Virtuoso ADE Environment Variables Reference - Part II

Product Version ICADVM20.1
October 2020

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

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Contents

Preface

The Virtuoso Analog Design Environment XL (ADE XL) is an advanced design and simulation environment.

This manual describes how you can set up tests, simulate your designs, and analyze output in the ADE XL environment. The information presented in this manual is intended for integrated circuit designers and assumes that you are familiar with analog design and simulation.

This preface describes the following:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node (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 on licensing in ADE XL, 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*.

Technology Information

- *Virtuoso Technology Data User Guide*
- *Virtuoso Technology Data ASCII Files Reference*.
- *Virtuoso Technology Data SKILL Reference*.

Virtuoso Tools

- *Virtuoso Schematic Editor User Guide*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*
- *Spectre Circuit Simulator Reference*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide*
- *Virtuoso UltraSim Simulator User Guide*

- *Virtuoso Parasitic Estimation and Analysis User Guide*
- *Virtuoso Visualization and Analysis Tool User Guide*
- *Component Description Format User Guide*
- *Analog Expression Language 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
- Virtuoso Electrically-Aware Design with Layout Dependent Effects

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.

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.

A

Environment Variables

This appendix describes the public environment variables that control the characteristics of the ADE XL and GXL Environment. You can customize the operation and behavior of ADE XL and GXL features and forms by changing the values of particular environment variables. The default value of each variable appears in the syntax descriptions.

See the following sections for more information:

adexl.setupdb

loadSetupToActiveAlsoViewsResults

Specifies if the *Load Setup To Active* command should display the results in addition to loading the setup details from a history.

By default, ADE XL loads the results of a history while loading the setup details. When the results are large, loading them takes a lot of time. Setting this variable to `nil` loads only the setup details.

In `.cdsenv`:

```
adexl.setupdb loadSetupToActiveAlsoViewsResults boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.setupdb" "loadSetupToActiveAlsoViewsResults" 'boolean
     nil)
```

Valid Values:

`t` The *Load Setup To Active* command displays the results in addition to loading the setup details from a history.

`nil` The *Load Setup To Active* command only loads the setup details from a history and does not show results.

Default Value: `t`

saveDir

Specifies where you want the program to write the setup database file.

Note: If you do not specify a `saveDir`, or if the `saveDir` you specify is not valid, the program writes the setup database file to the ADE XL view. If your design library is set up as read-only, you can use this environment variable to specify a writable location.

In `.cdsenv`:

```
adexl.setupdb saveDir string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.setupdb" "saveDir" 'string "" )
```

Valid Values:

Any valid directory path

percentageForNearSpec

Specifies the percentage value based on which the *near* status is displayed in the *Pass/Fail* column on the Results tab of the Outputs pane when one or more measured values for an output are no more than the percentage value outside the target value of the specification.

For more information about the *near* status in the *Pass/Fail* column on the Results tab, see Viewing Specification Results in the Results Tab.

In `.cdsenv`:

```
adexl.setupdb percentageForNearSpec int 10
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.setupdb" "percentageForNearSpec" 'int 10 )
```

Valid Values:

Any integer between than `0` and `99`

Note: If set to `0`, only the *pass* or *fail* status is displayed in the *Pass/Fail* column on the Results tab.

Default Value: `10`

useNMPForMapping

Specifies whether nmp-based name mapping scheme must be used for naming files created by ADE XL and ADE GXL.

Important

> Cadence recommends setting this environment variable to `t` if you are using a design management system.

In `.cdsenv`:

```
adexl.setupdb useNMPForMapping boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.setupdb" "useNMPForMapping" 'boolean t)
```

Valid Values:

`t` Uses nmp-based name mapping scheme for naming files.

Note: Only the files in views that were created when this variable is set to `t` will have names assigned using the nmp-based name mapping scheme. Files in views that were created when this variable is not set or set to `nil`, will continue to be named using the default name mapping scheme.

`nil` Uses the default name mapping scheme for naming files.

Default Value: `nil`

adexl.test

autoCopyCellviewVars

Controls copying of new design variables and new values for existing design variables from the design associated with a test when you open an ADE XL view or add a test in ADE XL.

In `.cdsenv`:

```
adexl.test autoCopyCellviewVars boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.test" "autoCopyCellviewVars" 'boolean t)
```

Valid Values:

`t` Automatically copies design variables from the design associated with a test when you open an ADE XL view or add tests in an ADE XL view.

`nil` Disables the automatic copy of design variables when you open an ADE XL view or add tests in an ADE XL view.

You can do one of the following to manually copy new design variables and new values for existing design variables from the design associated with a test:

- On the Variables tab of the Variables and Parameters pane, right-click the test and choose *Copy from Cellview*.
- In an expanded test tree on the Data Viewpane, right-click a design variable and choose *Copy from Cellview*.

Default Value: `nil`

autoPromoteVarsToGlobal

Controls whether design variables are automatically added as global variables on the Data

View and on the Variables tab of the Variables and Parameters pane.

In `.cdsenv`:

```
adexl.test autoPromoteVarsToGlobal boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.test" "autoPromoteVarsToGlobal" 'boolean t)
```

Valid Values:

`t`	All design variables are automatically added as global variables in the *Global Variables* tree on the Data View and the Variables tab of the Variables and Parameters pane.
`nil`	Disables the automatic addition of design variables as global variables.

Default Value: `t`

checkForUnsavedViewsUponRun

Controls whether unsaved design views should be checked before running simulations.

In `.cdsenv`:

```
adexl.test checkForUnsavedViewsUponRun boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.test" "checkForUnsavedViewsUponRun" 'boolean t)
```

Valid Values:

`t`	Checks for unsaved design views before running simulations.
`nil`	Checks for unsaved design views is deferred until netlisting.

Default Value: `t`

checkForNewCellviewVarsUponRun

Controls the check for new design variables before running simulations in the schematic hierarchy.

In `.cdsenv`:

```
adexl.test checkForNewCellviewVarsUponRun cyclic "Full"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.test" "checkForNewCellviewVarsUponRun" 'cyclic
     "Full")
```

Valid Values:

`"Full"`	Checks for all new design variables before running simulations.
`"No"`	Checks for new design variables is deferred until netlisting. This helps in reducing the netlisting time when the simulation hierarchy contains large number of instances.
`"SimInfoParameters"`	Checks for only those variables that are defined in the CDF `simInfo` section and are related to netlisting or simulation.

Default Value: `"Full"`

debugDataDir

Controls where to save results for the simulations run from ADE XL Test Editor.

In `.cdsenv`:

```
adexl.test debugDataDir boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.test" "debugDataDir" 'string "./debugResults")
```

Valid Values:

Any valid directory path

Default Value: `nil`

initiallyAddNameUniqifier

Appends a sequence number to the end of test name to make it unique.

When you create a new test, ADE XL provides a name to the test by using a default format. If the *`initiallyAddNameUniqifier`* environment variable is set to `t`, the tool appends a sequence number to it to make it unique.

In `.cdsenv`:

```
adexl.test initiallyAddNameUniqifier boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.test" "initiallyAddNameUniqifier" 'boolean nil
```

Valid Values:

`t`	Appends a unique number to the test name.
`nil`	Does not append a unique number to the test name.

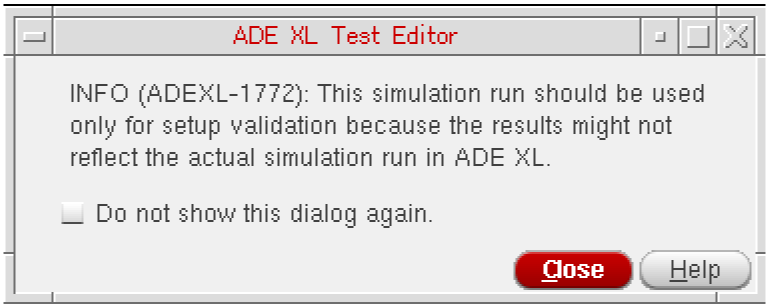
Default Value: `t`

adexl.testEditor

- adexlTestEditorSetupValidateMsg
- showAllMenus

adexlTestEditorSetupValidateMsg

Controls whether to display the following message when a debug test run is started from the ADE XL Test Editor.



ADE XL Test Editor

INFO (ADEXL-1772): This simulation run should be used only for setup validation because the results might not reflect the actual simulation run in ADE XL.

□ Do not show this dialog again.

In `.cdsenv`:

```
adexl.testEditor adexlTestEditorSetupValidateMsg boolean  t
```

Valid Values:

`t`	Displays the information message when a debug test run is started.
`nil`	Does not display the information message when a debug test run is started.

Default Value: `t`

showAllMenus

The ADE XL Test Editor window is a customized version of the Virtuoso Analog Design Environment L (ADE L) session window. By default, all ADE L menus, except the custom menus, are displayed in the ADE XL Test Editor window. Set this environment variable to `nil` to display only the ADE XL specific menus. For more details, refer to Opening the Test Editor Window.

In `.cdsenv`:

```
adexl.testEditor showAllMenus boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.testEditor" "showAllMenus" 'boolean t )
```

Valid Values:

`t`	Displays all ADE L menus, except the custom menus, in the ADE XL Test Editor window.
`nil`	Displays only the ADE XL-specific menus in the ADE XL Test Editor window.

Default Value: `t`

adexl.simulation

- showErrorForNonExistingVariables on page 50
- showWarningForReferenceNetlist on page 51
- singleNetlistForAllPoints on page 53
- sortVariableValues on page 54
- warnWhenSimsExceed on page 55

autoDetectNetlistProcs

Controls whether cellviews that use netlist procedures are automatically detected and renetlisted every time the design is netlisted.

In `.cdsenv`:

```
adexl.simulation autoDetectNetlistProcs boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "autoDetectNetlistProcs" 'boolean t)
```

Valid Values:

`t`	Automatically detects cellviews that use netlist procedures and renetlists these cellviews every time the design is netlisted.
`nil`	Disables the automatic-detection of cellviews that use netlist procedures. These cellviews will not be renetlisted every time the design is netlisted. **Note:** Even if the value is set to `nil`, a cellview that uses netlist procedures is renetlisted if parameters are specified in the *Parameters* tab of the Variables and Parameters pane for an instance of the cellview.

Default Value: `nil`

See also:

- overrideNetlistProcDetection on page 43

checkInstanceBindings

If config sweeps are used in the simulation setup, ADE Assembler checks for instance bindings in the Hierarchy Editor. If the instance or occurrence bindings in the config view conflict with the config sweeps, the tool displays a message prompt with a list of those instances and seeking confirmation whether the tool should clear the bindings in the config view. when this variable is set to nil, this check is not run, but the netlist may not be correct in that case..

In `.cdsenv`:

```
adexl.simulation checkInstanceBindings boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "checkInstanceBindings" 'boolean t)
```

Valid Values:

`t`	Before running a simulation, ADE Assembler runs a check to identify instance or occurrence bindings in the config view that conflict with the config sweeps.
`nil`	The check is not run

Default Value: `nil`

createCompositeSimLogFileWhenSimCountFewerThan

By default, ADE XL creates a composite output log if there are upto `100` points for which simulations are to be run. However, if the number of data points is notably large, a lot of disk space and time is taken to create and save the composite log. In such cases, you can use this variable to specify the maximum number of simulations up to which ADE XL should save a composite output log file for the outputs.

Note: If the composite output log file is not saved, the *Output Log* command in the context-sensitive menu for an output is not enabled.

In `.cdsenv`:

```
adexl.simulation createCompositeSimLogFileWhenSimCountFewerThan int
     50
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation"
     "createCompositeSimLogFileWhenSimCountFewerThan" 'int 45)
```

Valid Values:

A positive integer value between `0` and `100000`

Default Value: `101`

createRunLogForSweepsCorners

Specifies if the run log created for the *Single Run, Sweeps and Corners* run mode needs to include the details about the best design point. Adding this information in the run log takes time. Therefore, by default, ADE XL writes the best design point for this run mode only in the following two scenarios:

- When the *Single Run, Sweeps and Corners* simulation is run as part of the *Manual Tuning* run mode that aims at finding the best design point.
- When the number of points in the *Single Run, Sweeps and Corners* run mode is less than the limit specified by the `createRunLogWhenSimsFewerThan` environment variable.

However, if required, you can choose to include this information in other scenarios as well. For this, set this variable to one of the valid values listed in the table given below.

In `.cdsenv`:

```
adexl.simulation createRunLogForSweepsCorners cyclic
     "WhenMultipleDesignPoints"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "createRunLogForSweepsCorners"
     'cyclic "WhenMultipleDesignPoints")
```

Valid Values:

`Always`	Always appends the details of the best design point in the run log for *Single Run, Sweeps and Corners* run mode.
`ManualTuningOrSimLimited`	Appends the details of the best design point only when the *Single Run, Sweeps and Corners* run mode is run as part of the Manual Tuning run mode or when the number of points is less than the count specified by `createRunLogWhenSimsFewerThan`.
`WhenMultipleDesignPoints`	Appends the details of the best design point only when the simulation includes multiple design points. This information is not added to the run log for a simulation with a single design point.
`SimLimited`	Appends the details of the best design point if the number of points is less than the count specified by `createRunLogWhenSimsFewerThan`.
`OnlyInManualTuning`	Appends the details of the best design point only when the *Single Run, Sweeps and Corners* run mode is run as part of the Manual Tuning run.
`Never`	Never adds the details of the best design point to the run log for the *Single Run, Sweeps and Corners* run mode.

Default Value: `ManualTuningOrSimLimited`

See also:

- Viewing the Run Log for a Particular Checkpoint

createRunLogWhenSimsFewerThan

Specifies the maximum number of simulation points up to which the details of the best design point are appended to the run log for the *Single Run, Sweeps and Corners* run. ADE XL checks for this limit when the `createRunLogForSweepsCorners` environment variable is set to `ManualTuningOrSimLimited` or `SimLimited`.

In `.cdsenv`:

```
adexl.simulation createRunLogWhenSimsFewerThan int 50
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "createRunLogWhenSimsFewerThan" 'int
     50)
```

Valid Values:

A positive integer value between `0` and `1000000`

Default Value: `101`

See also:

- Viewing the Run Log for a Particular Checkpoint

diskLowWarningInterval

Sets the time interval, in milliseconds, when ADE XL must display a warning regarding low disk space after it has finished running the first run point.

In `.cdsenv`:

```
adexl.simulation diskLowWarningInterval int 50
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "diskLowWarningInterval" 'int 45)
```

Valid Values:

A positive integer value between `0` and `100000`

Default Value:	`100`

haltCurrentRunAfterPreRunTrigger

Halts the current simulation run after the preRun event occurs. When the preRun event is triggered, you can perform some checks before starting a simulation and set this environment variable to stop the simulation, if required.

For example, if you need to ensure that simulations are not run locally. Instead, they should run on remote computers only, you can use this environment variable, as shown in the code below, to halt the simulation if the distribution method is set to `Local`.

```
; define a callback function in .cdsinit
(define (RunStopper sessionName sdbHandle modeName testName)
(when ((axlGetAttachedJobPolicy)->distributionmethod == "Local")
(printf "Local distribution method used; terminating simulation\n")
(envSetVal "adexl.simulation" "haltCurrentRunAfterPreRunTrigger" 'boolean t))
)

; Connect the callback with the event
(define (connect_handlers session_name)
(axlSessionConnect session_name "preRun" 'RunStopper))

; Register the connected callback to connect the triggers on ADE XL session start
(axlSessionRegisterCreationCallback 'connect_handlers)
```

In `.cdsenv`:

```
adexl.simulation haltCurrentRunAfterPreRunTrigger boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "haltCurrentRunAfterPreRunTrigger"
     'boolean t)
```

Valid Values:

`t`	Halts the current simulation run after the preRun trigger
`nil`	Continues with the current simulation run

Default Value: `nil`

For more information and examples on triggers, refer to Working with ADE (G)XL Signals or Triggers.

ignoreAnalysisCheck

Specifies that a check for existence of analyses is not required before running a simulation. By default, before running a simulation, the tool runs a check to ensure that at least one analysis is defined. However, if the requirement is to run a simulation without any analysis, for example, when running a digital simulation, you can set this variable to `t` to ignore this check.

Additionally, if a simulation is run when the dynamic parameters from the Transient Analysis setup form are similar to the Global, Design, or Corner variable list, a dialog box pops up. To override this setting, set this environment variable to `t`.

In `.cdsenv`:

```
adexl.simulation ignoreAnalysisCheck boolean t
```

In `.cdsinit`:

```
envSetVal("adexl.simulation" "ignoreAnalysisCheck" 'boolean t)
```

Valid Values:

`t`: Ignores the analysis check.

`nil`: Runs the analysis check to ensure that at least one analysis is defined in the setup.

Default Value: `t`

ignoreDesignChangesDuringRun

Specifies whether ADE XL needs to ignore any design changes in the simulation run that is already running. For more details, refer to Ignoring Design Changes During Run.

In `.cdsenv`:

```
adexl.simulation ignoreDesignChangesDuringRun boolean t
```

In `.cdsinit`:

```
envSetVal("adexl.simulation" "ignoreDesignChangesDuringRun" 'boolean
     t)
```

Valid Values:

`t`: Ignores the design changes in the simulation run that is in progress.

`nil`: ADE XL may consider the design changes for the netlist creation and simulation of the pending design points in the current run.

Default Value: `nil`

ignoredLibsForDUT

Specifies the list of libraries that should not be displayed in the *Library* drop-down list in the Design Under Test form. Disabling the display of unnecessary libraries makes it easier to select the correct design under test library for Monte Carlo analysis.

Note: By default, the libraries `analogLib`, `cdsDefTechLib`, and `basic` are not displayed in the Library drop-down list.

In `.cdsenv`:

```
adexl.simulation ignoredLibsForDUT string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "ignoredLibsForDUT" 'string "" )
```

Valid Values:

A list of library names separated by spaces.

For example, specify the following in the `.cdsenv` file to ignore the libraries named `lib5` and `lib8`:

```
adexl.simulation ignoredLibsForDUT string "lib5 lib8"
```

includeStatementForNetlistInSimInputFile

Specifies how to include netlist file in the `input.scs` file.

Note: This variable is ignored when the ignoreDesignChangesDuringRun environment variable is set to `t`.

In `.cdsenv`:

```
adexl.simulation includeStatementForNetlistInSimInputFile boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation"
     "includeStatementForNetlistInSimInputFile" 'boolean nil )
```

Valid Values:

`t`: Includes the netlist file by using the following statement in the `input.scs` file:

```
include "netlist"
```

This helps in saving space consumed by the netlist directory because the netlist is directly included from the netlist file instead of copying the long netlist in the `input.scs` file.

`nil`: Appends the complete netlist to `input.scs` file

Default Values: `nil`

matlabResultTimeout

The time in seconds, for which ADE XL will wait after issuing a Matlab command until a prompt is returned.

In `.cdsenv`:

```
adexl.simulation matlabResultTimeout int "10"
```

In `.cdsinit` or the CIW:

```
envSetVal( " adexl.simulation” “matlabResultTimeout” ‘int "10" )
```

Valid Values:

An integer value between 1 and 10000.

Default Values: `10`

matlabStartTimeout

The time, in seconds, for which ADE XL waits for the Matlab script to start evaluation.

In `.cdsenv`:

```
adexl.simulation matlabStartTimeout int "60"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "matlabStartTimeout" 'int "60" )
```

Valid Values:

An integer value between 1 and 10000.

Default Value:	`60`

moveConfigsToNetlistDir

Specifies if the config views generated when using CONFIG global variables are to be saved in the netlist directory.

Important

> To ensure that the setting is available on the ICRP, set this variable either in `.cdsinit` or in `.cdsenv`. This will not be applied if set from CIW.

In `.cdsinit`:

```
envSetVal("adexl.simulation" "moveConfigsToNetlistDir" 'boolean nil )
```

Valid Values:

`t`	The generated configs are moved to the corresponding netlist directory.
`nil`	The config information generated for every data point is saved in the *`<library>`*`/`*`<cell>`*`/` directory structure. This option is helpful for debugging purposes as the config views saved in the design hierarchy can be viewed in Virtuoso Hierarchy Editor. However, the view list will become notably large as it will include the config view for all the data points.

Default Values: `t`

overrideNetlistProcDetection

Controls how messages are displayed when the autoDetectNetlistProcs environment variable is set to `t` and the netlisting mode for incremental simulation runs is set to *Use reference netlist* option in the Reference History form.

In `.cdsenv`:

```
adexl.simulation overrideNetlistProcDetection string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "overrideNetlistProcDetection" 'string
     "")
```

Valid Values:

`""`	Displays a message box that indicates that auto-detection and execution of netlist procedures is disabled because the netlisting mode for incremental simulation runs is set to *Use reference netlist*, and prompts you to continue or cancel the incremental simulation run.
`yes`	Displays a warning in the CIW that indicates that auto-detection and execution of netlist procedures is disabled because the netlisting mode for incremental simulation runs is set to *Use reference netlist*, and continues with the incremental simulation run.
`no`	Displays a message box that indicates that auto-detection and execution of netlist procedures is disabled because the netlisting mode for incremental simulation runs is set to *Use reference netlist* option, and requires you to either set the netlisting mode in the Reference History form to *New,* or set the autoDetectNetlistProcs environment variable to `nil`.

Default Value: `""`

overwriteHistory

Controls whether a specified history item is overwritten for subsequent simulation runs.

For more information, see Overwriting a History Item during Subsequent Simulation Runs.

In `.cdsenv`:

```
adexl.simulation overwriteHistory boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "overwriteHistory" 'boolean t)
```

Valid Values:

`t` Enables overwriting the specified history item for subsequent simulation runs.

Note: The value of this variable is automatically set to `t` if you specify `1` as the value for the saveLastNHistoryEntries environment variable.

`nil` Disables overwriting the specified history item for subsequent simulation runs.

A new history item will be created for each simulation run.

Default Value: `nil`

See also:

- overwriteHistoryName on page 44
- retainNetlistsOverwriteHistory on page 45

overwriteHistoryName

Specifies the name of the history item to be overwritten for subsequent simulation runs.

Note: The `overwriteHistoryName` variable impacts only the newly created adexl views. The existing views will retain the values they have in the active setup.

For more information, see Overwriting a History Item during Subsequent Simulation Runs.

In `.cdsenv`:

```
adexl.simulation overwriteHistoryName string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "overwriteHistoryName" 'string "" )
```

Valid Values:	`Next History Run`	Specifies that the next history item that is created should be overwritten for subsequent simulation runs.
	Name of any existing history item. For example, `Interactive.3`	
Default Value:	`Next History Run`	

See also:

- overwriteHistory on page 43
- retainNetlistsOverwriteHistory on page 45

retainNetlistsOverwriteHistory

Controls whether the netlist information in a history item that is specified to be overwritten is retained for subsequent simulation runs.

For more information, see Overwriting a History Item during Subsequent Simulation Runs.

In `.cdsenv`:

```
adexl.simulation retainNetlistsOverwriteHistory boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "retainNetlistsOverwriteHistory"
     'boolean t)
```

Valid Values:

`t`	Retains the netlist information in the history item for subsequent simulation runs.
`nil`	Deletes the netlist information before each subsequent simulation run.

Default Value: `nil`

See also:

- overwriteHistory on page 43
- overwriteHistoryName on page 44

saveBestNDesignPoints

Specifies the default number of best design points for which to save data when the *Save best* radio button is selected in the *Data Points per Optimization Run* group box on the Save Options form that appears when you choose *Options – Save* in the ADE GXL environment. See saveBestPointsStrategy for more information.

In `.cdsenv`:

```
adexl.simulation saveBestNDesignPoints int 10
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "saveBestNDesignPoints" 'int 10 )
```

Valid Values:

Any integer greater than `10`

saveBestPointsStrategy

Specifies which radio button is selected in the *Design Points per Optimization Run* group box on the Save Options form that appears when you choose *Options – Save* in the ADE GXL environment.

In `.cdsenv`:

```
adexl.simulation saveBestPointsStrategy cyclic "Save best"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "saveBestPointsStrategy" 'cyclic "Save
     best" )
```

Valid Values:

`Save all design points`	Saves data for all design points
`Save best`	Saves data for the specified number of best design points; use saveBestNDesignPoints to specify the number of points

saveLastNHistoryEntries

Specifies the number of history entries (checkpoints) to save above and beyond any locked entries. See also Specifying Options for Saving Simulation Results.

In `.cdsenv`:

```
adexl.simulation saveLastNHistoryEntries int 10
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "saveLastNHistoryEntries" 'int 10 )
```

Valid Values:

Any integer greater than `0`

saveNetlistData

Specifies whether to preserve the netlist data generated during a simulation run. This is similar to the *Save Netlists* check box in the Save Options form (See Specifying Options for Saving Simulation Results) that appears when you choose *Options – Save* in the ADE XL environment.

In `.cdsenv`:

```
adexl.simulation saveNetlistData cyclic "Save all points"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "saveNetlistData" 'cyclic "Save all
     points" )
```

Valid Values:

`Save all points`	Preserves netlist data
`Save none`	Deletes netlist data after the simulation run is complete

saveRawData

Specifies whether to preserve the simulation data generated during a simulation run. This is similar to the Save Simulation Data check box in the Save Options form (See Specifying Options for Saving Simulation Results) that appears when you choose *Options – Save* in the ADE XL environment.

In `.cdsenv`:

```
adexl.simulation saveRawData cyclic "Save all points"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "saveRawData" 'cyclic "Save all points"
     )
```

Valid Values:

`Save all points`	Preserves simulation data
`Save none`	Deletes simulation data after the simulation run is complete

saveRawDataMode

Specifies whether to save all the simulation data or only the quick plot data.

Note: This variable will work only if saveRawData is set to `Save all points`.

In `.cdsenv`:

```
adexl.simulation saveRawDataMode cyclic "All"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "saveRawDataMode" 'cyclic "All" )
```

Valid Values:

`All`	Preserves all the simulation data
`Quick Plot Data Only`	Preserves only the quick plot data

setCurrentRunPostSimulation

Specifies if ADE XL should internally open the psf data for the last run simulation. By default, this variable is set to `nil` and ADE XL does not open the psf data after running the simulation. The tool explicitly opens it when you perform post processing operations, such as plotting of graphs. This improves the performance in case of large number of sweeps and corners.

If you have any scripts that plot data after running simulation, set this variable to `t` so that the results are readily available.

In `.cdsenv`:

```
adexl.simulation setCurrentRunPostSimulation boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "setCurrentRunPostSimulation" 'boolean
     nil)
```

Valid Values:

`t`	Opens the psf data for the last run simulation
`nil`	Does not open the psf data.

Default Value: `nil`

showErrorForNonExistingVariables

Checks whether before running a simulation, ADE XL should match the design variables in the Corners Setup form with the list of global variables in the active setup. If the setup for corners uses any design variable that is not present in the active ADE XL setup, simulation is not run and an error is displayed suggesting you to either add that design variable in the active setup or to remove it from the Corners Setup form.

In `.cdsenv`:

```
adexl.simulation showErrorForNonExistingVariables boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "showErrorForNonExistingVariables"
     'boolean t)
```

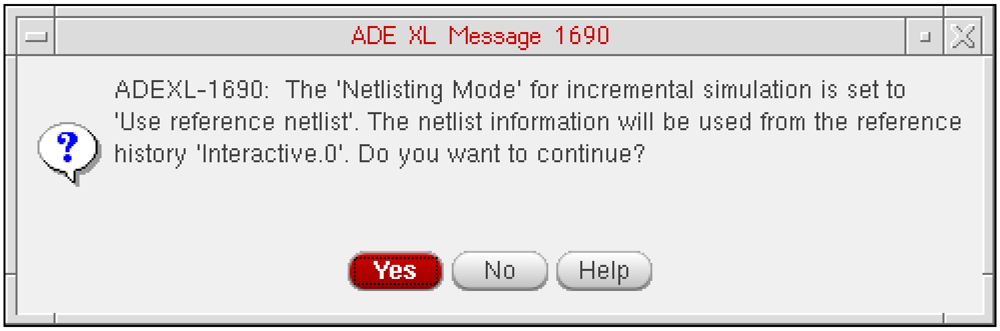
Valid Values:

`t`	Checks for the presence of non-existing variables in the setup for corners.
`nil`	Does not check for the presence of non-existing variables in the setup for corners.

Default Value: `nil`

showWarningForReferenceNetlist

Controls whether the following message box is displayed when you run an incremental simulation with the netlisting mode set to *Use reference netlist* in the Reference History form.



ADEXL-1690: The 'Netlisting Mode' for incremental simulation is set to 'Use reference netlist'. The netlist information will be used from the reference history 'Interactive.0'. Do you want to continue?

For more information about incremental simulation, see Running an Incremental Simulation.

In `.cdsenv`:

```
adexl.simulation showWarningForReferenceNetlist boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "showWarningForReferenceNetlist"
     'boolean t)
```

Valid Values:

`t`	Enables the display of the message box when you run an incremental simulation.
`nil`	Disables the display of the message box when you run an incremental simulation.

Default Value: `t`

singleNetlistForAllPoints

By default, ADE XL creates and saves a separate netlist file in the results directory for every design point. For large designs, this results in consuming huge space with same netlist file being saved in multiple directories.

This variable specifies that a common netlist is to be used for all the design points. When this variable is set, a single netlist file is created and a link to that is created in all the point directories. This helps in minimizing the size of the simulation directory.

Note: This variable is ignored in the following cases:

- When the ignoreDesignChangesDuringRun environment variable is set to t
- When device parameterization is enabled

In `.cdsenv`:

```
adexl.simulation singleNetlistForAllPoints boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "singleNetlistForAllPoints" 'boolean t
     )
```

Valid Values:

`t`: Specifies that only a single netlist will be created for all points

`nil`: Creates a separate netlist for each point

Default Value: `nil`

sortVariableValues

By default, while running simulations with corners, ADE XL saves the values of variables and model sections in the order in which they are specified by you. It maintains the same order while saving and displaying the results.

To sort the variable values and model sections in an alphabetical order, set this variable to `t`.

In `.cdsenv`:

```
adexl.simulation sortVariableValues boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "sortVariableValues" 'boolean nil )
```

Valid Values:

`t`: Sorts the variables values and model sections and displays the results in an alphabetical order.

`nil`: Uses the variables values and model sections in the user-specified order.

Default Value: `nil`

warnWhenSimsExceed

Specifies the maximum number of simulations after which the following warning message is displayed:

```
ADEXL-1703: You are about to run more than <max_number> simulations in ADE XL. Do
you want to continue ?
```

In `.cdsenv`:

```
adexl.simulation warnWhenSimsExceed int 100
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.simulation" "warnWhenSimsExceed" 'int 100 )
```

Valid Values:

Any number from `0` to `50000`. If `0`, the warning message will not be displayed irrespective of the number of simulations.

Default Value: `100`

adexl.distribute

continueICRPRunOnAbruptGUIExit

Enables continuation and completion of in-process simulations after the ADE XL GUI exits abruptly.

In `.cdsenv`:

```
adexl.distribute continueICRPRunOnAbruptGUIExit boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.distribute" "continueICRPRunOnAbruptGUIExit"
     'boolean t)
```

Valid Values:

`t`	If the ADE XL GUI exits abruptly, keeps the in-process simulations active. After completion of these simulations, saves their results in the results database.
`nil`	Stops the in-process simulations immediately after the ADE XL GUI exits.

Default Value:

```
nil
```

createUniqueLogsDirForICRPLogs

Specifies if, for each Virtuoso process started from a directory, a unique log subdirectory needs to be created under the `logs_<username>_logs<num>` directory in the Virtuoso working directory. This subdirectory will be used by all the ICRPs started by that Virtuoso process to write their job log files.

By default, this variable is set to `t` and a unique subdirectory is created for each Virtuoso process.

In `.cdsenv`:

```
adexl.distribute  createUniqueLogsDirForICRPLogs boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "createUniqueLogsDirForICRPLogs"
     'boolean t)
```

Valid Values:

`t` Creates unique subdirectories for each Virtuoso process started from a directory.

`nil` All Virtuoso processes share a common subdirectory under the `logs_<processID>` directory.

Default Value:

`t`

defaultRunInParallel

Specifies the default option for the Run in field in the Run Options form that appears when you choose *Options – Run Options* in the ADE XL environment.

In `.cdsenv`:

```
adexl.distribute defaultRunInParallel boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "defaultRunInParallel" 'boolean t )
```

Valid Values:

`t`	Sets *Parallel* as the default option for the Run in field in the Run Options form.
`nil`	Sets *Series* as the default option for the Run in field in the Run Options form.

defaultPerRunNumJobs

Specifies a default value for the Specify field in the Run Options form that appears when you choose *Options – Run Options* in the ADE XL environment.

In `.cdsenv`:

```
adexl.distribute defaultPerRunNumJobs int 5
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "defaultPerRunNumJobs" 'int 5 )
```

Valid Values:

Any positive integer

enableICRPReconnect

Enables reconnection of an ICRP job with the ADE XL GUI. This variable is useful in scenarios when due to some issues with a remote server running an ICRP job, the DRMS (LSF or SGE) migrates the ICRP to another resource, and reconnects it with the ADE XL GUI using the same job ID.

By default, when the DRMS migrates an ICRP job, the connection between the ADE XL GUI and the ICRP is broken. ADE XL assumes that the ICRP job has exited due to an error. If that job was running a point, ADE XL resubmits that point to another ICRP. In this case, when the original ICRP that was migrated to another resource tries to re-establish the connection, ADE XL ignores it and does not recognize it as a valid ICRP.

When you set the `enableICRPReconnect` environment variable to `t`, ADE XL allows the original ICRP to re-establish the connection and to run the same point again. In this case, the ICRP is considered as a new process and goes through all the states, such as starting, configuring, and evaluating. The in-progress simulations that were earlier running for the points submitted to the original ICRP might run on this new ICRP or some other ICRP that might be available at that time.

In `.cdsenv`:

```
adexl.distribute enableICRPReconnect boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "enableICRPReconnect" 'boolean t )
```

Valid Values:

`t`	Reconnects an ICRP job with the ADE XL GUI.
`nil`	Does not reconnect ICRP jobs with the ADE XL GUI. Instead, new ICRP jobs are started to submit the points.

estimateMemoryUnitForFarm

Specifies the unit for displaying the estimated memory usage in a simulation, which will also be an estimation of the memory required for farm machines.

In `.cdsenv`:

```
adexl.distribute estimateMemoryUnitForFarm cyclic "M"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "estimateMemoryUnitForFarm" 'cyclic "B"
    )
```

Valid Values:

`"B"`	Units in Byte
`"K"`	Units in Kilobyte
`"M"`	Units in Megabyte
`"G"`	Units in Gigabyte
`"T"`	Units in Terabyte

Default Value:

```
"M"
```

generateJobFileOnlyOnError

Specifies if the job log is to be saved only for jobs with an error or for all the jobs. By default, the job log is saved only when a point fails due to an error.

In `.cdsenv`:

```
adexl.distribute generateJobFileOnlyOnError boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "generateJobFileOnlyOnError" 'boolean
     t)
```

Valid Values:

`t`	Saves the job log only for the jobs that fail.
`nil`	Saves the job log for all the jobs.

Default Value: `t`

inferCommandICRPStatusFromProxy

Specifies whether ADE XL should consider the command jobs to be interactive or not so as to infer the ICRP status from the local shell or proxy process.

In `.cdsenv`:

```
adexl.distribute inferCommandICRPStatusFromProxy cyclic "Always"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "inferCommandICRPStatusFromProxy"
     'cyclic "Never" )
```

Valid Values:

`"Always"`	Specifies that the job is always interactive. **Note:** Use this value when you are sure that every job would be interactive because if it is not, the job distribution might not work correctly.
`"Never"`	Specifies that the job is never interactive.
`"GuessFromCommand"`	ADE XL will treat the jobs to be interactive only when interactive flags or commands are given. To know about the interactive flags that you can use, refer to Specifying a Command for DRMS. **Note:** ADE XL guesses the known interactive flags or commands only for LSF, SGE, and Network Computer. If you have any other DRMS, ADE XL will not be able to understand whether the jobs are interactive or not. In such case, set this variable to `Always`.

Default Value:

```
"GuessFromCommand"
```

isLSFMemSwapHostLimit

Sets the memory limit specified in the LSF queue for simulation runs by enabling or disabling the *Memory Host Limit* field on the Job Policy Setup form.

For more information, see Setting Up Run Options.

In `.cdsenv`:

```
adexl.distribute isLSFMemSwapHostLimit boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "isLSFMemSwapHostLimit" 'boolean nil )
```

Valid Values:

`t`	Enables the memory limit configured with the LSF queue for the jobs submitted for simulation. **Note:** The value of this variable is automatically set to `t` if you select the *Memory Host Limit* check box on the Job Policy Setup form.
`nil`	Disables the memory limit configured with the LSF queue for the jobs submitted for simulation. **Note:** The value of this variable is automatically set to `nil` if you deselect the *Memory Host Limit* check box on the Job Policy Setup form.

Default Value:

```
nil
```

jobFileHeader

Specifies what type of header information will be prepended to the Job logs which are saved in the `psf` directories for individual points.

Note: The variable `adexl.distribute generateJobFileOnlyOnError` controls whether these individual Job logs are saved for every point, or only for failed points.

In `.cdsenv`:

```
adexl.distribute jobFileHeader cyclic "None"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "jobFileHeader" 'cyclic "None" )
```

Valid Values:

`"None"`	Does not include any header information in the log.
`"CIWHeader"`	Includes the CIW header.
`"JobInfo"`	Writes the values of common environment variables from LSF and SGE.
`"CIWHeaderAndJobInfo"`	Includes both, CIW header and job information.
`"Custom"`	Calls a user-defined function, `axlCustomJobLogHeader()`. It accepts a disembodied property list as argument, which has the following properties:

- `CIWHeader`: contains the header information as a list of strings
- `JobInfo`: contains job information as a list of strings

The following example demonstrates how `axlCustomJobLogHeader()` can be used:

```
procedure( axlCustomJobLogHeader( myData )
println("Anything written to stdout, such as
this text, will appear in the job log header")
foreach( x myData->CIWHeader println(x) )
foreach( x myData->JobInfo  println(x) )
)
```

This example will print the header information similar to what `CIWHeaderAndJobInfo` will do.

Default Value:

```
"None"
```

jobFileDir

Specifies a location where the user logs are saved. By default, the logs are saved in the `logs_<user-name>` directory in the current run directory.

In `.cdsenv`:

```
adexl.distribute jobFileDir string <dir-path>
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "jobFileDir" 'string <dir-path> )
```

Valid Values:

Path to the directory where you want to save the user logs.

useAllLingeringJobs

Specifies whether idle or unconfigured jobs must be used when simulations runs are run in series.

For example, assume that you have specified that a maximum of five jobs must be used when simulation runs are run in series. You then run two runs in series, with the first run requiring five jobs to complete and the second run requiring three jobs to complete. When the first run is complete, there will be five idle jobs, but the second run requires only three jobs. If `useAllLingeringJobs` is set to `nil` (the default), the second run will use only three jobs and the remaining two idle jobs will timeout according to the specified linger timeout value. If `useAllLingeringJobs` is set to `t`, the second run will use all the five jobs.

For information about specifying the maximum number of jobs to be used when simulation runs are run in series, see Setting Up Run Options. For information about specifying the linger timeout value for jobs, see Specifying Job Timeouts.

In `.cdsenv`:

```
adexl.distribute useAllLingeringJobs boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "useAllLingeringJobs" 'boolean nil )
```

Valid Values:

`t`	Uses idle or unconfigured jobs when simulations are run in series.
`nil`	Does not use idle or unconfigured jobs when simulations are run in series.

Default Value:

```
nil
```

maxJobFailPerPolicy

Specifies the maximum number of times the application should restart an ICRP job if it fails to start.

An ICRP may fail to start due to system slowness or long queue for job distribution and as a result, an error message may appear. In such a case, set this environment variable to a large number, say `1000`.

In `.cdsenv`:

```
adexl.distribute maxJobFailPerPolicy int 1000
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "maxJobFailPerPolicy" 'int 1000 )
```

Valid Values:

`1` to `100000`

Default Value:

`3`

maxJobFailPerPolicyInBatch

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

In `.cdsenv`:

```
adexl.distribute maxJobFailPerPolicyInBatch int 20
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "maxJobFailPerPolicyInBatch" 'int 20 )
```

Valid Values:

`1` to `100000`

Default Value:

`20`

maxIPCJobsLimit

Specifies the maximum number of jobs that can be run at any time during your ADE XL session when the distribution method specified in your job policy is *Command*, *Local* or *Remote-Host* (uses SKILL-IPC). For more information on job policies, see Setting Up Job Policies.

Important

> This variable is not applicable when the distribution method is `LBS` or `Interface`. Therefore, if you want to run more than the number of jobs specified using this variable, use the axlAddJobPolicy or axlAttachJobPolicy SKILL function to set the distribution method to `LBS` or `Interface`. In the later case, you also need to provide a custom job interface derived from the `axlJobIntfC` class.

In `.cdsenv`:

```
adexl.distribute maxIPCJobsLimit int 1000
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "maxIPCJobsLimit" 'int 1000 )
```

Valid Values:

`1` to `1000000`

Default Value:

`1000`

maxJobsIsHardLimit

Controls the interaction between the maximum number of jobs specified for simulation runs, and the number of jobs specified in the *Max Jobs* field on the Job Policy Setup form.

For information about specifying the maximum number of jobs to be used for simulation runs, see Setting Up Run Options.

In `.cdsenv`:

```
adexl.distribute maxJobsIsHardLimit boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "maxJobsIsHardLimit" 'boolean t )
```

Valid Values:

`t`	Launches only the number of jobs specified in the *Max Jobs* field on the Job Policy Setup form, even if you have specified a greater number of jobs to be used for simulation runs.
`nil`	Launches the maximum number of jobs specified to be used for simulation runs, even if you have specified a lesser number of jobs in the *Max Jobs* field on the Job Policy Setup form.

Default Value:

```
t
```

numRetriesOnError

Specifies the maximum number of times ADE XL should retry to submit a simulation in case a job fails. By default, ADE XL resubmits a failed simulation once. If it fails again, ADE XL prints the output as `error`.

Note: There will be no retry in case of a partially successful simulation.

In `.cdsenv`:

```
adexl.distribute numRetriesOnError int 2
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.distribute" "numRetriesOnError" 'int 2 )
```

Valid Values:

`0` to `100`

Default Value:

`1`

runTimeoutScaleFactor

Specifies the scale factor to be used to calculate a scaled run timeout value if the `useAsRunTimeout` environment variable is set to `ScaledFromAvgSimTime` or `ScaledFromMaxSimTime`.

In `.cdsenv`:

```
"adexl.distribute" "runTimeoutScaleFactor" int 6
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.distribute" "runTimeoutScaleFactor" 'int 6)
```

Valid Values:

An integer value between `1` and `1000`

Default Value: `6`

See also:

- `useAsRunTimeout`
- `runTimeoutScalingStartsAfterSimCount`

runTimeoutScalingStartsAfterSimCount

Specifies the maximum number of simulations after which the run timeout value is to be scaled. The scale factor specified by `runTimeoutScaleFactor` is used to calculate the scaled timeout value.

In `.cdsenv`:

```
adexl.distribute runTimeoutScalingStartsAfterSimCount int 30
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.distribute" "runTimeoutScalingStartsAfterSimCount
     " 'int 30)
```

Valid Values:

An integer value between `1` and `1000`

Default Value: `20`

See also:

- `useAsRunTimeout`
- `runTimeoutScaleFactor`

useAsRunTimeout

Specifies the method to be used to calculate the run timeout value for a non-responsive ICRP job. By default, ADE XL uses the run timeout value specified in the job policy. You can use this variable to use an alternate value.

In `.cdsenv`:

```
adexl.distribute useAsRunTimeout cyclic "JobPolicyRunTimeoutValue"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.distribute" "useAsRunTimeout" 'cyclic
     "ScaledFromMaxSimTime")
```

Valid Values:

`JobPolicyRunTimeoutValue`	ADE XL uses the run timeout value from the job policy. If that value is set to `NULL`, ADE XL waits for an indefinite time for the ICRP job to confirm that the simulation is complete. In case of a large number of simulations, this can affect the completion of all the pending simulations.
`ScaledFromAvgSimTime`	If the simulation count is less than the limit specified by `runTimeoutScalingStartsAfterSimCount`, ADE XL uses the run timeout value from the job policy. If the simulation count is more than this limit, ADE XL calculates the run timeout value as: `Average sim time * runTimeoutScaleFactor`
`ScaledFromMaxSimTime`	If the simulation count is less than the limit specified by `runTimeoutScalingStartsAfterSimCount`, ADE XL uses the run timeout value from the job policy. If the simulation count is more than this limit, ADE XL calculates the run timeout value as: `Max sim time * runTimeoutScaleFactor`

Default Value: `JobPolicyRunTimeoutValue`

See also:

- `runTimeoutScaleFactor`
- `runTimeoutScalingStartsAfterSimCount`

useSameProcess

Specifies if the simulation for a single point simulation will be optimized by completing the netlist generation and expression evaluation tasks inside the ADE XL process and running the simulation as per the distribution method specified by the job policy.

In `.cdsenv`:

```
adexl.distribute useSameProcess boolean "t"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.distribute" "useSameProcess" 'boolean "t")
```

Valid Values:

`t`	The simulation with a single point is optimized by using a single process to complete all the tasks.
`nil`	The simulation with a single point is not optimized. An ICRP process is started to complete all the tasks.

Default Value:	`t`
GUI Equivalent:	*Optimize Single Point Run* on the Job Policy Setup form

adexl.monte

additionalNetlistOptions

Specifies the additional analysis options to be generated in the netlist.

In `.cdsenv`:

```
adexl.monte additionalNetlistOptions string ""
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "additionalNetlistOptions" 'string "" )
```

Valid Values:

	Any string	For example, "`nullmfactorcorrelation=yes`"
Default Value:	An empty string.	

GUI Equivalent:	The *Netlist Options* field in the *Monte Carlo* form.

applySaveOptionsToNetlist

Controls the writing of process and mismatch parameter information in the netlist.

In `.cdsenv`:

```
adexl.monte applySaveOptionsToNetlist boolean t
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "applySaveOptionsToNetlist" 'boolean t )
```

Valid Values:

`t`	Applies the `saveprocessparams` and `savemismatchparams` options in the netlist depending on the settings for the *Save Process Data* and *Save Mismatch Data* check boxes in the *Monte Carlo* form. For example, if the *Save Process Data* and *Save Mismatch Data* check boxes are not selected in the Monte Carlo form, the `saveprocessparams` and `savemismatchparams` options are set to `no` in the netlist and Spectre will not write process and mismatch parameter information to the disk. For more information about the `saveprocessparams` and `savemismatchparams` options, see the *Spectre Circuit Simulator Reference*.
`nil`	Writes process and mismatch parameter information in the netlist. When `nil`, the settings for the *Save Process Data* and *Save Mismatch Data* check boxes in the *Monte Carlo* form are not passed to the netlist.

Default Value:	`t`

createStatisticalCornerType

Specifies which method is to be used to create a statistical corner from the Monte Carlo results.

In `.cdsenv`:

```
adexl.monte createStatisticalCornerType cyclic "values"
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "createStatisticalCornerType" 'cyclic
     "values" )
```

Valid Values:

`sequence`	Create a statistical corner by using a sequence ID of a sample.
`values`	Create a statistical corner by using the statistical parameter values of a sample. **Note:** This requires saving the mismatch data while running Monte Carlo.
`prompt`	Displays the Create Statistical Corner form in which you can confirm which one of the two types mentioned above is to be used to create a statistical corner. The default choice selected in the form is to create the sequence ID-based corner. **Note:** If you have saved the mismatch data while running Monte Carlo, you can choose to create a statistical corner by using the statistical parameter values.
`promptValues`	Displays the Create Statistical Corner form in which you can confirm which one of the two types mentioned above is to be used to create a statistical corner. The default choice selected in the form is to create the statistical parameter-based corner.

`auto`

When the *Create Statistical* Corners check box is selected in the *Guided Mode* section of the Monte Carlo options form, ADE Assembler automatically creates values-based corners based on the selected advanced method, for example, K-Sigma Corners or Scaled-Sigma Sampling.

When the *Create Statistical* Corners check box is not selected in the *Guided Mode* section of the Monte Carlo options form:

- You saved statistical data by selecting the *Save Statistical Parameter Data* check box, ADE Assembler prompts you to choose a method either to create a sequence-based corner based on the Monte Carlo sequence or a value-based corner using the saved statistical parameter values.
- You did not save statistical data by selecting the *Statistical Parameter Data* check box, ADE Assembler automatically creates a sequence-based corner based on the Monte Carlo sequence.

Default Value: `auto`

See also:

- Creating Statistical Corners

enableMonteCarloOverStatisticalCorners

Enables you to run Monte Carlo analysis (mismatch variation) over parameter-based statistical corners, consisting of process variation.

In `.cdsenv`:

```
adexl.monte enableMonteCarloOverStatisticalCorners boolean nil
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "enableMonteCarloOverStatisticalCorners"
     'boolean nil)
```

Valid Values:

`t`	Monte Carlo analysis can be run over parameter-based statistical corners.
`nil`	Monte Carlo analysis cannot be run with statistical corners.

Default Value: `nil`

incrementalUpdate

Controls the update of Monte Carlo simulation results in the Results tab of the Outputs pane.

In `.cdsenv`:

```
adexl.monte incrementalUpdate boolean t
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "incrementalUpdate" 'boolean t )
```

Valid Values:

`t`	Monte Carlo simulation results are updated after each iteration of the Monte Carlo run. **Note:** Use the iterationUpdates environment variable to specify the number of iterations of the Monte Carlo run after which the simulation results are updated in the Results tab of the Outputs pane.
`nil`	Monte Carlo simulation results are displayed only after all iterations of the Monte Carlo run are over.

Default Value: `t`

iterationUpdates

Controls the number of iterations of the Monte Carlo run after which simulation results are updated in the Results tab of the Outputs pane.

In `.cdsenv`:

```
adexl.monte iterationUpdates int 10
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "iterationUpdates" 'int 10 )
```

Valid Values:

Any positive integer

Default Value: 1

minGroupSizeSplitAcrossIdleJobs

Specifies the minimum group size to be considered while splitting or distributing the Monte Carlo simulation points to the available idle jobs. By default, this variable is set to `0` and the points are not reallocated after an initial assignment to jobs. However, if there are multiple jobs, you can set this variable to a value greater than `1` to enable reallocation of points. In this case, ADE XL identifies the busy or idle jobs, and if the number of pending points on a busy job is more than the specified group size, it reallocates some of the points to the idle jobs, thereby helping in optimum utilization of the available resources.

For example, if a Monte Carlo simulation has 100 points and the *Max Jobs* field on the Job Policy Setup form is set to `2`, a set of 50 simulation points are allocated to each one of `job1` and `job2`. If the simulations running on `job1` are completed faster than those running on `job2`, `job1` becomes idle. If you have specified the minimum size for a group of simulations to be considered for the reallocation of points to 5 and the number of points pending with `job2` is greater than 5, some of the points are reallocated to `job1`. This can improve the overall run time for Monte Carlo simulations.

In `.cdsenv`:

```
adexl.monte minGroupSizeSplitAcrossIdleJobs int 0
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "minGroupSizeSplitAcrossIdleJobs" 'int 0)
```

Valid Values:

Any positive integer value greater than `1`.

Default Value: 0

numberOfPointsToView

Specifies the number of points to be displayed in the Detail results view for the Monte Carlo Sampling run mode. By default, the Detail view shows all the points of a Monte Carlo run. By setting this variable, you can choose to view only a selected number of worst points.

Note: When this environment variable is set to a zero, all the points are displayed in both Detail and Detail - Transpose results views. When it is set to a non-zero value, the Detail - Transpose results view is disabled for the Monte Carlo Sampling run mode.

In `.cdsenv`:

```
adexl.monte numberOfPointsToView int 0
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "numberOfPointsToView " 'int "0" )
```

Valid Values:

A positive integer value

Default Value: `0`

samplingMethod

Sets the default sampling method for the Monte Carlo Sampling run mode.

In `.cdsenv`:

```
adexl.monte samplingMethod string "lds"
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "samplingMethod" 'string "lds" )
```

Valid Values:

`"random"`	Random
`"lhs"`	Latin Hypercube Sampling
`"lds"`	Low-Discrepancy Sequence

Default Value: `"lds"`

savedatainseparatedir

Allows saving of raw data (`psf` files) for every Monte Carlo iteration in a separate directory so that you can perform post processing operations (like plotting, printing, annotation, re-evaluation, and so on) on individual iterations.

Note: This environment variable is honored only if the *Save Data to Allow Family Plots* check box in the *Monte Carlo* form is selected.

In `.cdsenv`:

```
adexl.monte savedatainseparatedir boolean t
```

In `.cdsinit`:

```
envSetVal( "adexl.monte" "savedatainseparatedir" 'boolean t )
```

Valid Values:

`t`	Saves raw data (`psf` files) for every Monte Carlo iteration in a separate directory. For example, if there are three iterations, the data for the iterations are saved in directories named `1`, `2` and `3` in the `libraryName/cellName/adexl/results/data/<history_item>` directory
`nil`	Disables the saving of raw data (`psf` files) for every Monte Carlo iteration in a separate directory.

Default Value: `t`

saveProcessOptionDefaultValue

Controls the default setting for the *Save Process Data* check box in the form.

In `.cdsenv`:

```
adexl.monte saveProcessOptionDefaultValue boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.monte" "saveProcessOptionDefaultValue" 'boolean t )
```

Valid Values:

`t`	The *Save Process Data* check box in the *Monte Carlo* form is selected by default (if the settings for this option is not there in the ADE XL setup database).
`nil`	The *Save Process Data* check box in the *Monte Carlo* form is deselected by default (if the settings for this option is not there in the ADE XL setup database).

Default Value: `t`

saveSimulationData

Specifies if the simulation data is to be saved so that it can be used for plotting. This environment variable sets the default value of the *Save Data To Allow Family Plots* check box in the *Monte Carlo* form.

In `.cdsenv`:

```
adexl.monte saveSimulationData boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.monte" "saveSimulationData" 'boolean nil)
```

Valid Values:

`t`	The *Save Data To Allow Family Plots* check box in the *Monte Carlo* form is selected by default.
`nil`	The *Save Data To Allow Family Plots* check box in the *Monte Carlo* form is deselected by default.

Default Value: `nil`

saveMismatchOptionDefaultValue

Controls the default setting for the *Save Mismatch Data* check box in the *Monte Carlo* form. By default, this check box is cleared and ADE XL does not save the mismatch parameters and their values in the associated Monte Carlo results files.

Note: Even if the *Save Mismatch Data* check box is cleared, the mismatch parameters do have an effect on Monte Carlo simulation.

In `.cdsenv`:

```
adexl.monte saveMismatchOptionDefaultValue boolean t
```

In `.cdsinit` (not in CIW):

```
envSetVal("adexl.monte" "saveMismatchOptionDefaultValue" 'boolean t)
```

Valid Values:

`t`	The *Save Mismatch Data* check box in the *Monte Carlo* form is selected by default (if the setting for this option is not there in the ADE XL setup database).
`nil`	Clears the *Save Mismatch Data* check box in the *Monte Carlo* form (if the setting for this option is not there in the ADE XL setup database).

Default Value: `nil`

warnWhenSimsExceed

Specifies a threshold limit for the number of simulations to be run for Monte Carlo. When the number of simulations to be run for Monte Carlo exceeds the specified limit, the tool shows a warning message to indicate that you have run the specified number of simulations and whether you want to continue further.

By default, the warning threshold is 8000. The warning will appear when the total number of simulations is greater than 8000.

Note: The tooldoes not apply this check when you use the auto stop feature to stop Monte Carlo run based on a specific criteria.

In `.cdsenv`:

```
adexl.monte warnWhenSimsExceed int 8000
```

In `.cdsinit` or in CIW:

```
envSetVal("adexl.monte" "warnWhenSimsExceed" 'int 8000)
```

Valid Values:

A positive integer value ranging between 8000 to 1000000

Default Value: `8000`

adexl.historyNamePrefix

- showNameHistoryForm
- initiallyAddHistoryNameUniquifier
- singleRunSweepsAndCorners
- monteCarloSampling
- WorstCaseCorners
- globalOptimization
- localOptimization
- improveYield
- highYieldEstimation
- sensitivityAnalysis
- feasibilityAnalysis
- manualTuning
- sizeOverCorners

showNameHistoryForm

Controls the display of the Specify History Name form before a simulation run is started. The Specify History Name form is used to specify the history name to be set for the current run.

In `.cdsenv`:

```
adexl.historyNamePrefix showNameHistoryForm boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "showNameHistoryForm" 'boolean t
     )
```

Valid Values:

`t`	Displays the Specify History Name form before the simulation run.
`nil`	Does not display the Specify History Name form before the simulation run.

Default Value: `nil`

initiallyAddHistoryNameUniquifier

Specifies if a unique incremental number is to be suffixed to the history name to keep each history name unique.

In `.cdsenv`:

```
adexl.historyNamePrefix initiallyAddHistoryNameUniquifier boolean
     nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix"
     "initiallyAddHistoryNameUniquifier" 'boolean t )
```

Valid Values:

`t`	Uses a unique incremental number as a suffix for the history names.
`nil`	Does not add any suffix to the history name. Only the history name is used for the first time. In subsequent runs that use the same history name, a unique number will be suffixed.

Default Value: `nil`

singleRunSweepsAndCorners

Specifies the default history name to be used for the Single Run, Sweeps, and Corners run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix singleRunSweepsAndCorners string
     "Interactive"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "singleRunSweepsAndCorners"
     'string "singleRun")
```

Valid Values:

A string specifying the history name.

Default Value: `"Interactive"`

monteCarloSampling

Specifies the default history name to be used for the Monte Carlo Sampling run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix monteCarloSampling string "MonteCarlo"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "monteCarloSampling" 'string
     "MonteCarlo")
```

Valid Values:

A string specifying the history name.

Default Value: `"MonteCarlo"`

WorstCaseCorners

Specifies the default history name to be used for the Worst Case Corners run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix WorstCaseCorners string "WorstCaseCorners"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "WorstCaseCorners" 'string
     "WorstCaseCorners")
```

Valid Values:

A string specifying the history name.

Default Value: `"WorstCaseCorners"`

globalOptimization

Specifies the default history name to be used for the Global Optimization run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix globalOptimization string "GlobalOpt"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "globalOptimization " 'string
     "GlobalOpt")
```

Valid Values:

A string specifying the history name.

Default Value: `"GlobalOpt"`

localOptimization

Specifies the default history name to be used for the Local Optimization run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix localOptimization string "LocalOpt"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "localOptimization" 'string
     "LocalOpt")
```

Valid Values:

A string specifying the history name.

Default Value: `"LocalOpt"`

improveYield

Specifies the default history name to be used for the Improve Yield run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix improveYield string "ImproveYield"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "improveYield" 'string
     "ImproveYield")
```

Valid Values:

A string specifying the history name.

Default Value: `"ImproveYield"`

highYieldEstimation

Specifies the default history name to be used for the High Yield Estimation run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix highYieldEstimation string
     "HighYieldEstimation"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "highYieldEstimation" 'string
     "HighYieldEstimation")
```

Valid Values:

A string specifying the history name.

Default Value: `"HighYieldEstimation"`

sensitivityAnalysis

Specifies the default history name to be used for the Sensitivity Analysis run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix sensitivityAnalysis string
     "SensitivityAnalysis"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "sensitivityAnalysis" 'string
     "SensitivityAnalysis")
```

Valid Values:

A string specifying the history name.

Default Value: `"SensitivityAnalysis"`

feasibilityAnalysis

Specifies the default history name to be used for the Feasibility Analysis run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix feasibilityAnalysis string
     "FeasibilityAnalysis"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "feasibilityAnalysis" 'string
     "FeasibilityAnalysis")
```

Valid Values:

A string specifying the history name.

Default Value: `"FeasibilityAnalysis"`

manualTuning

Specifies the default history name to be used for the Manual Tuning run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix manualTuning string "ManualTuning"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "manualTuning" 'string
     "ManualTuning")
```

Valid Values:

A string specifying the history name.

Default Value: `"ManualTuning"`

sizeOverCorners

Specifies the default history name to be used for the Size Over Corners run mode.

In `.cdsenv`:

```
adexl.historyNamePrefix sizeOverCorners string "SizeOverCorners"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.historyNamePrefix" "sizeOverCorners" 'string
     "SizeOverCorners")
```

Valid Values:

A string specifying the history name.

Default Value: `"SizeOverCorners"`

adexl.icrpStartup

binaryName

Specifies the name of the binary to run on the remote host.

In `.cdsenv`:

```
adexl.icrpStartup binaryName string "virtuoso"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "binaryName" 'string "virtuoso" )
```

Valid Values:

Any binary that is valid on the remote host (such as `virtuoso`)

defaultJobPolicy

Specifies the name of the job policy to be used if no job policy is specified in the Job Policy Setup form.

Note the following:

- If no job policy is specified in the Job Policy Setup form or using this variable, the program uses the default job policy settings.
- The job policy settings are overlaid in the following order. A setting from a previous policy is preserved in the final result if not overridden by a subsequent policy.

a. The default job policy settings.

b. The settings in the job policy specified using this variable.

c. The settings in the job policy specified in the Job Policy Setup form.

For more information, see the following examples:

Example 1

If the job policy specified using this variable has a *Max. Jobs* value of `5` and the job policy specified in the Job Policy Setup form has a *Max. Jobs* value of `10`, ADE XL uses a *Max. Jobs* value of `10` for simulation runs.

Example 2

If the job policy specified using this variable has a *Simulation Run Timeout* value of `600` and the job policy specified in the Job Policy Setup form does not have a *Simulation Run Timeout* value, ADE XL uses a *Simulation Run Timeout* value of `600` for simulation runs.

In `.cdsenv`:

```
adexl.icrpStartup defaultJobPolicy string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "defaultJobPolicy" 'string "" )
```

Valid Values:

Any valid job policy name

Note the following:

- Do not use the `.jp` job policy file extension when specifying the policy name. For example, specify `myPolicy` instead of `myPolicy.jp`.
- If the job policy name you specify is not defined, setting this environment variable does nothing and the program reverts to the default job policy settings or whatever you select on the Job Policy Setup form in the environment.

enableOutdir

Enables or disables the `-outdir` option, which refers to compiled verilogA module, in the APS or Spectre run script. By default, `-outdir` is included in the script.

In `.cdsenv`:

```
adexl.icrpStartup enableOutdir boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "enableOutdir" 'boolean nil)
```

Valid Values:

`t` : Includes the `-outdir` option, which refers to compiled verilogA module, in the APS or Spectre run script.

`nil` : Removes the `-outdir` option, which refers to compiled verilogA module, from the APS or Spectre run script.

Default Value: `t`

newCdsXVNCForeachICRP

Specifies if a new VNC server must be launched for each ICRP process.

Note: It is recommended to set this variable to `t` if you are using RHEL6.0+ with LSF 9.0+.

In `.cdsenv`:

```
adexl.icrpStartup newCdsXVNCForeachICRP boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "newCdsXVNCForeachICRP" 'boolean nil)
```

Valid Values:

`t` : Launches a new VNC server for each ICRP process.

`nil` : Multiple ICRP processes on the same machine will connect to the same cdsXvnc, up to a maximum of 255 simultaneous connections for a single cdsXvnc.

Default Value: `nil`

refreshCDF

Specifies when to refresh CDF to consider the base-level CDF values for netlist generation and to ignore the user-level CDF changes.

In `.cdsenv`:

```
adexl.icrpStartup refreshCDF cyclic "Always"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "refreshCDF" 'cyclic "UnlessUserCDF")
```

Valid Values:

`Always` : Always refreshes the CDF to consider the base values.

`Never` : Never refreshes the CDF. Set this value to use user-level CDF settings for ADE XL netlisting.

`UnlessUserCDF` : Refreshes the CDF only if user-level CDF is not available.

Default Value: `Always`

Any user-level modification in the CDF must be included in `.cdsinit`, so that ICRP picks it for correct netlisting. For example:

CDF is modified using the SKILL file *`modify_siminfo.il`*:

```
cdf=cdfGetCellCDF(ddGetObj("analogLib" "pmos"))
cdf->simInfo->spectre->instParameters=list('w 'l 'as)
cdfSaveCDF(cdf)
```

After the modification, include the *`modify_siminfo.il`* file in `.cdsinit`:

```
envSetVal("adexl.icrpStartup"  "refreshCDF" 'cyclic "UnlessUserCDF")
;envSetVal("adexl.icrpStartup"  "refreshCDF" 'cyclic "Always")
;envSetVal("adexl.icrpStartup"  "refreshCDF" 'cyclic "Never")
load("modify_siminfo.il")
```

showJobStdout

Specifies whether you want standard output messages from the job submit command (those that the program writes to standard output) to appear in the output area of the Command Interpreter Window (CIW).

Note: You can use this setting to debug problems that might occur while running jobs in Local, Remote-Host or Command mode.

In `.cdsenv`:

```
adexl.icrpStartup showJobStdout boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "showJobStdout" 'boolean nil )
```

Valid Values:

`t`	Write standard output messages from the job submit command to the CIW
`nil`	Do not write standard output messages from the job submit command to the CIW

showJobStderr

Specifies whether you want standard error messages from the job submit command (those that the program writes to standard error) to appear in the output area of the Command Interpreter Window (CIW).

Note: You can use this setting to debug problems that might occur while running jobs in Local, Remote-Host or Command mode.

In `.cdsenv`:

```
adexl.icrpStartup showJobStderr boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "showJobStderr" 'boolean nil )
```

Valid Values:

`t`	Write standard error messages from the job submit command to the CIW
`nil`	Do not write standard error messages from the job submit command to the CIW

showOutputLogOnError

Important

This variable is obsolete from the IC6.1.2 release and will be removed in a future release. Instead of specifying this variable, do one of the following:

- ❑ Select the *Show output log on error* check box in the Job Policy Setup form.
- ❑ Use the `axlSetJobPolicyProperty` SKILL function to specify the default behavior.

  For example, use the following function to display the simulation log file when the program encounters a simulation error:

  ```
  axlSetJobPolicyProperty (<policy_name> "showoutputlogerror" "1")
  ```

  Where the boolean value `"1"` specifies that the simulation log file must be displayed when the program encounters a simulation error. Use the value `"0"` to specify that the simulation log file must not be displayed.

For more information about the `axlSetJobPolicyProperty` SKILL function, see the *Virtuoso Analog Design Environment XL SKILL Reference*.

Specifies whether you want the program to display the simulation log file when it encounters a simulation error. Equivalent to selecting (`t`) or deselecting (`nil`) the *Show output log on error* check box in the Job Policy Setup form).

In `.cdsenv`:

```
adexl.icrpStartup showOutputLogOnError boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.icrpStartup" "showOutputLogOnError" 'boolean nil )
```

Valid Values:

`t`	Display the simulation log file when there is a simulation error
`nil`	Do not display the simulation log file

adexl.results

checksAssertsFiltersPath

Specifies a colon-separated list of directories containing Check/Asserts filter definitions (XML files).

In `.cdsenv`:

```
adexl.results checksAssertsFilterPath string "/myPath/myDir:/
     myPath1/myDir1"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.results" "checksAssertsFilterPath" 'string "/
     myPath/myDir:/myPath1/myDir1")
```

Valid Values:

Valid directory path(s) containing filter definitions.

Default Value: `nil`

checksAssertsViewTool

Specifies the browser in which you want to view the following two violation filter reports for Checks/Asserts:

- Netlisted Checks/Asserts
- Dynamic/Static Violation Report

In `.cdsenv`:

```
adexl.results checksAssertsViewTool string "firefox"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.results" "checksAssertsViewTool" 'string "firefox")
```

Valid Values:

Name of the browser executable in which you want to view the violation filter reports for Checks/Asserts.

Default Value: `"firefox"`

defaultBackAnnotationOption

Specifies the default option to be used while backannotating the values from the ADE XL results to the design schematic and ADE XL setup.

In `.cdsenv`:

```
adexl.results defaultBackAnnotationOption cyclic "All variables and
     parameters"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.results" "defaultBackAnnotationOption" 'cyclic "All
     variables and parameters")
```

Valid Values:

`"All variables and parameters"`	Backannotates all the global variables and device parameters
`"Only design variables"`	Backannotates only the global variables
`"Only device parameters"`	Backannotates only the device parameters

`"None"`	Does not backannotate any value

Default Value:

```
"All variables and parameters"
```

defaultResultsViewForMonteCarlo

Specifies the default results view for the Monte Carlo Sampling run mode.

In `.cdsenv`:

```
adexl.results defaultResultsViewForMonteCarlo cyclic "Detail"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.results" "defaultResultsViewForMonteCarlo" 'cyclic
     "Detail - Transpose")
```

Valid Values:

```
Detail, Detail - Transpose, Status, Summary, Yield
```

Default Value:

```
Yield
```

defaultResultsViewForSweepsCorners

Specifies the default results view for the Single Run, Sweeps, and Corners run mode.

In `.cdsenv`:

```
adexl.results defaultResultsViewForSweepsCorners cyclic "Detail"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.results" "defaultResultsViewForSweepsCorners"
     'cyclic "Detail-Transpose")
```

Valid Values:

```
Detail, Detail - Transpose, Optimization, Status, Summary, Yield
```

Default Value:

```
Detail
```

evalOutputsOnSimFailure

Controls the evaluation of outputs if an analysis fails. The setup for a particular test can contain more than one analysis. This variable controls how to display outputs for measurements in situations where the simulation for a particular analysis fails.

When this variable is set to `SkipFailedAnalyses` and the simulation for a particular analysis fails, then the expressions tied to the failed analysis are skipped. Their status is displayed as `sim err`.

The other outputs that are not dependent on the failing analysis will display the output value if the expression has been successfully evaluated, otherwise it will display `eval_error`.

Note: This environment variable is not supported for the Monte Carlo Sampling run mode.

In `.cdsenv`:

```
adexl.results evalOutputsOnSimFailure cyclic "SkipFailedAnalyses"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.results" "evalOutputsOnSimFailure" 'cyclic
     "SkipFailedAnalyses")
```

Valid Values:

`SkipFailedAnalyses`	Skips the expressions that are tied to failed analysis and calculates the results of other expressions. For the outputs that are tied to the failed analyses, the tool shows `sim err`. You can hover over the cell to display the tooltip with more details on the failed analysis. For the other outputs, which are not tied to a failed analysis, the tool shows the output value in case of successful evaluation. In case of an evaluation error, it displays `eval err`.
`None`	In case of a failed analysis, reports `sim err` for all the expressions.

`All`	All the expressions are evaluated irrespective of whether simulation has passed or failed. If any analysis fails, evaluation is done on partial data that is available in the simulation results directory.

Default Value: `SkipFailedAnalyses`

exportPreserveScalingFactors

By default, results are exported to CSV files in the scientific notation format. Set this environment variable to export results in the same format as they are displayed in the Results tab to the CSV file.

In `.cdsenv`:

```
adexl.results exportPreserveScalingFactors boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "exportPreserveScalingFactors" 'boolean t
     )
```

Valid Values:

`t`	Export results as they are displayed in the Results tab to the CSV file.
`nil`	Export results to CSV files in the scientific notation format.

Default Value: `nil`

retainReferenceSimResults

Controls whether the simulation results of the history item on which you ran *Re-run Unfinished/Error Points* are retained in that history item or not. For more information about running *Re-run Unfinished/Error Points*, see Simulating Only Error or Incomplete Points.

In `.cdsenv`:

```
adexl.results retainReferenceSimResults boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "retainReferenceSimResults" 'boolean t )
```

Valid Values:

`t`	Retains the simulation results and related netlists of the history item on which you ran *Re-run Unfinished/Error Points*.
`nil`	Does not retain the simulation results of the history item on which you ran *Re-run Unfinished/Error Points*. As a result, you will not be able to perform postprocessing operations (like plotting, printing, annotation, re-evaluation, and so on) on the history item.

Default Value: `nil`

saveDir

Specifies where you want the program to write results database information and run log files for an ADE XL session. When you set this environment variable, the program writes results database information and run log files to `*libraryName*/*cellName*/*viewName*/results` in the specified `saveDir` location.

If your design library is set up as read-only, you can use this environment variable to specify a writable location. See also the *ADE XL Results Database Location* field on the Save Options form that appears when you choose *Options – Save* in the ADE XL environment.

Note the following:

- If you do not specify a `saveDir`, the program writes results database information and run log files to `*libraryName*/*cellName*/adexl/results/data` in the ADE XL view.
- If you do not specify a `saveDir`, and you open the ADE XL view in read-only mode or do not have write permissions in the ADE XL view, the program writes results database information and run log files to `*libraryName*/*cellName*/adexl/results/data/<history_item>`

in the location specified using the asimenv.startup projectDir environment variable. The default setting for this environment variable is `$HOME/simulation`.

In `.cdsenv`:

```
adexl.results saveDir string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "saveDir" 'string "" )
```

Valid Values:

Any valid directory path

saveLocalPsfDir

If the useLocalPsfDir environment variable is set, use this environment variable to specify the path to the local directory on remote systems where the results for distributed simulation jobs run on each remote system must be saved.

Note: Ensure that the specified local directory path exists on all the remote systems on which a distributed simulation is run.

In `.cdsenv`:

```
adexl.results saveLocalPsfDir string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "saveLocalPsfDir" 'string "" )
```

Valid Values:

Any valid directory path

saveResDir

Specifies where you want the program to write simulation results generated during a run. When you set this environment variable, the program writes simulation results to *`libraryName`*`/`*`cellName`*`/adexl/results/data/<`*`history_item`*`>` in the specified `saveResDir` location.

If your design library is set up as read-only, you can use this environment variable to specify a writable location. See also the *Simulation Results Directory Location* field on the Save Options form that appears when you choose *Options – Save* in the ADE XL environment.

Note the following:

- If you do not specify a `saveResDir`, but specify a ADE XL results database location (see saveDir on page 109), the program writes simulation results to *`libraryName`*`/`*`cellName`*`/adexl/results/data/<`*`history_item`*`>` in the ADE XL results database location.
- If you do not specify a `saveResDir` or a ADE XL results database location (see saveDir on page 109), the program writes simulation results to *`libraryName`*`/`*`cellName`*`/adexl/results/data/<history_item>` in the location specified using the `asimenv.startup projectDir` environment variable. The default setting for this environment variable is `$HOME/simulation`.

In `.cdsenv`:

```
adexl.results saveResDir string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "saveResDir" 'string "" )
```

Valid Values:

Any valid directory path

saveResultsFromHistoryDir

Specifies a default value for the Save Directory field in the Save Results form that appears when you right-click a history item in the Data View pane, and choose *Save Results*.

In `.cdsenv`:

```
adexl.results saveResultsFromHistoryDir string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "saveResultsFromHistoryDir" 'string "" )
```

Valid Values:

Any valid directory path

useLocalPsfDir

By default, the results for distributed simulation runs are saved in the location specified using the `asimenv.startup projectDir` environment variable. Set this environment variable to save the results for distributed simulation jobs run on a remote system in a local directory on that system. Specify the local directory path using the saveLocalPsfDir environment variable.

In `.cdsenv`:

```
adexl.results useLocalPsfDir boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.results" "useLocalPsfDir" 'boolean t )
```

Valid Values:

`t`	Saves the results for distributed simulation jobs run on a remote system in a local directory on that system.
`nil`	Saves the results for distributed simulation runs are saved in the location specified using the `asimenv.startup projectDir` environment variable.

Default Value: `nil`

adexl.gui

autoCornerUpdate

Specifies if any changes related to corners or tests in the setup database should be automatically reflected in the *Corners Setup* form that is already open. When set to `t`, the already open *Corners Setup* form is automatically updated to show the changes.

When this variable is set to `nil`, the details are not automatically updated in the *Corners Setup* form. You need to close and re-open the form to view the updated details.

In `.cdsenv`:

```
adexl.gui autoCornerUpdate boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "autoCornerUpdate" 'boolean t )
```

Valid Values:

`t`	Automatically updates the *Corners Setup* form with the changes in corner and test details.
`nil`	Does not automatically update the *Corners Setup* form with the changes in corner and test details. You need to close and re-open the form to view the updated details.

Default Value: `nil`

continueJobsOnExitQuery

Specifies whether to continue running the in-progress simulations before exiting an ADE XL environment. This variable overrides the value of the continueICRPRunOnAbruptGUIExit environment variable.

In `.cdsenv`:

```
adexl.gui continueJobsOnExitQuery cyclic "Stop"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "continueJobsOnExitQuery" 'cyclic "Ask")
```

Valid Values:

`"Ask"` If any simulation is in progress when a session is being exited, ADE XL prompts you to confirm if the in-progress simulations are to be continued.

`"Stop"` ADE XL stops all the in-progress simulations before exiting the session.

Even if the continueICRPRunOnAbruptGUIExit variable is set to `t`, the simulations are stopped.

`"Continue"` ADE XL continues to run and complete the in-progress simulations after exiting the session.

Even if the continueICRPRunOnAbruptGUIExit variable is set to `nil`, the simulations are completed.

Default Value: `"Ask"`

copyMeasurementScripts

Controls whether the OCEAN script file specified for an output of type *MATLAB script* or *OCEAN script* in the Outputs Setup tab is copied to the ADE XL view or used from the original location.

In `.cdsenv`:

```
adexl.gui copyMeasurementScripts boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "copyMeasurementScripts" 'boolean t )
```

Valid Values:

`t`	Copies the Matlab or OCEAN script file to the ADE XL view. Only the file in the ADE XL view is used for simulation runs. As a result, any changes in the original file will not be applied for subsequent simulation runs.
`nil`	Does not copy the Matlab or OCEAN script file to the ADE XL view. The original file is used for simulation runs. As a result, any changes in the original file will be applied for subsequent simulation runs.

Default Value: `t`

Note: Any change in the value of this environment variable will be applied only to new outputs of type *MATLAB script* or *OCEAN script* that you add in the Outputs Setup tab. For example, if you add an OCEAN script output named `OCEAN1` when the value of this variable is `t`, the script file specified for the output is copied to the ADE XL view. However, if you later change the value of this environment variable to `nil`, the script file in the ADE XL view for the OCEAN script output named `OCEAN1` will continue to be used. The original file will not be used.

copyPreRunScripts

Controls whether or not the simulation uses a copy (snapshot) of the pre-run script that is saved inside the ADE XL view when you browse to the file and set it up as the pre-run script for the test.

In `.cdsenv`:

```
adexl.gui copyPreRunScripts boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "copyPreRunScripts" 'boolean t )
```

Valid Values:

`t` Each point in the simulation run uses the snapshot of the pre-run script that was copied into the ADE XL view when the file was selected in the Pre-Run Script setup form. Any changes to the pre-run script file are ignored because the snapshot is used.

`nil` Each point in the simulation run uses the pre-run script file that is maintained outside of the ADE XL view. Any changes in the script file are considered in the simulation run because the current version of the file is used.

Note: A snapshot of the pre-run script file will still be found in the ADE XL view for simulation purposes. The difference is that this snapshot is created at the simulation run time instead of the time when the script was first selected in the Pre-Run Script setup form.

Default Value: `t`

Note: Any change in the value of this environment variable will be applied only to the new simulation runs. For example, if you add a pre-run script when the value of this variable is `t`, the specified file is copied to the ADE XL view. However, if you later change the value of this environment variable to `nil`, the script file that has already been copied to the ADE XL view will continue to be used. The original file will not be used.

confirmReEvaluationWhen

Specifies when to show a confirmation dialog while re-evaluating results for a history.

In `.cdsenv`:

```
adexl.gui confirmReEvaluationWhen cyclic "Never" nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "confirmReEvaluationWhen" 'cyclic "Never" )
```

Valid Values:

`"Always"`	Always prompts the user to confirm if the results need to be re-evaluated.
`"AgeHoursThresholdExceeded"`	If the age of the history for which the results are to be re-evaluated is more than the threshold age specified by the reEvaluationAgeHoursThreshold environment variable, ADE XL prompts the user to confirm that re-evaluation is to be run.
`"RemovingOutputsThresholdExceeded"`	If the outputs in a history item are more than the outputs in the active setup and the difference between the number of outputs in the active setup and the history has exceeded the threshold value specified by the reEvaluationRemovingOutputsThreshold environment variable, ADE XL prompts the user to confirm that re-evaluation is to be run.
`"Never"`	Never prompts the user for confirmation. Instead, it always re-evaluates the results for previous history items.

Default Value `"Never"`

continueJobsOnExitQuery

Specifies whether to continue running the in-progress simulations before exiting an ADE XL session. This variable overrides the value of the `continueICRPRunOnAbruptGUIExit` environment variable.

In `.cdsenv`:

```
adexl.gui continueJobsOnExitQuery cyclic "Ask" nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "continueJobsOnExitQuery" 'cyclic "Stop" )
```

Valid Values:

`Ask`	Prompts the user to confirm if the already running jobs need to be continued after exiting ADE XL.
`Stop`	Always stops the already running jobs after exiting ADE XL.
`Continue`	Always continues the already running jobs after exiting ADE XL.

Default Value `Ask`

defaultCorners

Specifies the default corners setup (.sdb) file you want the program to load onto the Corners Setup form. Using this variable, you cannot load corners from a .csv file.

Note: The default corners will be loaded only if no other corners are defined in the Corners Setup form.

In `.cdsenv`:

```
adexl.gui defaultCorners string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "defaultCorners" 'string "" )
```

Valid Values:

String containing the path to a valid ADE XL corners setup file.

Default Value `""`

defaultCornerExportFileFormat

Specifies the default format in which the corner details exported from the Corners Setup form are saved.

In `.cdsenv`:

```
adexl.gui defaultCornerExportFileFormat cyclic "CSV"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "defaultCornerExportFileFormat" 'cyclic "SDB")
```

Valid Values:

`CSV`	Specifies that by default the corner details are to be saved in a `.csv` file
`SDB`	Specifies that by default the corner details are to be saved in a `.sdb` file
Default Value	`CSV`

Note: This variable is considered only before opening the Corners Setup form for the first time in an ADE XL session. After the form is opened in a session, the format used or specified in the form is saved as a user preference.

defaultCornerImportFileFormat

Specifies the default format from which the corner details are to be imported into the Corners Setup form.

In `.cdsenv`:

```
adexl.gui defaultCornerImportFileFormat cyclic "CSV"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "defaultCornerImportFileFormat" 'cyclic "SDB")
```

Valid Values:

`CSV`	Specifies that by default the corner details are to be imported from a `.csv` file
`SDB`	Specifies that by default the corner details are to be imported from a `.sdb` file
`PCF`	Specifies that by default the corner details are to be imported from a `.pcf` file

Default Value	`CSV`

Note: This variable is considered only before opening the Corners Setup form for the first time in an ADE XL session. After the form is opened in a session, the format used or specified in the form is saved as a user preference.

defaultParametersAssistantFilter

Specifies the default filter to be set for device instance parameters in the Parameters tab in the Variables and Parameters assistant.

In `.cdsenv`:

```
adexl.gui defaultParametersAssistantFilter string "MyFilter"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "defaultParametersAssistantFilter" 'strng
     "MyFilter")
```

Valid Values:

`Whitelist`	Displays parameters specified in the list that you can edit by using the Filter Setup form.
`Automatic`	Displays the parameters list by object, which can be further expanded to view the object names
`Editable`	Displays the parameters list by object, which can be further expanded to view the object names
`All`	Displays all the parameters
`Custom`	Displays the parameters as defined in the custom filter
Default Value	`"Whitelist"`

defaultParametersViewBy

Specifies the default view to be set for the *Parameters* tab in the Variables and Parameters assistant.

In `.cdsenv`:

```
adexl.gui defaultParametersViewBy cyclic "Object"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "defaultParametersViewBy" 'cyclic "Object")
```

Valid Values:

`Property`	Displays the parameters list by property names, which can be further expanded to view the object names
`Object`	Displays the parameters list by object, which can be further expanded to view the object names
Default Value	`"Property"`

Also see: Sorting Parameters by Properties and Objects

defaultParametersWhitelist

Specifies the default list of parameters to be displayed when the filter on the *Parameters* tab in the Variables and Parameters assistant is set to `Whitelist`.

In `.cdsenv`:

```
adexl.gui defaultParametersWhitelist string "c C cap* finger fingers
     fw l"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "defaultParametersWhitelist" 'string "c C cap*
     finger fingers fw l")
```

Valid Values:

	A space separated list of parameter names
Default Value	`"C c cap* finger* fw L l length M m nFin nf nfin numFinger* R r res* simM W Wfg w wf width"`

Also see: Filtering Device Instance Parameters

descendIntoSubcktForShowingInstOrNet

Specifies if a lower level subcircuit is to be opened to show the net or instance that violates a circuit check or assert.

In `.cdsenv`:

```
adexl.gui descendIntoSubcktForShowingInstOrNet boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "descendIntoSubcktForShowingInstOrNet"
     'boolean t)
```

Valid Values:

`t`	Opens the lower level subcircuit that contains the net or instance
`nil`	Opens the top hierarchy level that contains the subcircuit
Default Value	`t`

detailViewShowDefault

Specifies the default columns to be displayed in the *Detail* results view.

In `.cdsenv`:

```
adexl.gui detailViewShowDefault string "\"Test\" \"\"Test\" \"Spec/
     Weight\" \"Min/Max\" \"Corners\" \"Waveform Expressions\" \"Post
     Run Results\" \"Device Checks\" \"User-Defined Columns\"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "detailViewShowDefault" 'string "\"Test\"
     \"Spec/Weight\" \"Min/Max\" \"Corners\" \"Waveform Expressions\"
     \"Post Run Results\" \"Device Checks\" \"User-Defined
     Columns\"")
```

Valid Values:

A string with the list of space-separated column names:

```
\"Test\"

\"Spec/Weight\"

\"Min/Max\"

\"Corners\"

\"Scalar Expressions\"

\"Waveform Expressions\"

\"Signals\"

\"Corner Expressions\"

\"Post Run Results\"

\"Device Checks\"

\"User-Defined Columns\"
```

Default Value:

```
"\"Test\" \"Spec/Weight\" \"Min/Max\" \"Corners\" \"Scalar
     Expressions\" \"Waveform Expressions\" \"Signals\" \"Corner
     Expressions\" \"Post Run Results\" \"Device Checks\"
     \"User-Defined Columns\""
```

detailtransposeViewShowDefault

Specifies the default columns to be displayed in the *Detail-Transpose* results view.

In `.cdsenv`:

```
adexl.gui detailtransposeViewShowDefault string "\"Signals\" \"Corner
     Expressions\" \"Sweep Expressions\" \"Top Level Expressions\"
     \"Device Checks\""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "detailtransposeViewShowDefault" 'string
     "\"Signals\" \"Corner Expressions\" \"Sweep Expressions\" \"Top
     Level Expressions\" \"Device Checks\"" )
```

Valid Values:

A string with the list of space-separated column names:

```
\"Scalar Expressions\"

\"Waveform Expressions\"

\"Signals\"

\"Corner Expressions\"

\"Sweep Expressions\"

\"Top Level Expressions\"

\"Device Checks\"
```

Default Value:

```
"\"Scalar Expressions\" \"Waveform Expressions\"  \"Signals\"
     \"Corner Expressions\" \"Sweep Expressions\" \"Top Level
     Expressions\" \"Device Checks\""
```

disableConstraintsRead

Controls if ADE XL needs to elaborate the Constraint Manager hierarchy to find the matched parameter constraints and import them to the ADE XL setup. By default, this variable is set to `nil` and ADE XL elaborates the Constraint Manager hierarchy. Set this variable to `t` in any one of the following two cases:

- When the Constraints Manager does not contain matched parameters for your design.
- When you do not wish to automatically import all the matched parameters from the Constraints Manager to ADE XL.

Disabling constraints read helps in improving the performance of ADE XL.

In `.cdsenv`:

```
adexl.gui disableConstraintsRead boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "disableConstraintsRead" 'boolean t )
```

Valid Values:

`t`	Disables elaboration of the Constraint Manager hierarchy to find matched parameter constraints.
`nil`	Enables elaboration of the Constraint Manager hierarchy to find matched parameter constraints.

Default Value:	`nil`

disableNominalSimulation

Controls whether the *Nominal Corner* check box in the Run Summary assistant pane is selected or deselected by default when you start ADE XL. If the *Nominal Corner* check box is selected, the simulator runs nominal corner simulation.

In `.cdsenv`:

```
adexl.gui disableNominalSimulation boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "disableNominalSimulation" 'boolean t )
```

Valid Values:

`t`	Deselects the *Nominal Corner* check box by default when you start ADE XL.
`nil`	Selects the *Nominal Corner* check box by default when you start ADE XL.

Default Value: `nil`

disableRunInReadOnly

Controls whether simulations can be run when the ADE XL view is opened in read-only mode. For more information, see Working with Read-Only maestro Views.

In `.cdsenv`:

```
adexl.gui disableRunInReadOnly boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "disableRunInReadOnly" 'boolean t )
```

Valid Values:

`t`	Does not allows simulations to be run in ADE XL when the ADE XL view is opened in read-only mode.
`nil`	Allows simulations to be run in ADE XL when the ADE XL view is opened in read-only mode.

Default Value:	`nil`

disableSimulationsDefault

Specifies whether the nominal corner or other corners (corners other than the nominal corner) are enabled or disabled by default when you create a new ADE XL view.

Tip

This environment variable can be used along with the defaultCorners environment variable to only run the provided list of corners.

In `.cdsenv`:

```
adexl.gui disableSimulationsDefault cyclic "nominal"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "disableSimulationsDefault" 'cyclic "nominal"
    )
```

Valid Values:

`nominal`

Disables the nominal corner when you create a new ADE XL view.

The *Nominal Corner* check box in the Run Summary assistant pane is deselected by default.

Note: If tests are enabled in the ADE XL view but no corners are specified, the *Nominal Corner* check box in the Run Summary assistant pane is automatically enabled so that simulations can be run.

`corners`

Disables other corners (corners other than the nominal corner) when you create a new ADE XL view.

The *Corner* check box in the Run Summary assistant pane is deselected by default.

`none`

Enables all the corners when you create a new ADE XL view.

Default Value:

```
none
```

enableAutoRefreshSetupSummary

Enables or disables automatic refresh of information in the *Setup Summary* section on the *Run Preview* tab.

In `.cdsenv`:

```
adexl.gui enableAutoRefreshSetupSummary boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "enableAutoRefreshSetupSummary" 'boolean t )
```

Valid Values:

`t`	Enables automatic refresh of the setup summary details on the *Run Preview* tab.
`nil`	Disables automatic refresh of the setup summary details on the *Run Preview* tab.

Default Value: `t`

Also see: Using the Run Preview

enableAutoRefreshPointsTable

Enables or disables automatic refresh of the points table on the *Run Preview* tab.

In `.cdsenv`:

```
adexl.gui enableAutoRefreshPointsTable boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "enableAutoRefreshPointsTable" 'boolean t )
```

Valid Values:

`t`	Enables automatic refresh of the points table on the *Run Preview* tab.
`nil`	Disables automatic refresh of the points table on the *Run Preview* tab.

Default Value: `nil`

Also see: Using the Run Preview

enableDeviceChecking

Enables or disables the use of the Device Check Specifications form.

Starting IC6.1.7/ICADV12.2, circuit and device checks can be specified using the *Check/Asserts* tree in the Data View pane, whereas in IC6.1.6/ICADV12.1, device checks were specified using the Device Check Specifications form. Setting `enableDeviceChecking` to `t` enables the devices checks flow as in IC6.1.6/ICADV12.1.

Note:

- This setting is recommended if you are unable to migrate to the new Checks/Asserts flow.
- If you are using an existing adexl view containing device checks setup, ADE XL will automatically enable the device checks flow, irrespective of the value of `enableDeviceChecking`.

In `.cdsenv`:

```
adexl.gui enableDeviceChecking boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "enableDeviceChecking" 'boolean nil)
```

Valid Values:

`t`	Enables *Device Check* user interface in ADE XL.
`nil`	Disables *Device Check* user interface in ADE XL.

Default Value: `nil`

forceShowAutomaticExpressions

Enables or disables the automatic display of the results of intermediate expressions irrespective of the setting of the *Plot* check boxes corresponding to those expressions on the *Output Setup* tab. Using this variable, you can now choose to display the results of the final expressions only and hide the results of their intermediate expressions.

In `.cdsenv`:

```
adexl.gui forceShowAutomaticExpressions boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "forceShowAutomaticExpressions " 'boolean t )
```

Valid Values:

`t`	Displays the results of intermediate expressions automatically even if their corresponding *Plot* check boxes are not selected. The results are displayed in all the views on the *Results* tab*,* *Spec Summary* table, or datasheets.
`nil`	The results of intermediate expressions are displayed only if the *Plot* check boxes corresponding to those measures are selected on the *Outputs Setup* tab.

Default Value: `nil`

Also see: Creating Dependent Expressions

formatSpecValues

Specifies format of specification values for displaying results.

In `.cdsenv`:

```
adexl.gui formatSpecValues cyclic "AsEntered"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "formatSpecValues" 'cyclic "SuffixNotation")
```

Valid Values:

`"AsEntered"`	Sets the format as entered by you.
`"AsPrintNotationCdsenv"`	Sets the format as specified in the `auCore.userPref printNotation` environment variable.
`"SuffixNotation"`	Sets the format as suffix notation.

Default Value: `"AsEntered"`

filterCDFParamsWithZeroOrNegativeOneDefValue

Displays or hides CDF parameters in the Variables and Parameters assistant that have default value set as `0` or `-1`. By default, variables that have default values set to any one of t, "", "0", "-1", 0, or -1 are not displayed in the Variables and Parameters assistant.

In `.cdsenv`:

```
adexl.gui filterCDFParamsWithZeroOrNegativeOneDefValue boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui"
     "filterCDFParamsWithZeroOrNegativeOneDefValue" boolean nil )
```

Valid Values:

`t`	Hides or filters out those CDF parameters from the Variables and Parameters assistant that have default value set as `0` or `-1`.
`nil`	Displays those CDF parameters in the Variables and Parameters assistant that have default value set as `0` or `-1`.

Default Value:

```
t
```

headerAlignmentSide

Controls the text alignment of the specification column headers in the *Detail-Transpose* view.

In `.cdsenv`:

```
adexl.gui "headerAlignmentSide" 'string "Center"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "headerAlignmentSide" 'string "Center" )
```

Valid Values:

`Left`	Left aligns the column header.
`Right`	Right aligns the column header.
`Center`	Center aligns the column header.

Default Value:

```
Left
```

headerTruncationDirection

Specifies the direction in which the column header needs to be truncated. ADE XL truncates the column header along the direction specified by this variable when this variable is used with the `headerTruncationWidth` variable, thus retaining the number of characters specified by `headerTruncationWidth`.

In `.cdsenv`:

```
adexl.gui "headerTruncationDirection" 'string "Right"
```

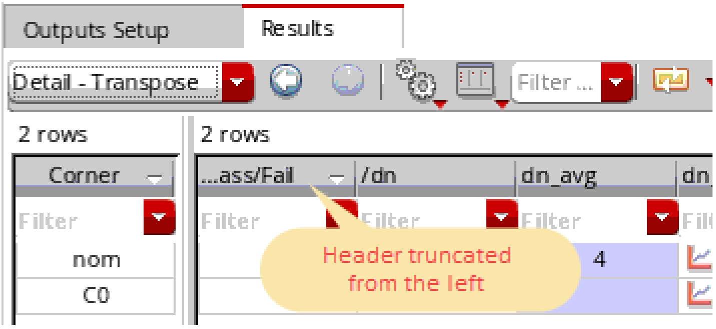
In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "headerTruncationDirection" 'string "Right" )
```

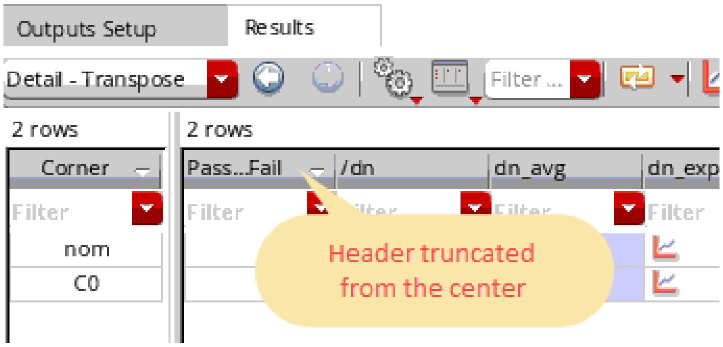
Valid Values:

`"Left"` Truncates the column header from the left and leaves the characters from the right.

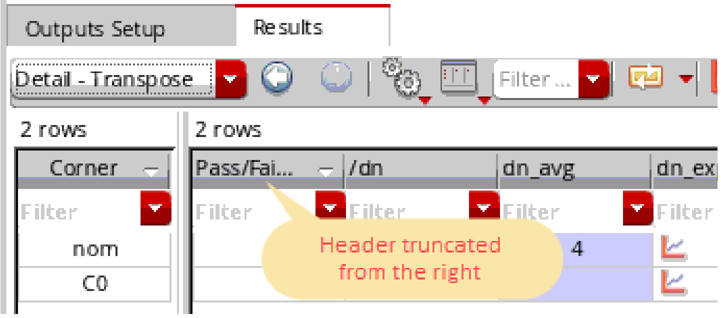
For example, if `headerTruncationWidth` is set to `8` and `headerTruncationDirection` is set to `Left`, the column headers appear as shown below.



`"Center"` Truncates the column header from the middle and leaves the characters from left and right, as shown in the example below.



`"Right"` Truncates the column header from the right and leaves the characters from the left, as shown in the example below.



Default Value:

`"Left"`

headerTruncationWidth

Specifies character length of the test name that appears in the specification column headers in the Detail-Transpose view.

In `.cdsenv`:

```
adexl.gui "headerTruncationWidth" 'int 30
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "headerTruncationWidth" 'int 30 )
```

Valid Values:

Any positive integer

Default Value:

```
24
```

LimitModelSections

Specifies how to handle errors when the model section name specified for a corner is not found in the corresponding model file or PCF file.

In `.cdsenv`:

```
adexl.gui LimitModelSections cyclic "InModelFile"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "LimitModelSections" 'cyclic "LimitedList" )
```

Valid Values:

`InModelFile`	If the section name specified for a corner is not present in model file, an error is displayed in Corners Setup form.
`LimitedList`	If the specified section name is not present in the PCF file, an error is displayed in Corners Setup form.
`No`	If the specified section name is not present in the model file or the PCF file, no error is displayed in Corners Setup form. However, an error is displayed during the simulation run.

Default Value:

`No`

maxNotesLength

Specifies the maximum character limit for a note, which can be added to an ADE XL element.

In `.cdsenv`:

```
adexl.gui maxNotesLength int 400
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "maxNotesLength" 'int 400 )
```

Valid Values:

Any positive integer between `1` and `5120`

Default Value:

```
512
```

See also:

- Adding Notes to a Test.

maxNotesRowsDisplay

Specifies the maximum limit of the number of lines of a note to be displayed in the tooltip for an ADE XL element.

In `.cdsenv`:

```
adexl.gui maxNotesRowsDisplay int 4
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "maxNotesRowsDisplay" 'int 4 )
```

Valid Values:

Any positive integer

Default Value:

```
10
```

See also:

- Adding Notes to a Test.

mismatchPairs

Specifies the default maximum number of device parameters for which mismatch results are displayed in the Show Mismatch form.

If the maximum number of device parameters in your design is lesser than this number, mismatch results are displayed for all the device parameters in your design. If the maximum number of device parameters in your design is greater than this number, mismatch results are displayed only for the number of device parameters specified using this environment variable.

For example, if you specify the value of the `mismatchPairs` variable as `20` and your design has `10` device parameters, mismatch results are displayed only for `10` device parameters. However, if your design has `25` design parameters, mismatch results are displayed only for the most sensitive `20` device parameters.

In `.cdsenv`:

```
adexl.gui mismatchPairs int 100
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "mismatchPairs" 'int 100 )
```

Valid Values:

Any positive integer.

Default Value:

```
200
```

modelSectionFilterFunction

Specifies a function used to filter the list of model sections displayed in the *Section* drop-down list in the Add/Edit Model Files form (see Adding Model Files to a Corner) that is opened from the Corners Setup form.

In `.cdsenv`:

```
adexl.gui modelSectionFilterFunction string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "modelSectionFilterFunction" 'string "" )
```

Valid Values:

String containing the name of a defined function that has the signature:

```
(t_fileName l_initialSectionList)
    => l_filteredSectionList
```

For example, if you have a model file named `mymodel.scs` that has the sections `tt`, `ss`, `fs`, and `unused`, do the following if you do not want the section `unused` to be displayed in *Section* drop-down list in the Add/Edit Model Files form that is opened from the Corners Setup form:

1. In your `.cdsinit` file or the CIW, define a function, say `CornerSectionFilt`, that specifies that the section `unused` must be filtered. For example:

```
procedure( CornerSectionFilt(model_file_name input_sections)
  let( ((file_tail car(last(parseString(model_file_name "/"))))
  output_sections)
  if( file_tail == "mymodel.scs" then
    output_sections = setof(name input_sections (name !=
"unused"))
  else
    output_sections = input_sections)
  output_sections))
```

2. Specify the `modelSectionFilterFunction` environment variable. For example, specify the following in your `.cdsenv` file:

```
adexl.gui modelSectionFilterFunction string "CornerSectionFilt"
```

numberOfBestPointsToView

Specifies the maximum number of best design points you want the program to display on the Results tab of the Outputs pane when you run an optimization in the Analog Design Environment GXL environment.

In `.cdsenv`:

```
adexl.gui numberOfBestPointsToView int 10
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "numberOfBestPointsToView" 'int 10 )
```

Valid Values:

Any positive integer

Default Value: 10

omitUndefinedVarsAndParamsInCornersCSV

Specifies whether to omit the variables and parameters that are not defined or found in the Data View assistant while exporting or importing corners from or to the Corners Setup form.

In `.cdsenv`:

```
adexl.gui omitUndefinedVarsAndParamsInCornersCSV boolean t
```

In `.cdsinit` or the CIW:

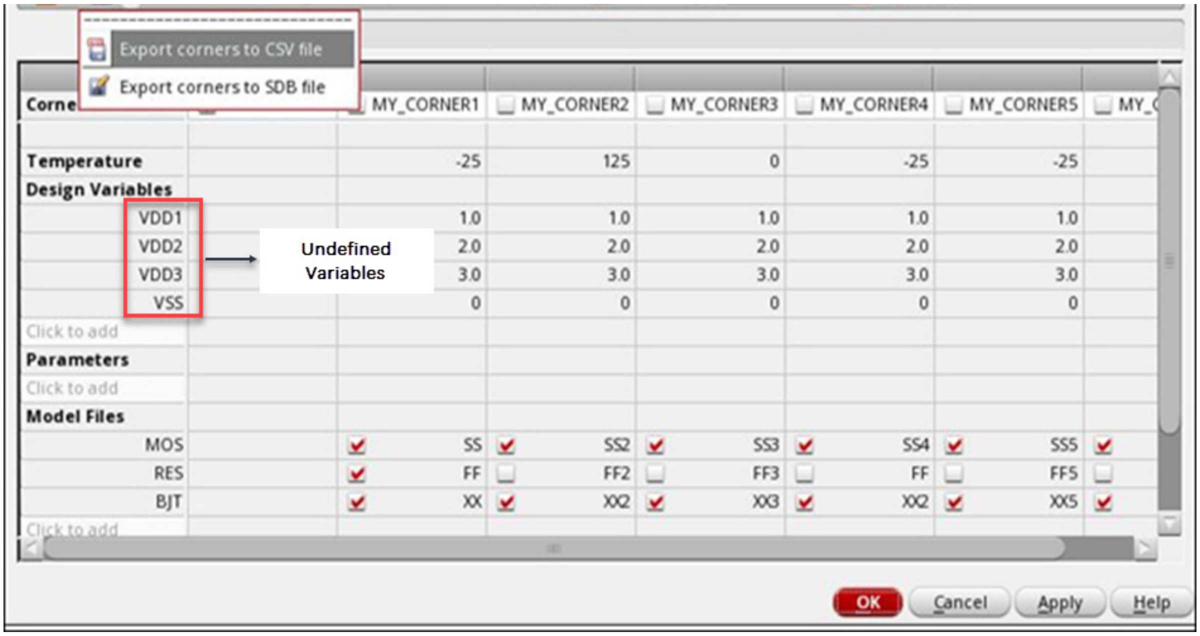
```
envSetVal( "adexl.gui" "omitUndefinedVarsAndParamsInCornersCSV"
     'boolean t)
```

Valid Values:

`t`	All undefined variables or parameters in Corners Setup are omitted during export to CSV
`nil`	All defined and undefined variables and parameters are imported or exported to CSV

Default Value: `t`

Defined and Undefined Variables or Parameters



Corne		□ MY_CORNER1	□ MY_CORNER2	□ MY_CORNER3	□ MY_CORNER4	□ MY_CORNER5	□ MY_
Temperature		-25	125	0	-25	-25	
Design Variables							
VDD1		1.0	1.0	1.0	1.0	1.0	
VDD2		2.0	2.0	2.0	2.0	2.0	
VDD3		3.0	3.0	3.0	3.0	3.0	
VSS		0	0	0	0	0	
Click to add							
Parameters							
Click to add							
Model Files							
MOS		✓ SS	✓ SS2	✓ SS3	✓ SS4	✓ SS5	✓
RES		✓ FF	□ FF2	□ FF3	□ FF	□ FF5	□
BJT		✓ XX	✓ XX2	✓ XX3	✓ XX2	✓ XX5	✓
Click to add							

Suppose you have an SDB file that contains variables for VDD1, VDD2, VDD3 and VSS. These variables have not been defined as design variables or global variables, and can be referred to as undefined variables. The variables that exist in the design or that are defined as global variables can be referred to as defined variables.

When this SDB file is imported to the Corners Setup form, all variables and parameters are imported to the Corners Setup irrespective of their being defined or undefined.

However, during CSV export, when `omitUndefinedVarsAndParamsInCornersCSV` is set to t (default), the undefined variables are not exported as these variables do not exist in the design.

Sample CSV export when `envSetVal( "adexl.gui" "omitUndefinedVarsAndParamsInCornersCSV" 'boolean t):`

```
Corner,MY_CORNER1,MY_CORNER2,MY_CORNER3,MY_CORNER4,MY_CORNE
R5,MY_CORNER6
Enable,f,f,f,f,f,f
Temperature,-25,125,0,-25,-25,125
Modelfile::./MY_TEST_CORNER/my.scs,t SS,t SS2,t SS3,t SS4,t SS5,t SS6

Modelfile::./MY_TEST_CORNER/my.scs,t FF,f FF2,f FF3,f FF,f FF5,f FF6
Modelfile::./MY_TEST_CORNER/my2.scs,t XX,t XX2,t XX3,t XX2,t XX5,t XX6
t Test::opamps:OpAmp_lab1_AC_top:1,t,t,t,t,t,t
f Test::AC,t,t,t,t,t,t
t Test::TRAN,t,t,t,t,t,t
```

If `omitUndefinedVarsAndParamsInCornersCSV` is set to nil, all defined and undefined variables are exported from within the file.

Sample CSV export when `envSetVal( "adexl.gui" "omitUndefinedVarsAndParamsInCornersCSV" 'boolean nil):`

```
Corner,MY_CORNER1,MY_CORNER2,MY_CORNER3,MY_CORNER4,MY_CORNE
R5,MY_CORNER6
Enable,f,f,f,f,f,f
Temperature,-25,125,0,-25,-25,125
VDD1,1.0,1.0,1.0,1.0,1.0,1.0
VDD2,2.0,2.0,2.0,2.0,2.0,2.0
VDD3,3.0,3.0,3.0,3.0,3.0,3.0
VSS,0,0,0,0,0,0
Modelfile::./MY_TEST_CORNER/my.scs,t SS,t SS2,t SS3,t SS4,t SS5,t SS6
Modelfile::./MY_TEST_CORNER/my.scs,t FF,f FF2,f FF3,f FF,f FF5,f FF6
Modelfile::./MY_TEST_CORNER/my2.scs,t XX,t XX2,t XX3,t XX2,t XX5,t XX6
t Test::opamps:OpAmp_lab1_AC_top:1,t,t,t,t,t,t
```

The above CSV export contains the undefined variables VDD1, VDD2, VDD3, and VSS.

openDesignAccessMode

Specifies the default mode in which designs are opened when you right-click a test in the Data View pane and choose *Open Design in Tab*.

In `.cdsenv`:

```
adexl.gui openDesignAccessMode cyclic "w"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "openDesignAccessMode" 'cyclic "w" )
```

Valid Values:

`r`	Opens designs in read mode.
`a`	Opens designs in append mode retaining the original design.
`w`	Deletes the original design and opens a new design.

Default Value: `r`

openDesignInNewTab

Controls whether a design is displayed in a new or current tab.

In `.cdsenv`:

```
adexl.gui openDesignInNewTab boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "openDesignInNewTab" 'boolean nil)
```

Valid Values:

`t`	Opens designs in the current tab.

`nil` — In config view:

If no design window is open, the Open Configuration window is displayed. Whereas, if design window already open, then it is made current.

In schematic view:

If no design window is open, the design is opened in a new tab. Whereas, if design window already open, then it is made current.

Other view type(s):

Design is always opened in a new tab.

Default Value: `nil`

openSchInWin

Controls whether a schematic opened from the Outputs Setup tab will be displayed in a new window, or in a new tab in the current window.

In `.cdsenv`:

```
adexl.gui openSchInWin boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "openSchInWin" 'boolean t )
```

Valid Values:

`t` — Displays the schematic opened from the Outputs Setup tab in a new window.

For example, if you right click a test name in the Outputs Setup tab and choose *To be Plotted*, the schematic is displayed in a new window.

`nil` — Displays the schematic opened from the Outputs Setup tab in a new tab in the current window.

openTerminalCommand

Specifies the shell command you want the program to use when you select *Open Terminal* from the context-sensitive menu for a history item on the Data Assistant pane. The program uses the shell command to open a terminal window in the directory containing results for the selected history item.

In `.cdsenv`:

```
adexl.gui openTerminalCommand string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "openTerminalCommand" 'string "" )
```

Valid Values:

Any valid command string

The default shell command when `openTerminalCommand` is not set (or when it is set to the empty string as shown above) is

```
xterm -T "historyResultsDirectory"
```

where *`historyResultsDirectory`* is the name of the directory containing results for the selected history item (such as `Interactive.0` or `GlobalOpt.1` or, for *ImproveYield* history items, the *historychildren* item name, such as `ImproveYield.0.GlobalOpt.0`). The `xterm` command must be in your path.

Note: You must not put `&` at the end of the `openTerminalCommand` string.

optimizationViewShowDefault

Specifies the default columns to be displayed in the *Optimization* results view.

In `.cdsenv`:

```
adexl.gui optimizationViewShowDefault string "\"Test\" \"Spec/
     Weight\" \"Min/Max\" \"Corners\" \"Scalar Expressions\"
     \"User-Defined Columns\"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "optimizationViewShowDefault" 'string
     "\"Test\" \"Spec/Weight\" \"Min/Max\" \"Corners\" \"Scalar
     Expressions\" \"User-Defined Columns\"")
```

Valid Values:

A string with the list of space-separated column names:

`\"Test\"`

`\"Spec/Weight\"`

`\"Min/Max\"`

`\"Corners\"`

`\"Scalar Expressions\"`

`\"Corner Expressions\"`

`\"Post Run Results\"`

`\"User-Defined Columns\"`

Default Value:

```
"\"Test\" \"Spec/Weight\" \"Min/Max\" \"Corners\" \"Scalar
     Expressions\" \"Corner Expressions\" \"Post Run Results\"
     \"User-Defined Columns\""
```

outputTabsShowDefault

Controls the display of the *Run Preview* and *Diagnostics* tabs in the Outputs pane.

In `.cdsenv`:

```
adexl.gui outputTabsShowDefault string ""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "outputTabsShowDefault" 'string "" )
```

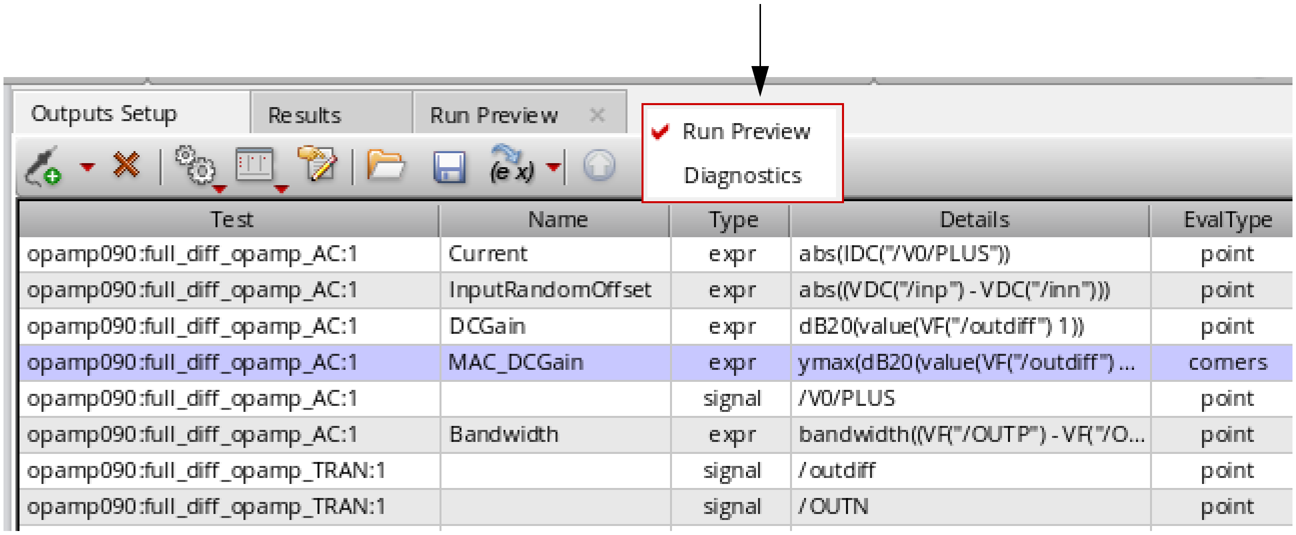
Valid Values:

`"\"Run Preview\""`	Displays the *Run Preview* tab.
`"\"Diagnostics\""`	Displays the *Diagnostics* tab.
`""`	Empty string. Hides both the tabs.

Default Value: `""`

Using GUI to Show/Hide

To display or hide the *Run Preview* or *Diagnostics* tabs using GUI, right-click outside the Outputs pane and select the required tab name.



Right-click to display tab names

Test	Name	Type	Details	EvalType
opamp090:full_diff_opamp_AC:1	Current	expr	abs(IDC("/V0/PLUS"))	point
opamp090:full_diff_opamp_AC:1	InputRandomOffset	expr	abs((VDC("/inp") - VDC("/inn")))	point
opamp090:full_diff_opamp_AC:1	DCGain	expr	dB20(value(VF("/outdiff") 1))	point
opamp090:full_diff_opamp_AC:1	MAC_DCGain	expr	ymax(dB20(value(VF("/outdiff") ...	corners
opamp090:full_diff_opamp_AC:1		signal	/V0/PLUS	point
opamp090:full_diff_opamp_AC:1	Bandwidth	expr	bandwidth((VR("/OUTP") - VR("/O...	point
opamp090:full_diff_opamp_TRAN:1		signal	/outdiff	point
opamp090:full_diff_opamp_TRAN:1		signal	/OUTN	point

pcfPrependBasePath

By default, when you import a process customization file (PCF) or design customization file (DCF) to create corners, the related process model file names are displayed in the Corners Setup form. However, you must specify the path to the directory containing the process models as an include path in the Simulation Files Setup form so that the simulator can read the process model files from the specified directory.

Use this environment variable to control whether the path to the process model files are included in the Corners Setup form so that you need not specify the path to the directory containing the process models as an include path in the Simulation Files Setup form.

For more information about importing PCF or DCF files, see Importing Corners From Customization Files.

In `.cdsenv`:

```
adexl.gui pcfPrependBasePath boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "pcfPrependBasePath" 'boolean t )
```

Valid Values:

`t`	Includes process model file paths in the Corners Setup form when you import PCF or DCF files.
`nil`	Does not include process model file paths when you import PCF or DCF files.

Default Value: `nil`

reEvalOnlyMostRecentHistory

Specifies if the re-evaluation of results can be done only for the recent history or for older history items as well.

In `.cdsenv`:

```
adexl.gui reEvalOnlyMostRecentHistory boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "reEvalOnlyMostRecentHistory " 'boolean nil)
```

Valid Values:

`t`	Re-evaluation of results is done only for the most recent history.
`nil`	Re-evaluation of results can also be done for older history items. For this, right-click a history and choose *View Results*. After the results are displayed on the Results tab, click *Re-evaluate* to evaluate the results according to the outputs defined on the Outputs Setup tab. If there are any changes in the output expressions and signals, the results are re-evaluated according to those changes. The re-evaluated results are written back to the results database of the history.

Default Value: `nil`

reEvaluationAgeHoursThreshold

Specifies the threshold age limit (in hours) against which ADE XL checks the age of the history before re-evaluating the results. If the age of a history is more than the specified threshold and the confirmReEvaluationWhen variable is set to `AgeHoursThresholdExceeded`, ADE XL prompts you to confirm that the results need to be re-evaluated.

In `.cdsenv`:

```
adexl.gui reEvaluationAgeHoursThreshold int 168
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "reEvaluationAgeHoursThreshold" 'int 168 )
```

Valid Values:

`int`	A positive integer value specifying the threshold age in hours

Default Value: `168`

reEvaluationMode

Specifies if the results of all or only the revised outputs need to be re-evaluated.

In `.cdsenv`:

```
adexl.gui reEvaluationMode cyclic "incremental" ("incremental" )
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "reEvaluationMode" 'cyclic "incremental" )
```

Valid Values:

`"incremental"`	Re-evaluates the results of only the revised outputs.
`"full"`	Re-evaluates the results of all the outputs.

Default Value: `"incremental"`

reEvaluationRemovingOutputsThreshold

Specifies the threshold limit for the difference in the number of outputs in the active ADE XL setup and the outputs in the history for which the results are being re-evaluated. If the difference is more than the threshold specified by this environment variable and the confirmReEvaluationWhen variable is set to `RemovingOutputsThresholdExceeded`, ADE XL prompts you to confirm that the results need to be re-evaluated.

In `.cdsenv`:

```
adexl.gui reEvaluationRemovingOutputsThreshold int 2
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "reEvaluationRemovingOutputsThreshold" 'int 2
     )
```

Valid Values:

	`int`	A positive number
Default Value:	`2`	

reEvaluationWhenActiveAndHistoryTestsDiffer

Specifies the action to be taken when the tests that exist in the history for which you are re-evaluating the results are not found in the active ADE XL setup.

In `.cdsenv`:

```
adexl.gui reEvaluationWhenActiveAndHistoryTestsDiffer cyclic
     "Disallow"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "reEvaluationWhenActiveAndHistoryTestsDiffer"
     'cyclic "Disallow" )
```

Valid Values:

`"Disallow"`	If the tests in the active ADE XL setup and the history setup do not match, the results are not re-evaluated for the history.
`"removeDeletedTestsFromHistory"`	The results are re-evaluated for the history. The results for the tests not found in the active ADE XL setup are removed from the results database for the history. Only the new results are saved.
`"persistDeletedTestsInHistory"`	The results are re-evaluated for the history. If the tests in the history setup and the active ADE XL setup are same, new results are written back to the results database. If any test from the history setup is not found in the active setup, the previous results for those tests are retained in the results database for the history.

Default Value: `"Disallow"`

saveStateQuery

Specifies the requirement to save any unsaved changes in the active state before exiting an ADE XL session.

In `.cdsenv`:

```
adexl.gui saveStateQuery cyclic "Stop"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "saveStateQuery" 'cyclic "Ask")
```

Valid Values:

`"Ask"`	If there are any unsaved changes in the active state, ADE XL prompts you to confirm if the changes are to be saved in the state.
`"saveIfPossible"`	If there are any unsaved changes in the active state, ADE XL tries to save the state. If the active state is read only, a warning message is displayed to indicate that the state cannot be saved.
`"Discard"`	Discards any unsaved changes and exits ADE XL.

Default Value: `"Ask"`

sendOutputsToEEFilter

Sets the default value for the *Send to Expression Editor* command on the Outputs toolbar. It specifies the type of outputs to be copied to the Expression Editor in ViVA XL Calculator.

In `.cdsenv`:

```
adexl.gui sendOutputsToEEFilter cyclic "Expressions"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "sendOutputsToEEFilter" 'cyclic "Expressions")
```

Valid Values:

`"All"`	Sends all the expressions and signals to the Expression Editor in ViVA XL Calculator.
`"Expressions"`	Sends only expressions to the Expression Editor.
`"Signals"`	Sends only signals to the Expression Editor

Default Value: `"Expressions"`

setHistoryPrefixToSetupStateNameOnLoad

Specifies if the name of the loaded setup state should be used as a prefix in the history name. By default, when you load a setup state and run simulation, the history name takes the setup state name as a prefix. To use `Interactive` as a prefix, set this variable to `nil`.

In `.cdsenv`:

```
adexl.gui setHistoryPrefixToSetupStateNameOnLoad boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "setHistoryPrefixToSetupStateNameOnLoad"
     'boolean t )
```

Valid Values:

`t`	Displays the setup state name as a prefix in the history name.
`nil`	Displays `interactive` as a prefix in the history name.

Default Value: `t`

setupFormDefaultEnabled

Specifies the check boxes that will be selected by default in the:

- *What to Import* group box in the Import Setup form (see Importing the Simulation Setup)
- *What to Export* group box in the Export Setup form (see Exporting the Simulation Setup)
- *What to Save* group box in the Save Setup State form (see Creating or Updating a Setup State)
- *What to Load* group box in the Load Setup State form (see Loading a Setup State)

In `.cdsenv`:

```
adexl.gui setupFormDefaultEnabled string "all"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "setupFormDefaultEnabled" 'string "all" )
```

Valid Values:

`all`	All the check boxes are selected by default.
`""`	Empty string. All the check boxes are deselected by default.

`tests`

`vars`

`parameters`

`currentmode`

`allsweepsenabled`

`allcornersenabled`

`defaultcornerenabled`

`runoptions`

`specs`

`corners`

`modelgroups`

`extensions`

List of option names separated by a comma, semicolon or space. The check boxes for the specified options are selected by default.

For example, to select the *Tests*, *Variables* and *Parameters* check boxes, specify the value as:

`"tests, vars, parameters"`

For more information about these options, see the following topics:

- Importing the Simulation Setup
- Exporting the Simulation Setup
- Creating or Updating a Setup State
- Loading a Setup State

Default Value: `all`

setupFormDefaultLoadOperation

Specifies the default value of the *Operation* drop-down list in the Load Setup State form (see Loading a Setup State) and the Import Setup form (see Importing the Simulation Setup).

In `.cdsenv`:

```
adexl.gui setupFormDefaultLoadOperation cyclic "retain"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "setupFormDefaultLoadOperation" 'cyclic
     "retain" )
```

Valid Values:

`retain`

`merge`

`overwrite`

For more information about these options, see the following topics:

- Loading a Setup State
- Importing the Simulation Setup.

Default Value:	`retain`

significantDigits

Specifies the number of significant digits you want the program to display for values in the *Nominal* column on the Results tab.

In `.cdsenv`:

```
adexl.gui significantDigits int 4
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "significantDigits" 'int 4 )
```

Valid Values:

Any integer from `2` through `15`

showSimLogForOnePointSim

Opens the simulator log file for a single test and single point run. This is similar to the ADE L environment.

In `.cdsenv`:

```
adexl.gui showSimLogForOnePointSim boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "showSimLogForOnePointSim" 'boolean t )
```

Valid Values:

`t`	Opens the simulator log file for a single test and single point run after the simulation is complete.
`nil`	Does not open the simulator log file by default. To view the output log for a point, right-click any result value for that point on the *Results* tab and choose *Output Log*.

Default Value `t`

specComparisonMode

Specifies the default comparison mode in the Spec Comparison form. For more information, see Comparing Results.

In `.cdsenv`:

```
adexl.gui specComparisonMode cyclic "Histories"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "specComparisonMode" 'cyclic "Histories" )
```

Valid Values:

`Histories`

`Design Points`

Default Value: `Histories`

statusViewShowDefault

Specifies the default sections to be displayed in the *Status* results view.

In `.cdsenv`:

```
adexl.gui statusViewShowDefault string "\"Progress Bar\" \"Status
     Table\""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "statusViewShowDefault" 'string "\"Progress
     Bar\" \"Status Table\"" )
```

Valid Values:

A string with the list of space-separated section names:

`\"Progress Bar\"`

`\"Status Table\"`

`\"Run Log\"`

Default Value: `"\"Progress Bar\" \"Status Table\" \"Run Log\""`

summaryViewShowDefault

Specifies the default columns to be displayed in the *Summary* results view.

In `.cdsenv`:

```
adexl.gui summaryViewShowDefault string "\"Test\"
     \"Mean-Median-Stddev\" \"User-Defined Columns\""
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "summaryViewShowDefault" 'string "\"Test\"
     \"Mean-Median-Stddev\" \"User-Defined Columns\"" )
```

Valid Values:

A string with the list of space-separated column names:

`\"Test\"`

`\"Mean-Median-Stddev\"`

`\"User-Defined Columns\"`

Default Value: `"\"Test\" \"Mean-Median-Stddev\" \"User-Defined Columns\""`

testsShownInOutputsSetup

Controls the display of outputs for selected tests in the *Outputs Setup* tab.

In `.cdsenv`:

```
adexl.gui testsShownInOutputsSetup cyclic "all"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "testsShownInOutputsSetup" 'cyclic "none"
     )
```

Valid Values:

`"all"`	Displays the outputs for all the tests defined in the Data View pane.
`"none"`	Hides the outputs for all the tests defined in the Data View pane.
`"enabled"`	Displays the outputs only for the tests that are enabled in the Data View pane. As you enable or disable tests in the Data View pane, ADE XL dynamically shows the outputs corresponding to the enabled tests.
`"disabled"`	Displays the outputs only for the tests that are disabled in the Data View pane. As you enable or disable tests in the Data View pane, ADE XL dynamically shows the outputs corresponding to the disabled tests.

Default Value `"all"`

Also see: Hiding and Showing Outputs

toolbarButtonStyle

Specifies whether the *Match Parameters* and *Ratio Matched Parameters* buttons on the Parameters tab of the Variables and Parameters assistant pane use an icon or text.

In `.cdsenv`:

```
adexl.gui toolbarButtonStyle cyclic "Histories"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "toolbarButtonStyle" 'cyclic "icon" )
```

Valid Values:

`"icon"`	If `icon`, the *Match Parameters* button uses the ▯=▯ icon and the *Ratio Matched Parameters* button uses the 1:n icon.
`"text"`	If `text`, the *Match Parameters* button uses the text *Match* and the *Ratio Matched Parameters* button uses the text *Ratio*.

Default Value: `"icon"`

yieldViewShowDefault

Specifies the default columns to be displayed in the yield view for Monte Carlo results.

In `.cdsenv`:

```
adexl.gui yieldViewShowDefault string "\"Min\" \"Target\" \"Max\"
     \"Mean\" \"Mean +-kSigma\" \"Std Dev\" \"Cpk\" \"Errors\" "
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.gui" "yieldViewShowDefault" 'string "\"Min\"
     \"Target\" \"Max\" \"Mean\" \"Mean +-kSigma\" \"Std Dev\"
     \"Cpk\" \"Errors\" ")
```

Valid Values:

`string`	A string with the list of space-separated column names

Default Value:

```
"\"Min\" \"Target\" \"Max\" \"Mean\" \"Std Dev\" \"Cpk\" \"Errors\"
     \"User-Defined Columns\""
```

zoomToProbedInstOrNet

Specifies if the net or instance that violates a check needs to be zoomed into when you probe it from the Checks/Asserts results view.

In `.cdsenv`:

```
adexl.gui zoomToProbedInstOrNet boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" " zoomToProbedInstOrNet " 'boolean t)
```

Valid Values:

`t` The schematic editor zooms into the probed net or instance. The zoom scale is controlled by the autoZoomScale environment variable.

`nil` The schematic editor does not zoom into the probed net or instance.

Default Value: `t`

adexl.cpupdtr

copyResultsData

Copies the simulation results when you copy an ADE XL view. For more information, see Copying Everything Contained in a maestro Cellview.

Note: The `copyResultsData` variable is used by both Virtuoso and a supporting utility. Therefore, it is recommended to set it in the `~/.cdsenv` file that is read by both. If you set this variable in the `<current-working-directory>/.cdsenv` file, you must also set the `CDS_LOAD_ENV` variable in the UNIX environment to CSF. The variable is not used when specified in the `.cdsinit` file.

In `.cdsenv`:

```
adexl.cpupdtr copyResultsData boolean  t
```

Valid Values:

`t`	Copy the simulation results data.
`nil`	Do not copy the simulation results data.

Default Value: `nil`

adexl.datasheet

author

Specifies the author name to be printed in the footer of the datasheet. By default, the UNIX login ID and name of the user is printed in the footer. You can use this variable to print a different author name.

In `.cdsenv`:

```
adexl.datasheet author string " "
```

In the CIW:

```
envSetVal( "adexl.datasheet" "author" 'string " " )
```

Valid Values:

A string value.

CSSFile

Specifies a custom Cascading Style Sheet (CSS) file for controlling the formatting of the main datasheet file and the datasheet files that contain the results information for each test. For more information, see Customizing the Datasheet Format and Structure.

In `.cdsenv`:

```
adexl.datasheet CSSFile string " "
```

In the CIW:

```
envSetVal( "adexl.datasheet" "CSSFile" 'string " " )
```

Valid Values:

String containing the path to a custom CSS file for the datasheet pages.

customFiles

Specifies the files to be copied to the datasheet directory when a datasheet is created.

Tip

If you have customized your datasheet format using custom XSLT stylesheets, you can use this environment variable to copy files such as the image file for your company logo and other support files that are required by the custom stylesheets. For more information about customizing the datasheet format, see Customizing the Datasheet Format and Structure.

In `.cdsenv`:

```
adexl.datasheet customFiles string " "
```

In the CIW:

```
envSetVal( "adexl.datasheet" "customFiles" 'string " " )
```

Valid Values:

String containing the path to a file or directory.

- If the path to a file is specified, only that file is copied to the datasheet directory when a datasheet is created.
- If the path to a directory is specified, all the files and sub-directories in the directory are copied to the datasheet directory when a datasheet is created.

mainDocXSLFile

Specifies a custom XSLT stylesheet for controlling the structure of the main datasheet file. For more information, see Customizing the Datasheet Format and Structure.

In `.cdsenv`:

```
adexl.datasheet mainDocXSLFile string " "
```

In the CIW:

```
envSetVal( "adexl.datasheet" "mainDocXSLFile" 'string " " )
```

Valid Values:

String containing the path to a custom XSLT file for the main datasheet page.

testDocXSLFile

Specifies a custom XSLT file for controlling the structure of the datasheet files that contain the results information for each test. For more information, see Customizing the Datasheet Format and Structure.

In `.cdsenv`:

```
adexl.datasheet testDocXSLFile string " "
```

In the CIW:

```
envSetVal( "adexl.datasheet" "testDocXSLFile" 'string " " )
```

Valid Values:

String containing the path to a custom XSLT file for the datasheet pages for each test.

waveformFileExtension

Specifies the default file extension for the ADE XL datasheet image files.

In `.cdsenv`:

```
adexl.datasheet waveformFileExtension string "pdf"
```

In the CIW:

```
envSetVal( "adexl.datasheet" "waveformFileExtension" 'string "pdf" )
```

Valid Values:

Any one of the valid formats listed below.

`"bmp"`, `"png"`, `"tiff"`, `"eps"`, `"pdf"`, `"ppm"`, `"jpg"`, `"jpeg"`, `"svg"`, `"xpm"`

Default Value	`""` When set to `""`, ADE XL uses the default image format from ViVA XL, which is png.

whatToSaveDefault

Specifies the check boxes to be enabled by default in the What to Save section of the Create Datasheet form.

In `.cdsenv`:

```
adexl.datasheet whatToSaveDefault string "\"Results\" \"Tests\"
     \"Waveforms\" \"Variables\" \"Setup\" \" Corners\" \"Schematic
     Images\""
```

In the CIW:

```
envSetVal( "adexl.datasheet" "whatToSaveDefault" 'string "\"Results\"
     \"Tests\" \"Waveforms\" \"Variables\" \"Setup\" \" Corners\"
     \"Schematic Images\"")
```

Valid Values:

A string listing space-separated check box names.

Default Value `"\"Results\" \"Tests\" \"Waveforms\" \"Variables\" \"Setup\" \"Corners\" \"Schematic Images\""`

ams.envOpts

exportOceanScriptWithNetlistDirSupport

If this variable is set to `t`, the `design` command in the exported ocean script uses the netlist path instead of the library, cell, and view names to specify the design to be netlisted and simulated. And the `resultsDir` command, which specifies the directory where the netlisting result is saved, is added to the ocean script.

In `.cdsenv`:

```
ams.envOpts exportOceanScriptWithNetlistDirSupport boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "ams.envOpts" "exportOceanScriptWithNetlistDirSupport"
     'boolean t)
```

Valid Values:

`t`	The following changes are done in the exported ocean script: ■ The `design` command uses the netlist path to specify the design to be netlisted and simulated, as shown below: `design("`*`path-to-the-netlist-file`*` / netlist")` ■ The `resultsDir` command is added to the ocean script
`nil`	The `design` command uses `lib/cell/view` format to specify the design to be netlisted and simulated, as shown below: `design("libName" "cellName" "viewName")`

Default Value: `nil`

asimenv

allowSignalsExpressionInSameSubwindow

Plots signals and expressions in the same subwindow. This variable works when you select some expressions and/or signals from the results, right-click them, and choose *Plot* or *Plot All options*.

In `.cdsenv`:

```
asimenv allowSignalsExpressionInSameSubwindow boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("asimenv" "allowSignalsExpressionInSameSubwindow" 'boolean
     t)
```

Valid Values:

`t`	Signals and expression waveforms are plotted in the same subwindow.
`nil`	Signals and expression waveforms are plotted in different subwindows.

Default Value: `t`

asimenv.startup

copyDesignVarsFromCellview

Controls copy of design variables from a cellview to the design variables of ADE XL test.

In `.cdsenv`:

```
asimenv.startup copyDesignVarsFromCellview boolean t
```

Valid Values:

`t`	Enables copy of design variables from the cellview property to a test
`nil`	Stops copying design variables from the cellview property.

Default Value: `t`

adexl.oceanxl

includeSimLogInJobLog

Controls whether the simulation log is to be included in the job log generated by the ICRP for an OCEAN run. By default, the log for an OCEAN run does not include the simulator output because for large simulations, this can result in large job log files. To save the simulator log in the ICRP job log for an OCEAN run, set this variable to `t` before running a simulation.

In `.cdsenv`:

```
adexl.oceanxl includeSimLogInJobLog boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.oceanxl" "includeSimLogInJobLog" 'boolean t )
```

Valid Values:

`t`	Includes the simulation log in the job log for an OCEAN run.
`nil`	Does not include the simulation log in the job log for an OCEAN run.

Default Value: `nil`

adexl.plotting

histogramBins

Specifies the default value for the *Number of Bins* field on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting histogramBins int 10
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "histogramBins" 'int 10 )
```

Valid Values:

```
A positive integer value
```

Default Value: `10`

histogramType

Sets the default value for the *Type* drop-down list on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting histogramType string "pass/fail"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "histogramType" 'string "pass/fail" )
```

Valid Values:

`pass/fail`	Plots a standard histogram with pass/fail spec markers
`standard`	Plots a standard histogram
`cumulative line`	Plots a cumulative line histogram
`cumulative box`	Plots a cumulative box histogram

Default Value: `pass/fail`

histogramQQPlot

Specifies the default value for the *Normal Quantile Plot* annotation option on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting histogramQQPlot boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "histogramQQPlot" 'boolean nil )
```

Valid Values:

`t`	Plots the quantile plots (Q-Q plots) along with the histogram
`nil`	Does not plot the quantile plots (Q-Q plots)

Default Value: `nil`

maxHistogramBins

Specifies the maximum limit for the *Number of Bins* field on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting maxHistogramBins int 500
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "maxHistogramBins" 'int 500)
```

Valid Values:

```
A positive integer value
```

Default Value: `1000`

modelFilesAre

Specifies how the model file paths are to be displayed in the plots and sweep visibility filter. Use this variable when the signal names and expressions contain long model file paths.

Important

Specify this variable before running simulation, and use the same value while plotting results or re-evaluating results.

In `.cdsenv`:

```
adexl.plotting modelFilesAre cyclic "Omitted"
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.plotting" "modelFilesAre" 'cyclic "Omitted")
```

Valid Values:

`ReplacedWithCornerName` Replaces the model file paths with corner names.

`ShownInFull` Displays the actual model file path.

`IndividualModels` Displays each model file as if it was a corner parameter. This is usually used when you use the same model file across corners, but vary only the sections.

	`Shortened`	Displays shortened model file paths in the file-name:sectionName format.
	`Omitted`	Does not display model file paths in plots.
Default Value:	`Shortened`	

plotScalarExpressions

Controls whether expressions that evaluate to scalar values are automatically plotted after the simulation run is complete.

In `.cdsenv`:

```
adexl.plotting plotScalarExpressions boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "plotScalarExpressions" 'boolean t )
```

Valid Values:

`t`	Enables automatic plotting of expressions that evaluate to scalar values after the simulation run is complete.
`nil`	Disables automatic plotting of expressions that evaluate to scalar values after the simulation run is complete.

Default Value: `t`

plotScalarsAsLine

Controls whether expressions that evaluate to scalar values are automatically plotted with dots or lines after the simulation run is complete. For more information, see Plotting Dependent Expressions.

In `.cdsenv`:

```
adexl.plotting plotScalarsAsLine boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "plotScalarsAsLine" 'boolean t )
```

Valid Values:

`t`	Enables automatic plotting of dependent variables in expressions to lines that evaluate to scalar values after the simulation run is complete.
`nil`	Disables automatic plotting of expressions that evaluate to scalar values, with lines, after the simulation run is complete. These expressions may not contain dependent variables and may plot to lines or dots. **Note:** String parameters always plot as dots.

Default Value: `t`

plotSignals

Controls whether signals are automatically plotted after the simulation run is complete.

In `.cdsenv`:

```
adexl.plotting plotSignals boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "plotSignals" 'boolean t )
```

Valid Values:

`t`	Enables automatic plotting of signals after the simulation run is complete.
`nil`	Disables automatic plotting of signals after the simulation run is complete.

Default Value: `t`

plotType

Specifies the default plotting type to be used for all tests. For more information, see Setting Default Plotting Options for All Tests.

In `.cdsenv`:

```
adexl.plotting plotType cyclic "Auto"
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "plotType" 'cyclic "Auto" )
```

Valid Values:

`Auto`	Automatically plots outputs after the simulation run is complete. For every subsequent simulation run, a new graph replaces the existing graph.
`Refresh`	Automatically plots outputs after the simulation run is complete, but refreshes the existing graph in the same window.
`None`	Disables automatic plotting of results after the simulation run.

Default Value: `Auto`

plotWaveExpressions

Controls whether expressions that evaluate to waveforms are automatically plotted after the simulation run is complete.

In `.cdsenv`:

```
adexl.plotting plotWaveExpressions boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "plotWaveExpressions" 'boolean t )
```

Valid Values:

`t`	Enables automatic plotting of expressions that evaluate to waveforms after the simulation run is complete.
`nil`	Disables automatic plotting of expressions that evaluate to waveforms after the simulation run is complete.

Default Value: `t`

resultsCacheSize

By default, ADE XL saves the latest four plotting results from the root results directory to a dedicated cache. Caching the results improves performance. The resultsCacheSize variable defines the number of the results to be saved to this dedicated cache. Results are stacked based on the First In First Out (FIFO) strategy, hence the most recently opened result is cached. To disable dedicated caching, set this variable to zero or a negative integer.

In a scenario where resultsCacheSize is set to a high value, it will cache more directories resulting in improved performance, but due to this caching, the memory consumption will be higher too.

In `.cdsenv`:

```
adexl.plotting resultsCacheSize int 4
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "resultsCacheSize" 'int 4 )
```

Valid Values:

Any integer	If set to a value less than or equal to zero, no cache is maintained.

Default Value: `4`

showHistogramDensity

Specifies the default value for the *Density Estimator* annotation option on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting showHistogramDensity boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "showHistogramDensity" 'boolean nil )
```

Valid Values:

`t`	Plots a curve on the histogram that estimates the distribution concentration.
`nil`	Does not plot the curve.

Default Value: `t`

showHistogramDeviation

Specifies the default value for the *Std Dev Lines* annotation option on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting showHistogramDeviation boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "showHistogramDeviation" 'boolean nil )
```

Valid Values:

`t`	Shows the standard deviation lines in the graph indicating the mean, mean – standard deviation, and mean + standard deviation values.
`nil`	Does not show the standard deviation lines with the histogram.

Default Value: `t`

showHistogramPoints

Controls the display of data points on histograms to enable cross-selection between ADE XL results table and the histogram plotted in the Virtuoso Visualization and Analysis XL window.

By default, the histogram data points are visible and the bars are filled with a transparent or alpha color to make the points clearly visible. You can select a histogram point to cross-select the corresponding result in the ADE XL Results tab. Set this variable to `nil` to disable cross-selection from histogram points. In this case, the bars are filled with solid color.

Note: You can also show or hide the data points on histograms using the Trace Properties form or by selecting or clearing the *Symbols On* command in the context-sensitive menu of histograms.

In `.cdsenv`:

```
adexl.plotting showHistogramPoints boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "showHistogramPoints" 'boolean nil )
```

Valid Values:

`t`	Shows data points on histograms. The histogram bars are filled with a transparent or alpha color to make the data points clearly visible. You can change the style of data points by using the *Symbols* command on the context-sensitive menu of histogram.
`nil`	Hides data points on histograms. Histogram bars are filled with solid colors and points are not visible.

Default Value: `t`

Also see: Cross-Probing Results from Histogram Plots

showHistogramPercentMarkers

Specifies the default value for the *% Markers* annotation option on the Plot Histogram form.

In `.cdsenv`:

```
adexl.plotting showHistogramPercentMarkers boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal( "adexl.plotting" "showHistogramPercentMarkers" 'boolean
     nil )
```

Valid Values:

	`t`	Shows vertical marker lines in the histogram at percentages corresponding to 1, 2, and 3 standard deviations of a normal distribution If the distribution of the output is normal, these lines will be in the same position as the sigma lines. You can display the marker lines to get an idea about the distribution of points.
	`nil`	Does not show the vertical marker lines.
Default Value:	`nil`	

asimenv.plotting

- specMarkers on page 189
- useQPDataToCreateDataSheet on page 189

specMarkers

Controls whether spec markers should be displayed in the graphs plotted after simulation.

In `.cdsenv`:

```
asimenv.plotting specMarkers boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal( "asimenv.plotting" "specMarkers" 'boolean nil )
```

Valid Values:

`t`	Displays spec markers in the graphs. When this variable is set to `t`, the *Spec Markers* graph annotations option on the Printing/Plotting Options form is enabled.
`nil`	Does not display spec markers in the graphs.

Default Value: `nil`

useQPDataToCreateDataSheet

Determines whether quick plot data is to be used for printing waveforms in the datasheet.

The default value for this variable is `t`, which means quick plot data is used by default for printing waveforms in the datasheet.

In `.cdsenv`:

```
asimenv.plotting useQPDataToCreateDataSheet boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("asimenv.plotting" "useQPDataToCreateDataSheet" 'boolean
    t)
```

Valid Values:

	`t`	Quick plot data is used for printing waveforms in the datasheet. In case quick plot data does not exist and this variable is set to `t`, quick plot data will be generated first.
	`nil`	Full waveform data is used and the *Use Quickplot data* option is disabled on the Create Datasheet form.
Default Value:	`t`	

Environment Variables for Advanced Run Modes

You can set the following environment variables in your `.cdsenv` or `.cdsinit` files to customize the settings for simulations or results:

- ignoreFailedPointsInWCCRun
- digitsToShowForYieldInPercentage
- sortVariablesOpt
- stopManualTuningOnSessionExit
- useDoubleSidedSigma
- toleranceComparionRatiorForRSM
- useOptInWCD
- yieldProbability
- WCCEnableNewlyCreatedCorners

ignoreFailedPointsInWCCRun

Controls whether to filter failed points during the worst case corners run. While searching Worst Case Corner for a specification, by default a corner is created for a failed point if it exists. If you set this variable, the failed points are ignored while creating corner.

In `.cdsenv`:

```
"adexl.simulation" "ignoreFailedPointsInWCCRun" 'boolean t
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.simulation" "ignoreFailedPointsInWCCRun" 'boolean
     t)
```

Valid Values:

`t`	When set to `t`, the failed points are filtered out.
`nil` (default value)	When set to `nil`, the failed points are not filtered out.

digitsToShowForYieldInPercentage

Specifies the number of digits to be displayed for values in the *Yield In Percentage* column on the Results tab for High Yield Estimation run.

In `.cdsenv`:

```
adexl.gui digitsToShowForYieldInPercentage int 6
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "digitsToShowForYieldInPercentage" 'int t)
```

Valid Values:

A positive integer value

Default Value: `10`

sortVariablesOpt

Specifies if the variables and parameters should be sorted before generating random samples for an optimization run. By default, the variables are not sorted before the run is started. However, you can sort them by setting this variable to `t` so as to ensure that the result of different optimization runs is same irrespective of the order of variables and parameters.

In `.cdsenv`:

```
adexl.algorithm sortVariablesOpt boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.algorithm" "sortVariablesOpt" 'boolean t)
```

Valid Values:

`t`	Sorts the variables and parameters before generating random samples.
`nil`	Does not sort the variables and parameters before generating random samples. This is the default value.

stopManualTuningOnSessionExit

Specifies if the Manual Tuning run mode should be stopped when ADE XL GUI is closed while that run is in progress.

In `.cdsenv`:

```
adexl.simulation stopManualTuningOnSessionExit boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.simulation" "stopManualTuningOnSessionExit"
     'boolean t)
```

Valid Values:

`t`

Stops the currently running Manual Tuning run when the ADE XL GUI is closed. In the next session, the Run Simulation button is green in color and you can either start a new Manual Tuning run or submit new points in the previous Manual Tuning run.

`nil`

Does not stop the currently running Manual Tuning run when the ADE XL GUI is closed. When you open a new ADE XL session, the Run Simulation button is yellow in color and you can continue with the same Manual Tuning run that was running earlier.

This is the default value.

useDoubleSidedSigma

Specifies whether the K-sigma statistical corner is single-sided or double-sided. You can set this variable while creating K-sigma statistical corner from the following methods:

- ❑ K-sigma corner from Monte Carlo
- ❑ K-sigma corner from High Yield Estimation

This variable indicates how the yield number is presented in sigma. When it is set to `t`, K-sigma means that yield is represented as probability integration from `-K` sigma to `+K` sigma in Gaussian distribution.

When set to `nil`, K-sigma means that yield is represented as probability integration from `-infinity` to `+K` sigma in Gaussian distribution.

The setting of this variable does not impact the number of statistical corners created, instead, it impacts the High Yield Estimation results report (number of sigma), the High Yield statistical corner, the fast K-sigma results report, and the fast K-sigma statistical corner. To create fast K-sigma statistical corners, select *K-Sigma Corners* autostop method in the Monte Carlo run options and provide the target sigma value. The sigma information is displayed in the Monte Carlo run log.

In `.cdsenv`:

```
adexl.algorithm useDoubleSidedSigma boolean nil nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.algorithm" "useDoubleSidedSigma" 'boolean nil)
```

Valid Values:

`t` When this variable is set to `t`, the double-sided K-sigma statistical corner is created.

`nil` When this variable is set to `nil`, the single-sided K-sigma statistical corner is created.

This is the default value.

toleranceComparionRatiorForRSM

Ignores the RSM model coefficent if its ratio to the maximum coefficient value is less than the specified limit.

In `.cdsenv`:

```
adexl.gui toleranceComparionRatiorForRSM float 1e-6
```

In `.cdsinit` or the CIW:

```
envSetVal(“adexl.gui" "toleranceComparionRatiorForRSM" 'float 1e-6)
```

Valid Values:

Any floating point number

Default: `1e-6`

useOptInWCD

Controls whether to use the optimization search option for the Worst Case Distance method in the High Yield Estimation run mode.

In `.cdsenv`:

```
adexl.algorithm useOptInWCD boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.algorithm" "useOptInWCD" 'boolean nil)
```

Valid Values:

`t`	When this variable is set to `t`, the optimization search option is used.
`nil` (default)	When this variable is set to `nil`, the optiomization is not used.

yieldProbability

Defines the significance level in percentage. Probability values closer to 100% require more simulations before the yield estimate is determined to be lower or higher than the target. Smaller probability values require less simulations before the auto-stop is triggered.

In `.cdsenv`:

```
adexl.monte yieldProbability float 90
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.monte" "yieldProbability" 'float 90)
```

Valid Values:

Any value between 0 and 100

Default: `95.0`

WCCEnableNewlyCreatedCorners

Specifies whether to enable or disable the corners created after the worst case corner simulation run.

In `.cdsenv`:

```
adexl.gui WCCEnableNewlyCreatedCorners boolean nil
```

In `.cdsinit` or the CIW:

```
envSetVal("adexl.gui" "WCCEnableNewlyCreatedCorners" 'boolean nil)
```

Valid Values:

`t`	When this variable is set to `t`, the corners are enabled.
`nil` (default)	When this variable is set to `nil`, the corners are disabled.