function calls axlGetResultsLocation to get the results location.

Argument

x_history	Handle to a history entry.

Value Returned

t_results	Results database.
nil	Unsuccessful operation.

Example

```
data_session = axlCreateSession( "data_session" )
design_data = axlRunAllTestsWithCallback( data_session "Single Run, Sweeps and
Corners" "( callbackProcedure )" )
…
axlGetHistoryResults( axlGetRunData( data_session design_data ) )
"Interactive.0.rdb"
```

Reference

axlCreateSession, axlRunAllTestsWithCallback

axlGetOverwriteHistory

```
axlGetOverwriteHistory(
     x_history
     )
     =>t / nil
```

Description

Returns a boolean value specifying the status of the Overwrite History option for the active setup.

Arguments

`x_history`	Specifies a handle to the active setup.

Value Returned

`t`	If the overwrite history is enabled.
`nil`	Returns `nil` otherwise.

Example

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))
=>2417
axlGetOverwriteHistory(x_activeSetup)
=>t
axlGetOverwriteHistoryName(x_activeSetup)
=>"Next History Run"
```

axlGetOverwriteHistoryName

```
axlGetOverwriteHistoryName(
     x_setup
     )
     =>t_historyName / nil
```

Description

Returns the name of the history that is set to be overwritten for the active setup.

Argument

x_setup	Handle to the active setup.

Value Returned

t_historyName	Name of the history set to be overwritten.
nil	Otherwise.

Example

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))
=>2417
axlGetOverwriteHistory(x_activeSetup)
=>t
axlGetOverwriteHistoryName(x_activeSetup)
=>"Next History Run"
```

axlLoadHistory

```
axlLoadHistory(
     x_to
     x_from
     )
     => x_hsdb / nil
```

Description

Copies the setup database branch and returns the handle to the copy.

Arguments

x_to	Handle to target (copied) setup database.
x_from	Handle to source setup database branch.

Value Returned

x_hsdb	Handle to copied setup database.
nil	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlLoadHistory( data_sdb
axlGetHistoryCheckpoint( axlGetHistoryEntry( data_sdb "data_design_verification" )
) )
1050
```

Reference

axlCreateSession, axlGetHistoryEntry, axlSetMainSetupDB

axlSetHistoryLock

```
axlSetHistoryLock(
     x_handleHistory
     g_enable
     )
     => t / nil
```

Description

Locks the specified checkpoint history. After it is locked, you cannot delete the history or the simulation data saved for it.

Arguments

x_handleHistory	Handle to the history in the setup database for which the lock status is to be changed
g_enable	The lock status to be set for the specified history Valid values: `t` or `nil`

Value Returned

`t`	When the specified history is locked successfully
`nil`	Unsuccessful operation

Example

The following example shows how to lock the current history:

```
session=axlGetWindowSession()
=> "session1"
handleHistory=axlGetCurrentHistory(session)
=> 1067
axlSetHistoryLock(handleHistory t)
=> t
```

The following example shows how to set the lock for the given checkpoint history:

```
session=axlGetWindowSession()
=> "session1"
x_mainSDB=axlGetMainSetupDB(session)
```

```
=> 2468
handleHistory=axlGetHistoryCheckpoint( axlGetHistoryEntry(x_mainSDB
"Interactive.1"))
=> 1067
axlSetHistoryLock(handleHistory t)
=> t
; The returned value t shows that the Interactive.1 history has been locked
; in the database.
```

Related Functions

axlGetCurrentHistory, axlGetHistoryCheckpoint, axlGetHistoryLock

axlSetHistoryName

```
axlSetHistoryName (
     x_historyHandle
     t_newHistoryName
     )
     => t / nil
```

Description

Sets a new name for the specified history.

Arguments

x_historyHandle	Handle to the history for which you need to change the name
t_newHistoryName	New name to be set for the history

Value Returned

`t`	History name is changed successfully
`nil`	Unsuccessful operation

Example

The following example shows how to rename a history:

```
session=axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
historyHandle=axlGetHistoryEntry(x_mainSDB "SingleRun.1")
=> 4127
axlSetHistoryName( historyHandle "newHistoryName")
=> t
```

axlSetHistoryPrefixInPreRunTrigger

```
axlSetHistoryPrefixInPreRunTrigger (
     t_session
     t_historyPrefix
     )
     => t_historyPrefix / nil
```

Description

Sets a prefix to be used in the history name for a new run.

Arguments

t_session	Name of the current ADE XL session
t_historyPrefix	Prefix to be used for the new history

Value Returned

t_historyPrefix	The prefix value successfully set by this function
nil	Unsuccessful operation

Example

The following example shows how to set the history name before running a simulation:

```
session=axlGetWindowSession()
=> "session0"

axlSetHistoryPrefixInPreRunTrigger( session "newName")
=> "newName"

axlRunSimulation()
```

axlSetOverwriteHistory

```
axlSetOverwriteHistory(
     x_setup
     g_overwriteStatus
     )
     =>t / nil
```

Description

Sets the Overwrite History option for the active setup.

Arguments

x_setup	Handle to the active setup.
g_overwriteStatus	Status to be set to enable or disable overwrite history for the specified setup.

Value Returned

t	If the status of overwrite history is set successfully.
nil	Otherwise.

Example

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))
=>2417

axlSetOverwriteHistory(x_activeSetup t)
=>t

axlSetOverwriteHistoryName(x_activeSetup "Interactive.1")
=>"2535"
```

axlSetOverwriteHistoryName

```
axlSetOverwriteHistoryName(
     x_setup
     t_overwriteHistoryName
     )
     =>t / nil
```

Description

Sets the name of the history to be overwritten for the specified active setup.

Arguments

x_setup	Handle to the active setup.
t_overwriteHistoryName	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`	If the overwrite history name is set successfully.
`nil`	Otherwise.

Example

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))
=>2417
axlSetOverwriteHistory(x_activeSetup t)
=>t
axlSetOverwriteHistoryName(x_activeSetup "Interactive.1")
=>"2535"
```

axlOpenResDB

```
axlOpenResDB(
     t_fileName
     )
     => o_obj / nil
```

Description

Opens the database file specified by *t_fileName*. If the file does not exist, it is created.

Argument

t_fileName	Database file to be opened.

Value Returned

o_obj	Object handle to the database.
nil	Unsuccessful operation.

Example

```
resDB=axlGetHistoryResults(axlGetRunData(session runid))
obj = axlOpenResDB(resDB)
```

Reference

axlGetHistoryResults

axlPutHistoryEntry

```
axlPutHistoryEntry(
     x_hsdb
     t_historyName
     )
     => x_history / nil
```

Description

Inserts or finds a history entry in the setup database and returns a handle to that entry.

Arguments

x_hsdb	Setup database handle.
t_historyName	History entry name.

Value Returned

x_history	Handle to a history entry.
nil	Unsuccessful operation.

Example

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())
axlPutHistoryEntry( data_sdb "data_design_verification" )
1006
```

Reference

axlCreateSession, axlSetMainSetupDB

axlReEvaluateHistory

```
axlReEvaluateHistory(
     t_sessionName
     x_historyHandle
     S_mode
     )
     => t / nil
```

Description

Re-evaluates the history in the specified mode with respect to the active setup for a given session.

Arguments

t_sessionName	Name of the ADE XL or `maestro` view.
x_historyHandle	Setup database handle.
S_mode	Symbol representing the mode. The valid values are: ■ `all`: re-evaluates expressions and specifications ■ `specs`: re-evaluates specifications ■ `exprs`: re-evaluates expressions ■ `partial`: re-evaluates both expressions and specifications using the partially complete simulation data Default value is `all`.

Value Returned

`t`	Successfully re-evaluated the history.
`nil`	Unsuccessful operation.

Example

```
session = axlGetWindowSession( hiGetCurrentWindow())
history = axlGetCurrentHistory(session)
axlReEvaluateHistory(session history 'all)
```

axlRemoveSimulationResults

```
axlRemoveSimulationResults(
     x_historySDB
     )
     => t / nil
```

Description

Removes the simulation results data for the given history. The function removes only the results saved by the simulator. The ADE XL results database and the history item is not removed.

Note: If you need to remove the entire history, use axlRemoveElement for the history element.

To remove the simulation results data for a history from the ADE XL GUI, right-click the history name in the Data View assistant and choose *Delete Simulation Data* from the context-sensitive menu.

Argument

x_historySDB	Handle to the history for which the simulation data is to be deleted. This cannot be a handle to the checkpoint of a history.

Value Returned

`t`	Simulation results data for the given history is successfully deleted.
`nil`	Deletion of the simulation results data is not successful.

Example

The following sample code demonstrates how to delete the simulation data for a history, `Interactive.9`:

```
session = (axlGetWindowSession)
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
x_historySDB=axlGetHistoryEntry(x_mainSDB "Interactive.9")
=> 1115
```

```
axlRemoveSimulationResults(x_historySDB)
=> t
```

axlRestoreHistory

```
axlRestoreHistory(
     t_session
     x_historyEntry
     )
     => t
```

Description

Sets the given history as active setup in the given ADE XL session.

Arguments

t_session	Name of session in which setup from the history is to be applied.
x_historyEntry	Setup database handle for a history item.

Value Returned

`t`	Successful operation.
`nil`	Successful operation.

Example

```
session = (axlGetWindowSession)
=> "session0"
main_sdb = axlGetMainSetupDB(session)
=> 1001
x_historyEntry=axlGetHistoryEntry(x_mainSDB "Interactive.1")
=> 1112
axlRestoreHistory(t_session x_historyEntry)
=> t
```

axlViewHistoryResults

```
axlViewHistoryResults(
     t_session
     x_hsdb
     )
     => t
```

Description

Display the results for the specified history item on the Results tab of the given ADE XL session.

Arguments

t_session	Name of session in which the results should be displayed.
x_hsdb	Setup database handle for a history item.

Value Returned

t	Successful operation.

Example

```
session = (axlGetWindowSession)
=> "session0"
main_sdb = axlGetMainSetupDB(session)
first_history_sdb = axlGetHistoryEntry(main_sdb caadr( axlGetHistory(
main_sdb)))
axlViewHistoryResults(session first_history_sdb)
```

axlWriteMonteCarloResultsCSV

```
axlWriteMonteCarloResultsCSV
     t_session
     t_historyName
     [ ?testName t_testName ]
     [ ?cornerName t_cornerName ]
     [ ?outputName t_outputPath ]
     )
     => t / nil
```

Description

Writes the results of the given Monte Carlo run history in CSV format. ADE XL saves the results for each corner in a separate .csv file.

Arguments

t_session	Session name or the handle to the ADE XL session.
t_historyName	Name of the Monte Carlo history.
?testName *t_testName*	Name of the test for which the results are to be exported. Default: `nil`, which means that the results for all the tests are exported.
?cornerName *t_cornerName*	Name of the corner for which the results are to be exported. Default: `nil`, which means that the results for all the corners are exported.
?outputName *t_outputPath*	Path where the exported .csv files are to be saved. Default: ADE XL view directory

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
session = axlGetWindowSession()
=> "session0"
axlWriteMonteCarloResultsCSV("session0" "MonteCarlo.1" ?testName "AC" ?cornerName
"C1" ?outputPath "/tmp/csvfiles/")
=> t
```

13

Job Policy Functions

When you run a simulation in ADE XL, it starts IC Remote Processes (`ICRPs`) where it runs simulations. Each ICRP is also called a `job` and can be configured to run one or more simulation points. ADE XL internally uses these ICRPs or jobs to efficiently distribute time-consuming tasks that can be performed in parallel. Settings for these jobs such as how many remote processes to start; where the processes should run, on local or remote machines; or the time for which a remote process should stay active and wait for a simulation to run; are set as a job policy. The functions described in this chapter are used to configure and manage job policies for ADE XL.

Job Policy SKILL Functions

Function	Description
axlAddJobPolicy	Adds or saves a job policy at the specified location.
axlAttachJobPolicy	Adds and attaches a job policy to the ADE XL setup.
axlDeleteJobPolicy	Deletes the named job policy from the setup.
axlDetachJobPolicy	Detaches a job policy from the specified test.
axlJobIntfcDebugPrintf	Formats and writes output to the log if interface debugging is enabled.
axlJobIntfcDebugToFile	Enables the interface job debugging and sets the output to a file.
axlJobIntfcDebugp	Specifies whether interface debugging is enabled.
axlJobIntfcExitMethod	Job Interface member function used to exit a job. ADE XL usually attempts to call exit(0) on remote job cleanup the job's resources properly. However, if this fails or if ADE XL is forced to kill all jobs, the exit method will be called on every remote job. This method may be called after the job has already exited. This job can also be called multiple times.

Job Policy SKILL Functions, *continued*

Function	Description
axlJobIntfcHealthMethod	Job Interface member function used to return the current health of the job. ADE XL calls this function regularly (currently, every 5 seconds) on each job in order to recognize health changes. The available health types are: `unknown`, `alive`, or `dead`. If the job Interface detects that the process is still pending, return `unknown`. If the Job Interface detects that the job has launched, return `alive`. Otherwise, ADE XL will eventually receive a start message from the remote job and automatically change the current health of the job to `alive`. If the Job Interface detects that the job has exited, return `dead`. Otherwise, ADE XL will eventually recognize that remote communication has failed and automatically change the current health of the job to `dead`.Once marked `dead`, the health will not be queried, `dead` is a terminal state. If no change is detected, the current state should be returned.
axlJobIntfcSetDebug	Enables or disables printing of the job interface diagnostics to the CIW.
axlJobIntfcStartMethod	Job Interface member function to start a job. For each new job ID, a new instance of the selected interface class will be created. After some basic properties are set on the instance, it will be passed to the start method of the class.
axlJPGUICustDiffer	Job Policy GUI Customization member function that determines whether `l_propList1` and `l_propList2` differ. The Job Policy GUI uses this method to determine if the GUI settings differ from those already attached to ADE XL.
axlJPGUICustHIFields	Job Policy GUI Customization member function to create the HI field displayed for a particular JP GUI customization. The Job Policy GUI calls this method once during initialization. If no customizations are desired, the function does not need to be specialized.
axlJPGUICustOffset	Job policy GUI customization member function to return the `y` size of the HI field customizations provided with the `axlJPGUICustHIFields` method. The value is obtained by adding 10 to the y position of the last HI element. The Job Policy GUI uses this value as the `y` offset for the form elements underneath the customization area.

Job Policy SKILL Functions, *continued*

Function	Description
axlJPGUICustReadFromForm	Job Policy GUI Customization member function to read any HI customization into a property list that will be saved as a job policy.
axlJPGUICustSelected	Job Policy GUI Customization member function to enable/disable any HI customizations. The Job Policy GUI calls this function every time the job policy type is changed.
axlRegisterJobIntfc	Registers a job interface class into ADE XL. Job interfaces should be registered before use, preferably during virtuoso initialization.
axlRegisteredJobIntfcNames	Retrieve the registered job interfaces.
axlRegisterJPGUICust	Job Policy GUI Customization member function to register a customization into the Job Policy GUI. Any registered customizations will appear the next time the job policy GUI is displayed.
axlGetAttachedJobPolicy	Returns the current job policy attached to the setup or to the given test.
axlGetJobPolicy	Returns a disembodied property list containing property-value pairs for the job policy.
axlGetJobPolicyTypes	Returns a list containing names of all available job policies.
axlIsICRPProcess	Returns `t` if the code is currently running in a remote child process for ADE XL. You can use this function in your `.cdsinit` file or in custom SKILL code.
axlSaveJobPolicy	Saves the policy given by policyName.
axlSetJobPolicyProperty	Sets a property name-value pair for the specified job policy. You can use this function to update the properties of an existing policy. To apply the updated properties to all the ADE XL sessions, set the updated policy as the default policy for ADE XL by using the defaultJobPolicy environment variable.
axlStopAllJobs	Stops all the ICRPs jobs present in the system.
axlStopJob	Stops a job.

Property List for a Job Policy

A job policy is defined by a set of properties, which you need to provide as a list of property name-value pairs in the axlAddJobPolicy, axlAttachJobPolicy, and axlSetJobPolicyProperty SKILL functions. A similar list is returned by the axlGetJobPolicy and axlGetAttachedJobPolicy functions.

The following table describes all the properties that can be defined for a property.

Note: This table does not include the properties that can be set for the interface distribution method, which can be user-specific.

Job Policy Property	Description
`distributionmethod "t_method"`	Job distribution method. The job distribution method you specify determines which properties you can specify. Valid Values: `LBS` - The ICRP jobs are run using Cadence LBS layer, which can be configured to run any of the cdsqmgr, LSF or SGE cluster. `Command` - The ICRP jobs are started using the jobsubmitcommand argument. `Remote-Host` - The ICRP jobs are run on the remote host that you specify by using the jobhostname argument. `Local` - The ICRP jobs are run on the local host. This is the default value. `Interface` - The ICRP jobs are run by using an interface defined in an interface class specified by using the `interfacename` argument.
`configuretimeout "x_configureTimeout"`	Integer number of seconds to wait for the `icrp` process to report back to ADE XL that it has configured the job. The wait time starts as soon as ADE XL sends the job configure command. See "Specifying Job Timeouts" in the *Virtuoso Analog Design Environment XL User Guide* for more information.

`starttimeout "x_startTimeout"`

Integer number of seconds of time to wait for the `icrp` process to report back to ADE XL that it has started the job. The wait time starts as soon as ADE XL submits the job. See "Specifying Job Timeouts" in the *Virtuoso Analog Design Environment XL User Guide* for more information.

`runtimeout "x_runTimeout"`

Integer number of seconds to wait for the `icrp` process to report back to ADE XL that it has run the simulation job. The wait time starts as soon as ADE XL sends the run command for the job.

Specify `"-1"` for infinite.

See "Specifying Job Timeouts" in the *Virtuoso Analog Design Environment XL User Guide* for more information.

`configuredtimeout "t_configuredTimeout"`

Integer number of seconds of time for which an `icrp` process can wait for a simulation request after being configured. If the job does not receive a simulation request in this time, it is timed out.

Specify `"-1"` for infinite.

`unconfiguredtimeout "t_unconfiguredTimeout"`

Integer number of seconds of time for which an `icrp` job can stay unconfigured, that is, in the started and waiting to be configured state, before it is timed out. If the job does not receive a configure request in this time, it is timed out.

Specify `"-1"` for infinite.

`lingertimeout "t_lingerTimeout"`

Integer number of seconds of time after which an `icrp` job is to be killed after the simulations finish.

Specify `"-1"` for infinite.

`jobjobname "t_jobName"`

Name of the job to identify the simulation.

`maxjobs "x_maxJobs"`

Maximum number of jobs that can be present in the system at a time.

`preemptivestart "t_preemptiveStart"`

If set to `"1"`, when ADE XL is launched, it immediately submits the specified minimum number of `icrp` jobs, which is equal to *maxjobs* or the number of tests, whichever is less.

If set to `"0"`, ADE XL does not start any icrp job when it is launched.

Default value: `1`

`startmaxjobsimmed "t_startMaxJobsImmed"`

Case 1: When simulation run is started.

If set to `"1"`, ADE immediately submits the specified maximum number of `icrp` jobs, or one job for every corner sweep point in a test, whichever is less.

If set to "`0`", ADE waits until the last job is started and communicates back to the GUI before starting the next one. The jobs start one-by-one.

Case 2: When ADE is launched.

If set to `"1"`, and

- `preemptivestart=0`, jobs are not started preemptively on ADE launch or when adding new ADE Tests.
- `preemptivestart=1`, ADE starts multiple jobs immediately on ADE launch or one new job on addition of a new ADE Tests.

If set to `"0"`, and

- `preemptivestart=0`, jobs are not started preemptively on ADE launch or when adding new ADE Tests.
- `preemptivestart=1`, ADE starts only 1 job immediately on ADE launch or on the addition of a new ADE Tests.

Default value: `1`

Note: Setting `startmaxjobssimmed` to "`0`" is not recommended because it will slow down the run time.

`reconfigureimmediately "t_reconfigureImmed"`

When set to `"1"`, in case of multiple runs, immediately reassigns an ICRP job for a new run. When `"0"`, waits until the currently running simulations complete before assigning a job for a new run.

`jobhostname "t_remoteHostName"`

(`LBS` and `Remote-Host` only) Name of the remote host on which to start the job.

`jobqueue "t_queueName"`

(`LBS` only) Name of the LBS/LSF queue.

`joboutfilename "t_jobOutFileName"`

(`LBS` only) File name prefix for the standard output stream.

`joberrfilename "t_jobErrFileName"`

(`LBS` only) File name prefix for the standard error stream.

`loginshell "t_loginShellName"`

Name of the execution environment to be initialized on the execution host, instead of propogating the environment of the submittor to the job. The property value indicates which shell to use to initialize the environment.

Possible values: `"sh"`, `"ksh"`, `"csh"`

`jobresourcerequirements "t_LSFresources"`

(LBS running over LSF only) Additional job resource requirements for LSF. For details, see LSF Resource Requirement String Format in the *Virtuoso Analog Distributed Processing Option User Guide*.

`jobprojectname "t_LSFProjectNames"`

(LBS running over LSF only) Name of the license project

`jobusergroup "t_LSFUserGroup"`

(LBS running over LSF only) Name of the user group who can submit jobs on the LSF resource

`blockemail "t_lbsBlockEmail"`

(`LBS` only) Blocks e-mail

`parallelnumprocs "t_lbsParallelNumProcs"`

(`LBS` only) Name of processors to run in parallel

`sgehardresource "t_SGEHardResources"`

(LBS running over SGE only) Hardware resource requirements string for SGE

`sgesoftresource "t_SGESoftResources"`

(LBS running over SGE only) Software resource requirements string for SGE

`sgepriority "t_SGEPriority"`

(LBS running over SGE only) Priority of the job being submitted

`sgeProjectName "t_SGEProjectName"`

(LBS running over SGE only) Name of the project that contains the jobs to be submitted

Note: "`sgeProjectName`" is applicable for `adexl` cellviews created using ADE XL and `maestro` cellviews created using ADE Assembler or ADE Explorer.

`sgeparallelenv "t_SGEParallelEnv"`

(LBS running over SGE only) Name of a parallel environment

`jobsubmitcommand "t_command"`

(Command mode only) Job submit command. See "Specifying a Job Submit Command" in the *Virtuoso Analog Design Environment XL User Guide* for more information.

`showoutputlogonerror "t_showOutputLogOnErr"`

When set to `"1"`, ADE XL shows an output log for each run corresponding to a test (that is, for each test-run combination).

When set to `"0"`, no log is displayed.

`showerrorwhenretrying "t_showErrorWhenRetry"`

When set to `"1"`, ADE XL displays the output log file on the occurrence of an error for a test, even if the distribution system is retrying the test.

When set to `"0"`, ADE XL does not display the output log if the distribution system is retrying the test.

`interfacename "t_interfaceName"`

Name of the class to be used for interface job. This is a SKILL class derived from the `axlJobIntfC` class. For more details, refer to axlRegisterJobIntfc.

axlAddJobPolicy

```
axlAddJobPolicy(
     t_jobPolicyName
     t_selectedPath
     l_jobPolicyProperties
     )
     => t / nil
```

Description

Adds or saves a job policy at the specified location.

Note: This function does not apply the job policy to ADE XL. It only saves a new job policy at the given location. To use this job policy, you need to set this policy in the ADE XL Job Policy form or by using the *defaultJobPolicy* environment variable. To apply a job policy while defining its properties, use the axlAttachJobPolicy function.

Arguments

t_jobPolicyName

Job policy name.

t_selectedPath

Location of the `.cadence` directory where the job policy file is to be stored.

You can store the job policy file in the `.cadence` directory in one of the following default locations specified in the `setup.loc` file at `<your_inst_dir>/share/cdssetup` in your Cadence installation, or customize the `setup.loc` file to specify more locations to save the job policy file:

- `.cadence` directory in the current directory
- The `.cadence` directory in the path specified in the `CDS_WORKAREA` environment variable.
- `$HOME/.cadence` (the `.cadence` directory in your home directory)
- The `.cadence` directory in the path specified in the `CDS_PROJECT` environment variable.
- The `.cadence` directory in the path specified in the `CDS_SITE` environment variable.

Note: Ensure that you have write permissions in the .cadence directory where you want to store the job policy file.

The job policy file is saved in the `jobpolicy` directory under the specified `.cadence` directory. The job policy file has the `.jp` extension. For more information, see "Saving a Job Policy" in the *Virtuoso Analog Design Environment XL User Guide*.

l_jobPolicyProperties

List of job policy property name-value pairs. For details on the job policy properties, refer to Property List for a Job Policy.

Value Returned

`t`	Successful addition of the job policy.
`nil`	Unsuccessful addition of the job policy.

Example

The following example code saves the `mypolicy` policy in the `.cadence` directory:

```
axlAddJobPolicy( "mypolicy"
          "./.cadence"
          '( nil distributionmethod "LBS"
          configuretimeout "1200"
          maxjobs "5"
          name "LBS_Policy"
          runtimeout "3600"
          starttimeout "300" ) )
=> t
;;
;;After saving a job policy, you can set this as the default policy for an ADE XL
session by using the defaultJobPolicy environment variable.
```

axlAttachJobPolicy

```
axlAttachJobPolicy(
     t_sessionName
     t_jobPolicyName
     t_toolName
     l_jobPolicyProperties
     [ t_testName ]
     )
     => t / nil
```

Description

Adds and attaches a job policy to the ADE XL setup.

The fourth argument is a disembodied property list of job policy properties. The function overwrites properties of the named job policy if it already exists.

See "Setting Up Job Policies" in the *Virtuoso Analog Design Environment XL User Guide* for more information.

Arguments

t_sessionName	The session name
t_jobPolicyName	Job policy name.
t_toolName	Tool name. Valid Values: "ICRP"
l_jobPolicyProperties	List of job policy property name-value pairs. For details on the job policy properties, refer to Property List for a Job Policy.
t_testName	Name of the test to which the job policy is attached.

Value Returned

t	Successful addition and attachment of the job policy.
nil	Unsuccessful addition and attachment of the job policy.

Example

The following example code shows how to define a job policy, mypolicy, and attach it to a test Test1:

```
session0 = axlGetWindowSession(hiGetCurrentWindow())
axlAttachJobPolicy( "session0" "mypolicy" "ICRP" '( nil distributionmethod "LBS"
configuretimeout "300" maxjobs "5" name "default" runtimeout "3600" starttimeout
"300" ) "Test1")
t
```

axlDeleteJobPolicy

```
axlDeleteJobPolicy(
     t_jobPolicyName
     )
     => t / nil
```

Description

Deletes the named job policy from the setup.

Argument

t_jobPolicyName	Job policy name.

Value Returned

`t`	Successful deletion of the named job policy.
`nil`	Unsuccessful deletion of the named job policy.

Example

The following example code shows how to delete a job policy, `mypolicy`:

```
axlDeleteJobPolicy( "mypolicy" )
t
```

axlDetachJobPolicy

```
axlDetachJobPolicy(
     t_sessionName
     t_jobType
     t_testName
     => t / nil
```

Description

Detaches a job policy from the specified test.

After the job policy is detached, the test uses the default job policy specified for the given session.

Arguments

t_sessionName	Specifies the name of the ADE XL session from which the job policy needs to be detached.
t_jobType	Specifies the type of the job policy to be detached. Valid Values: "`ICRP`"
t_testName	Specifies the name of the test from which job policy is to be detached.

Value Returned

`t`	Returns `t` if parameter value is successfully updated
`nil`	Returns `nil` otherwise

Example

The following example detachs the `ICRP` job policy from test `AC` in session `session0`.

```
axlDetachJobPolicy("session0" "ICRP" "AC")
```

axlJobIntfcDebugPrintf

```
axlJobIntfcDebugPrintf(
     t_formatString
     [g_arg1 ... g_argn]
     )
     => t / nil
```

Description

Formats and writes output to the log if interface debugging is enabled.

Arguments

t_formatString	String name
g_arg1	Argument passed

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlJobIntfcDebugPrintf("hello world")
=> "hello world"
=> t
```

axlJobIntfcDebugToFile

```
axlJobIntfcDebugToFile(
     t_file_name
     )
     => t / nil
```

Description

Enables the interface job debugging and sets the output to a file.

Argument

t_file_name	file to be opened

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Example

```
axlJobIntfcDebugToFile("~/intfc_debug")
```

axlJobIntfcDebugp

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

Description

Specifies whether interface debugging is enabled.

Arguments

None

Value Returned

`t`	Successful operation.
`nil`	if debugging is disabled.

Example

```
axlJobIntfcDebugp => nil (debugging disabled)
```

axlJobIntfcExitMethod

```
axlJobIntfcExitMethod(
     g_inst
     )
     => nil
```

Description

Job Interface member function used to exit a job. ADE XL usually attempts to call exit(0) on remote job cleanup the job's resources properly. However, if this fails or if ADE XL is forced to kill all jobs, the exit method will be called on every remote job. This method may be called after the job has already exited. This job can also be called multiple times.

Argument

`g_inst`	Subclass of axlJobIntfc

Value Returned

`nil`

Example

```
;; example setup
(defclass IPCJob ( axlJobIntfc )
    (
     (ipcHandle)
     ))
;; axlJobIntfcExitMethod
(defmethod axlJobIntfcExitMethod ( (inst IPCJob) )
    (ipcKillProcess inst->ipcHandle)
    )
```

axlJobIntfcHealthMethod

```
axlJobIntfcHealthMethod(
     g_inst
     S_currentHealth
     )
     => S_newHealth
```

Description

Job Interface member function used to return the current health of the job. ADE XL calls this function regularly (currently, every 5 seconds) on each job in order to recognize health changes. The available health types are: `unknown`, `alive`, or `dead`. If the job Interface detects that the process is still pending, return `unknown`. If the Job Interface detects that the job has launched, return `alive`. Otherwise, ADE XL will eventually receive a start message from the remote job and automatically change the current health of the job to `alive`. If the Job Interface detects that the job has exited, return `dead`. Otherwise, ADE XL will eventually recognize that remote communication has failed and automatically change the current health of the job to `dead`.Once marked `dead`, the health will not be queried, `dead` is a terminal state. If no change is detected, the current state should be returned.

Argument

g_inst	Subclass of `axlJobIntfc`

Value Returned

S_currentHealth	unknown, alive or dead

Example

```
;; example setup
(defclass IPCJob ( axlJobIntfc )
    (
     (ipcHandle)
     ))
;; health method
(defmethod axlJobIntfcHealthMethod ( (inst IPCJob) current_health )
    (cond
     ((and (current_health == "unknown")
                 (null (zerop (ipcGetExitStatus
```

```
inst->ipcHandle))))
        "dead")
     ((and (current_health == "alive")
                 (null (ipcIsAliveProcess
inst->ipcHandle)))
        "dead")
     (t
        current_health)))
```

axlJobIntfcSetDebug

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

Description

Enables or disables printing of the job interface diagnostics to the CIW.

Argument

g_enable	Enable or disable debugging

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
axlJobIntfcSetDebug(nil)
=> debugging disabled
```

axlJobIntfcStartMethod

```
axlJobIntfcStartMethod(
     g_inst
     )
     => t / nil
```

Description

Job Interface member function to start a job. For each new job ID, a new instance of the selected interface class will be created. After some basic properties are set on the instance, it will be passed to the start method of the class.

Argument

g_inst	Subclass for `axlJobIntfc`

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
;; example setup
(defclass IPCJob ( axlJobIntfc )
    (
     (ipcHandle)
     ))
;; axlJobIntfcStartMethod
(defmethod axlJobIntfcStartMethod ( (inst IPCJob) )
    (let ((ipc_handle (ipcBeginProcess inst->command)))
        inst->ipcHandle = ipc_handle
        ;; ipc_handle will be nil on failure; non-nil on success
        ipc_handle))
```

axlJPGUICustDiffer

```
axlJPGUICustDiffer(
     g_inst
     l_propList1
     l_propList2
     )
     => t / nil
```

Description

Job Policy GUI Customization member function that determines whether `l_propList1` and `l_propList2` differ. The Job Policy GUI uses this method to determine if the GUI settings differ from those already attached to ADE XL.

Arguments

g_inst	Instance of the class axlJPGUICust
l_propList1	First DPL. Any properties stored in the DPL would have been written by a previous call to `axlJPGUICustReadFrom` Form.
l_propList2	Second DPL. Any properties stored in the DPL would have been written by a previous call to `axlJPGUICustReadFrom` Form.

Value Returned

`t`	Returns `t`, if `l_propList1` and `l_propList2` differ in a way that causes reapplication of the job policy
`nil`	Unsuccessful operation.

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
())
(defmethod axlJPGUICustReadFromForm ((inst RemoteHost) form dataDpl)
(putprop dataDpl form->RemoteHostName->value 'remotehostname))
;; differ method.
(defmethod axlJPGUICustDiffer ((inst RemoteHost) dp1 dp2 )
(nequal dp1->remotehostname dp2->remotehostname))
```

axlJPGUICustHIFields

```
axlJPGUICustHIFields(
     g_inst
     x_offset
     )
     => l_fields
```

Description

Job Policy GUI Customization member function to create the HI field displayed for a particular JP GUI customization. The Job Policy GUI calls this method once during initialization. If no customizations are desired, the function does not need to be specialized.

Arguments

g_inst	Instance of the class axlJPGUICustHIFields.
x_offset	Initial form offset. Any HI fields created must be based on thisoffset

Value Returned

l_fields	List of HI form elements

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
    ())
;; hifields method
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)
    `((,(hiCreateStringField
            ?name 'RemoteHostName
            ?prompt "Host"
            ?invisible nil)
        (20 ,yoffset)
        (370 30) 147))
    )
```

axlJPGUICustOffset

```
axlJPGUICustOffset(
     g_inst
     )
     => x_offset
```

Description

Job policy GUI customization member function to return the `y` size of the HI field customizations provided with the `axlJPGUICustHIFields` method. The value is obtained by adding 10 to the y position of the last HI element. The Job Policy GUI uses this value as the `y` offset for the form elements underneath the customization area.

Argument

g_inst	Instance of the class `axlJPGUICust`

Value Returned

x_offset	Offset

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
    ())
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)
    `((,(hiCreateStringField
            ?name 'RemoteHostName
            ?prompt "Host"
            ?invisible nil)
        (20 ,yoffset)
        (370 30) 147))
    )
;; offset method.  40 is the height of the stringfield (30) + 10
(defmethod axlJPGUICustOffset ((inst RemoteHost))
    40)
```

axlJPGUICustReadFromForm

```
axlJPGUICustReadFromForm(
     g_inst
     g_form
     l_dataDpl
     )
     => nil
```

Description

Job Policy GUI Customization member function to read any HI customization into a property list that will be saved as a job policy.

Arguments

g_inst	Instance of the class axlJPGUICust
g_form	Form to be read. HI field customizations added by axlJPGUICustHIFields will be properties on the form
l_dataDpl	Property list to modify

Value Returned

None

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
    ())
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)
    `((,(hiCreateStringField
            ?name 'RemoteHostName
            ?prompt "Host"
            ?invisible nil)
        (20 ,yoffset)
        (370 30) 147))
    )
;; readfromform method.  This will store the GUI's value as the job policy
property remotehostname
(defmethod axlJPGUICustReadFromForm ((inst RemoteHost) form dataDpl)
    (putprop dataDpl form->RemoteHostName->value 'remotehostname))
```

axlJPGUICustSelected

```
axlJPGUICustSelected(
     g_inst
     g_form
     g_enabled
     )
     => nil
```

Description

Job Policy GUI Customization member function to enable/disable any HI customizations. The Job Policy GUI calls this function every time the job policy type is changed.

Arguments

g_inst	Instance of the class `axlJPGUICust`
g_form	Form to be read. HI field customizations added by `axlJPGUICustHIFields` will be properties on the form
g_enabled	Boolean set if customization is being shown and nil if hidden

Value Returned

None

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
    ())
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)
    `((,(hiCreateStringField
            ?name 'RemoteHostName
            ?prompt "Host"
            ?invisible nil)
        (20 ,yoffset)
        (370 30) 147))
    )
;; selected method
(defmethod axlJPGUICustSelected ((inst RemoteHost) form enabled)
    form->RemoteHostName->invisible = !enabled
    form->RemoteHostName->enabled = enabled)
```

axlRegisterJobIntfc

```
axlRegisterJobIntfc(
     s_displayName
     s_className
     [ ?isInitializedFun u_isInitFun ]
     [ ?displayInGUI g_shouldDisplay ]
     )
     => t
```

Description

Registers a job interface class into ADE XL. Job interfaces should be registered before use, preferably during virtuoso initialization.

Arguments

s_displayName String name to store in the job policy and display in the Job Policy GUI

s_className Name of a derived class from `axlJobIntfc`

`?isInitializedFun` *u_isInitFun*

Specifies a function to be called on initialization, before any jobs are submitted. The function shall accept two string arguments: the first is the displayName, the second, the className.It shall return `t` if initialization was succesful and `nil` otherwise. If initialization fails, the ADE XL will display an error and use the built in job policy.

The default function only returns `t`.

`?displayInGUI` *g_shouldDisplay*

Boolean that switches whether the displayName will be listed in the *Distribution Method* cyclic field on the Job Policy Setup form. It would be useful to pass nil in order to provide a policy-specific Job Policy GUI Customization (as registered with axlRegisterJPGUICust) which could be used to provide additional interface arguments.

Value Returned

`t` Successful

Example

```
axlRegisterJobIntfc("Localhost IPC" '_axlIPCJobIntfc)
=> t
```

axlJPGUICustWriteToForm

```
axlJPGUICustWriteToForm(
     g_inst
     g_form
     l_dataDpl
     )
     => nil
```

Description

Job Policy GUI Customization member function to load a property list that is saved as a job policy into HI customizations.

Arguments

g_inst	Instance of the class `axlJPGUICust`
g_form	Form to be read. HI field customizations added by axlJPGUICustHIFields will be properties on the form
l_dataDpl	Property list to read. The property names would have been saved by a previous call to `axlJPCustReadFromForm`

Value Returned

None

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
    ())
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)
    `((,(hiCreateStringField
            ?name 'RemoteHostName
            ?prompt "Host"
            ?invisible nil)
        (20 ,yoffset)
        (370 30) 147))
    )
```

```
;; readfromform method. This will set the GUI's value from the job policy
property remotehostname
(defmethod axlJPGUICustWriteToForm ((inst RemoteHost) form dataDpl)
    form->RemoteHostName->value = dataDpl->remotehostname)
```

axlRegisteredJobIntfcNames

```
axlRegisteredJobIntfcNames(
     )
     => l_names / nil
```

Description

Retrieve the registered job interfaces.

Arguments

None

Value Returned

`l_names`	List of registered names.
`nil`	Unsuccessful operation.

Example

```
axlRegisteredJobIntfcNames()
=> ("my_interface")
```

axlRegisterJPGUICust

```
axlRegisteredJPGUICust(
     S_name
     g_inst
     )
     => t
```

Description

Job Policy GUI Customization member function to register a customization into the Job Policy GUI. Any registered customizations will appear the next time the job policy GUI is displayed.

Arguments

S_name	String name for distribution method type
g_inst	instance of the class `axlJPGUICust`

Value Returned

`t`	Successful operation

Example

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
    ())
;; axlRegisterJPGUICust method
axlRegisterJPGUICust("Remote Machine" makeInstance('RemoteHost))
=> t
```

axlGetAttachedJobPolicy

```
axlGetAttachedJobPolicy(
     [ t_sessionName ]
     [ t_toolType ]
     [ t_testName ]
     )
     => l_jobPolicyProperties / nil
```

Description

Returns the current job policy attached to the setup or to the given test.

Arguments

t_sessionName	If specified, return the policy attached to the session name. By default, returns the default policy attached to the current session.
t_toolName	If provided, return the policy associated with the distribution tool type. Valid Values: "ICRP" Default Value: "ICRP"
t_testName	Name of the test to which the job policy is attached.

Value Returned

l_jobPolicyProperties

Disembodied property list (DPL) of properties of the current attached job policies. For details on the job policy properties, refer to Property List for a Job Policy.

`nil` Unsuccessful operation.

Examples

Example 1:

The following example code shows how to get the job policy attached to the current ADE XL session:

```
axlGetAttachedJobPolicy()
```

```
=> (nil blockemail "1" configuretimeout "300" distributionmethod "Local"
lingertimeout "300" maxjobs "1" name "ADE XL Default" preemptivestart "1"
runtimeout "-1" showerrorwhenretrying "1" showoutputlogerror "0"
startmaxjobsimmed "1" starttimeout "300")
```

Example 2:

The following example code shows how to get the job policy attached to test `AC` in the current ADE XL session:

```
;;
axlGetAttachedJobPolicy(axlGetWindowSession() "ICRP" "AC")
```

```
=>(nil blockemail "1" configuretimeout "200" distributionmethod "LBS"
lingertimeout "300" maxjobs "5" name "mypolicy" preemptivestart "1"
reconfigureimmediately "1" runtimeout "3600" showerrorwhenretrying
"1" showoutputlogerror "0" startmaxjobsimmed "1" starttimeout "300" )
```

axlGetJobPolicy

```
axlGetJobPolicy(
     t_jobPolicyName
     )
     => l_jobPolicyProperties / nil
```

Description

Returns a disembodied property list containing property-value pairs for the job policy.

Argument

t_jobPolicyName	Job policy name.

Value Returned

l_jobPolicyProperties

Disembodied property list (DPL) of job policy properties. For details on the job policy properties, refer to Property List for a Job Policy.

`nil` Named job policy not found.

Example

The following example returns the property name-value list for the `ADE XL Default` policy:

```
axlGetJobPolicy( "ADE XL Default" )
'( nil configuretimeout "300" distributionmethod "Local" maxjobs "1" runtimeout
"3600" starttimeout "300" )
```

```
axlGetJobPolicy( "default1" )
nil
```

axlGetJobPolicyTypes

```
axlGetJobPolicyTypes(
     )
     => l_jobPolicyNames / nil
```

Description

Returns a list containing names of all available job policies.

Arguments

None.

Value Returned

l_jobPolicyNames	List containing names of all job policies in the setup.
`nil`	No job policy is available.

Example

```
axlGetJobPolicyTypes( )
'( "mypolicy" "default" )
```

```
axlGetJobPolicyTypes( )
nil
```

axlIsICRPProcess

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

Description

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

Arguments

None.

Value Returned

`t`	Child IC remote process is running.
`nil`	Child IC remote process is not running.

Example

```
axlIsICRPProcess( )
t
```

axlSaveJobPolicy

```
axlSaveJobPolicy(
     t_policyName
     [t_targetDirectory]
     )
     => t / nil
```

Description

Saves the policy given by policyName.

Arguments

t_policyName	Job policy name.
t_targetDirectory	Directory in which the policy name will be saved. If you have specified a path, the tool saves the policy in the `jobpolicy` directory in that path. If the `jobpolicy` directory is not found, it given an error.

Value Returned

`t`	Successful operation.
`nil`	Error in case the directory does not exist or you do not have write permission.

Example

The following example code shows how to save a job policy:

```
axlSaveJobPolicy("mypolicy" "../../policies/")
```

axlSetJobPolicyProperty

```
axlSetJobPolicyProperty(
     t_jobPolicyName
     t_jobPropertyName
     t_jobPropertyValue
     )
     => t / nil
```

Description

Sets a property name-value pair for the specified job policy. You can use this function to update the properties of an existing policy. To apply the updated properties to all the ADE XL sessions, set the updated policy as the default policy for ADE XL by using the defaultJobPolicy environment variable.

Arguments

t_jobPolicyName	Name of an existing job to which you want to add a new property.
t_jobPropertyName	Job policy property.
t_jobPropertyValue	Job policy property value.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example code shows how to get the job policy attached to test Test1 and change the configuretimeout property for it:

```
;; get the job policy attached to Test1
axlGetAttachedJobPolicy("session0" "ICRP" "Test1")
(nil blockemail "1" configuretimeout "200"
distributionmethod "LBS" lingertimeout "300" maxjobs
"5" name "mypolicy" preemptivestart "1"
```

```
reconfigureimmediately "1" runtimeout "3600" showerrorwhenretrying "1"
showoutputlogerror "0" startmaxjobsimmed "1" starttimeout "300" )
;; change the configuretimeout property
axlSetJobPolicyProperty( "mypolicy" "configuretimeout" "500" )
t
```

axlStopAllJobs

```
axlStopAllJobs(
     [ t_sessionName ]
     [ g_forceFlag ]
     )
     => t / nil
```

Description

Stops all the ICRPs jobs present in the system.

Arguments

t_sessionName	If specified, terminates only the jobs associated with the session name.
g_forceFlag	Forcefully terminates the jobs.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

The following example code shows how you can use the optional arguments to stop the jobs in different ways:

```
;;Stops all the jobs
axlStopAllJobs( )
t
;; Stops only the jobs associated with the session named session0.
axlStopAllJobs("session0")
t
;;Forcefully terminates all the jobs
axlStopAllJobs( t )
t
;; Forcefully terminates the jobs associated with the session named session0.
```

```
axlStopAllJobs("session0" t)
t
```

axlStopJob

```
axlStopJob(
     t_sessionName
     x_jobId
     [ g_forceFlag ]
     )
     => t / nil
```

Description

Stops a job.

Arguments

t_sessionName	Name of the ADE XL session
x_jobId	Job ID
g_forceFlag	Forcefully terminates the job. By default, this is set to `nil` and ADE XL does not stop a job if a simulation is running on it.

Value Returned

`t`	Successful operation.
`nil`	Unsuccessful operation.

Example

```
;;Stops the job with the ID 1.
axlStopJob( 1 )
t
;;Forcefully terminates the job with the ID 1.
axlStopJob( 1 t )
t
```

Index

A

D

N