Virtuoso ADE Environment Variables Reference - Part I

Product Version ICADVM20.1
October 2020

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

Contents

Preface . . . 21
Scope . . . 21
Licensing Requirements . . . 22
Related Documentation . . . 22
What's New and KPNS . . . 22
Installation, Environment, and Infrastructure . . . 22
Technology Information . . . 22
Virtuoso Tools . . . 23
Additional Learning Resources . . . 23
Video Library . . . 23
Virtuoso Videos Book . . . 24
Rapid Adoption Kits . . . 24
Help and Support Facilities . . . 24
Customer Support . . . 25
Feedback about Documentation . . . 25
Typographic and Syntax Conventions . . . 26

A
Environment Variables . . . 27
ADE Simulation Environment . . . 28
adelExitQuery . . . 28
adeMaestroCheckoutOrder . . . 28
adePanicStatePath . . . 29
allowAdePanicStateSaving . . . 30
allowInvalidObjectSelection . . . 30
appendLibNameToProjectDir . . . 31
artistPlottingMode . . . 31
waveformReaderThreadCount . . . 31
autoPlot . . . 32
awvResizeWindow . . . 32

browserCenterMode . . . 33
createCDFdata . . . 33
createCDFtermOrder . . . 34
createComponentName . . . 34
createNamePrefix . . . 35
createNetlistProc . . . 35
createOceanScriptBeforeSimulation . . . 36
nlcustomizedTermOrderList . . . 36
defaultHierSave . . . 37
defaultTools . . . 37
designEditMode . . . 38
designName . . . 38
designVarSetting . . . 38
digitalHost . . . 39
digitalHostMode . . . 39
digits . . . 40
directPlotPlottingMode . . . 40
displayPcellNameCustomFormat . . . 41
displayPointNetlistTimeStamp . . . 41
doNotOverwriteVarValuesWhileCopyingFromCV . . . 42
drlBufferMemory . . . 42
findVariablesSkip . . . 43
host . . . 44
hostMode . . . 44
icons . . . 45
ignoreDesignCheckout . . . 45
ignoreSchModified . . . 45
includeAsBundle . . . 46
innerSweepSwappableSizeLimit . . . 46
immediatePlot . . . 47
immediatePrint . . . 47
useHierarchicalSelection . . . 48
inductorDevicesNames . . . 48
keepShellVarsInModelLibPath . . . 48
netlistAccess . . . 49
includeSimLogInOCEAN . . . 50

numberOfSavedRuns . . . 50
obsoleteWarnings . . . 51
oceanScriptFile . . . 51
outputsImportExportVersion . . . 52
paraplotUpdateSimulatorLog . . . 52
pcellNameGenerationStyle . . . 53
pcellNameCustomFormat . . . 53
postSaveOceanScript . . . 54
preferTermOrderOverPinOrder . . . 54
preSaveOceanScript . . . 55
printCommentChar . . . 56
printInlines . . . 56
printNotation . . . 56
projectDir . . . 57
remoteDir . . . 57
resizeMode . . . 58
retainDesignVarNotation . . . 58
retainStateSettings . . . 59
saveAsCellview . . . 60
saveDefaultsToOCEAN . . . 61
saveDir . . . 61
saveStateWhenLibMarkedForDeletion . . . 62
saveQuery . . . 62
scalarOutputs . . . 63
schematicBased . . . 63
sevResolveSymLinks . . . 63
showConvertNotifyDialog . . . 64
showWhatsNew . . . 64
simulationDate . . . 65
simulator . . . 65
skipModelSectionsRetrieval . . . 66
smartViewDSPFDirectory . . . 67
stateName . . . 68
stateOverWriteSkipList . . . 68
strictCDFparseAsNumber . . . 69
temperature . . . 69

toolList . . . 69
tryNFSSync . . . 70
tryNFSSyncLocal . . . 70
updateCDFtermOrder . . . 71
updateCDFtermOrder . . . 71
useDisplayDrf . . . 72
useNamePrefix . . . 73
variables . . . 73
vaTemplateScript . . . 73
windowBased . . . 75
x . . . 75
x . . . 75
y . . . 76
y . . . 76
ADE XL . . . 78
defaultLibName . . . 78
showMenu . . . 78
showOpenViewDialog . . . 78
viewNamePrefix . . . 79
AMS . . . 80
ac_severity . . . 80
AMS_IGNORE_IGNORE . . . 80
assert_severity_default . . . 80
dc_severity . . . 81
dcOp_severity . . . 81
disallowRedefinition . . . 82
dresolution . . . 82
enablePkgImport . . . 83
fglobalSignals . . . 83
hnlVerilogIgnoreTermNameList . . . 85
ignorePrintNettype . . . 85
lTnetlistMaxWarn . . . 86
netlistBlackboxWithExtHdl . . . 86
netlistNoWarn . . . 87
print_control_vars . . . 87
shortInstanceList . . . 88

simOutputFormat . . . 89
tran_severity . . . 89
upgradeMsgSevError . . . 90
upgradeMsgSevWarn . . . 90
useEffectiveCDF . . . 91
ams.envOpts . . . 92
Calculator . . . 93
dstack . . . 93
eval . . . 93
mode . . . 94
uimode . . . 94
Distributed Processing . . . 95
autoJobSubmit . . . 95
block . . . 95
copyMode . . . 95
copyModeDir . . . 96
copyNetlist . . . 96
daysBeforeExpire . . . 97
drmsCommandList . . . 97
emailNotify . . . 98
expDay . . . 98
expTime . . . 99
externalServer . . . 99
hostName . . . 100
jobArgsInOceanScript . . . 100
loginShell . . . 101
logsInEmail . . . 101
mailAllLogs . . . 101
mailTo . . . 102
numOfTasks . . . 102
puttogetherqueue . . . 103
queueName . . . 103
setupFunction . . . 103
showMessages . . . 106
startDay . . . 106
startTime . . . 107

stateFile . . . 107
timeLimit . . . 108
HspiceD . . . 109
checkOutLicenseDuringNetlistAndRun . . . 109
hspiceMaxLineLength . . . 109
hspiceSoftLineLength . . . 110
mapGndNetToZero . . . 110
netlistModelFileFirst . . . 111
setTopLevelAsSubckt . . . 111
userCmdLineOption . . . 112
Analysis . . . 113
relxpert_gradual_aging . . . 113
spectre_analysis_ . . . 113
Spectre . . . 114
AC Match Analysis Environment Variables . . . 114
ac_severity . . . 117
analysisOrder . . . 117
apsplus . . . 118
assert_severity_default . . . 118
autoDisplay . . . 119
autoDisplayBBox . . . 119
checkpoint . . . 120
controlMode . . . 120
currents . . . 121
dc_severity . . . 121
dcOp_severity . . . 122
definitionFiles . . . 122
displayToolTip . . . 123
dochecklimit . . . 123
dspfFile . . . 124
emLayerMap . . . 124
emirEnable . . . 125
emirLCVDisable . . . 125
emirSumList . . . 125
emTechFile . . . 126
enableArclength . . . 126

fastViewOption . . . 127
finalTimeOp . . . 127
firstRun . . . 128
ignorePortOrderMismatch . . . 128
infoOptions . . . 129
liclog . . . 131
licQueueSleep . . . 132
licQueueTimeOut . . . 132
licQueueToken . . . 133
includePath . . . 133
modelFiles . . . 134
modelParamInfo . . . 135
nestlvl . . . 135
netlistBBox . . . 136
nport_default_interp . . . 136
nportirfiledir . . . 137
outputParamInfo . . . 137
osc_accuracy . . . 138
osc_version . . . 139
paramRangeCheckFile . . . 140
preserveSubcktTermNamesByOrder . . . 141
printComments . . . 141
pwr . . . 142
recover . . . 142
resetSpectreApsSharedOptions . . . 143
save . . . 143
saveahdlvars . . . 144
setEngNotation . . . 144
simExecName . . . 145
simOutputFormat . . . 145
stimulusFile . . . 146
stopViewList . . . 149
subcktprobelvl . . . 149
switchViewList . . . 150
tran_severity . . . 150
uniMode . . . 151

useprobes . . . 151
OSS . . . 152
Generating Customized Cellnames . . . 152
Ultrasim . . . 155
fastViewOptionl . . . 155
useOtherOutputFormat . . . 155
wf_format . . . 156

B
Environment Variables for Spectre Simulator Options Form . . . 157

spectre.opts . . . 158
Simulator Options - Main . . . 161
generalnoiseinst . . . 161
generalnoiseinstonoff . . . 161
highvoltage . . . 161
iabstol . . . 162
multithread . . . 162
noiseOffType . . . 163
noiseOnType . . . 163
nthreads . . . 164
reltol . . . 164
residualtol . . . 164
temp . . . 165
tempeffects . . . 165
tnom . . . 166
vabstol . . . 166
Simulator Options - Algorithm . . . 167
convdbg . . . 167
dc_pivot_check . . . 167
dptran_gmethod . . . 167
gmethod . . . 168
gmin . . . 168
gmin_check . . . 169
gmindc . . . 169

homotopy . . . 170
icpriority . . . 170
ilimit . . . 171
nonconv_topnum . . . 171
pivabs . . . 171
pivotdc . . . 172
pivrel . . . 172
preorder . . . 173
rabsclamp . . . 173
rabsshort . . . 174
rebuild_matrix . . . 174
rforce . . . 175
rthresh . . . 175
try_fast_op . . . 176
Simulator Options - Component . . . 177
approx . . . 177
auto_minductor . . . 177
vth_vdsmin . . . 177
ivthl . . . 178
ivthn . . . 178
ivthp . . . 179
ivthw . . . 179
imacromodels . . . 179
maxrsd . . . 180
nport_default_passivity . . . 180
nportcompress . . . 181
nportcompressfiledir . . . 181
nportirfiledir . . . 182
nportirreuse . . . 182
nportunusedportgmin . . . 183
nportunusedportrmin . . . 183
minr . . . 184
scale . . . 184
scalefactor . . . 185
scalem . . . 185
tmevthmod . . . 186

vthmod . . . 186
Simulator Options - Check . . . 187
checklimitdest . . . 187
checklimitfile . . . 187
checklimitskipfile . . . 187
checklimitskipsubs . . . 188
diagnose . . . 188
iccheck . . . 189
gnshorts . . . 189
redefinedparams . . . 190
opptcheck . . . 190
topcheck . . . 191
Simulator Options - Annotation . . . 192
audit . . . 192
cols . . . 192
colslog . . . 192
debug . . . 193
digits . . . 193
error . . . 194
info . . . 194
inventory . . . 194
maxnotes . . . 195
maxnotestologfile . . . 195
maxwarns . . . 196
maxwarnstologfile . . . 196
notation . . . 197
note . . . 197
narrate . . . 198
printstep . . . 198
simstat . . . 198
title . . . 199
warn . . . 199
Simulator Options - Miscellaneous . . . 201
additionalArgs . . . 201
ahdllint . . . 201
ahdllint_maxwarn . . . 201

ahdllint_on . . . 202
flow . . . 202
quantities . . . 203
sensbinparam . . . 203
sensfile . . . 204
sensfileonly . . . 204
sensformat . . . 204
senstype . . . 205
value1 . . . 205
. . . 206

C
Variables for ams.env Files . . . 207
List of ams.env Variables . . . 208
Detailed Descriptions of ams.env Variables . . . 214
aliasInstFormat . . . 215
allowDeviantBuses . . . 216
allowIllegalIdentifiers . . . 218
allowNameCollisions . . . 220
allowSparseBuses . . . 222
allowUndefParams . . . 224
amsCompMode . . . 225
amsDefinitionViews . . . 226
amsEligibleViewTypes . . . 228
amsExcludeParams . . . 229
amsExpScalingFactor . . . 230
amsLSB_MSB . . . 232
amsMaxErrors . . . 233
amsScalarInstances . . . 234
amsVerbose . . . 235
analogControlFile . . . 236
artistStateDirectory . . . 237
bindCdsAliasLib . . . 238
bindCdsAliasView . . . 239
cdsGlobalsLib . . . 240

cdsGlobalsView . . . 241
checkAndNetlist . . . 242
checkOnly . . . 243
checktasks . . . 244
compileAsAMS . . . 245
compileExcludeLibs . . . 246
compileMode . . . 247
confirmADEStateImport . . . 249
connectRulesCell . . . 250
connectRulesCell2 . . . 251
connectRulesLib . . . 252
connectRulesView . . . 253
defaultRunDir . . . 254
detailedDisciplineRes . . . 255
discipline . . . 256
errOutInconsistentMasters . . . 257
excludeViewNames . . . 258
hdlVarFile . . . 259
headerText . . . 260
ieee1364 . . . 261
ifdefLanguageExtensions . . . 262
ignoreIllegalCDFParams . . . 263
implicitTmpDir . . . 264
incdir . . . 265
includeFiles . . . 266
includeInstCdfParams . . . 267
initFile . . . 268
instClashFormat . . . 269
iterInstExpFormat . . . 270
language . . . 271
lexpragma . . . 272
logFileAction . . . 274
logFileName . . . 275
macro . . . 277
markcelldefines . . . 278
maxErrors . . . 279

messages . . . 280
modifyParamScope . . . 281
ncelabAccess . . . 285
ncelabAfile . . . 286
ncelabAnnoSimtime . . . 287
ncelabArguments . . . 288
ncelabCoverage . . . 289
ncelabDelayMode . . . 290
ncelabDelayType through ncelabMessages . . . 292
ncelabMixEsc . . . 293
ncelabModelFilePaths . . . 294
ncelabNeverwarn through ncelabVipdelay . . . 295
ncsimArguments . . . 298
ncsimEpulseNoMsg through ncsimExtassertmsg . . . 299
ncsimGUI . . . 300
ncsimLoadvpi through ncsimStatus . . . 301
ncsimTcl . . . 302
ncsimUnbuffered through ncsimUseAddArgs . . . 303
ncvhdlArguments . . . 304
ncvlogArguments . . . 305
ncvlogUseAddArgs . . . 306
netClashFormat . . . 307
netlistAfterCdfChange . . . 308
netlistMode . . . 309
netlistToRunDir . . . 310
netlistUDFAsMacro . . . 313
neverwarn . . . 315
noline . . . 316
nomempack . . . 317
nopragmawarn . . . 318
nostdout . . . 319
nowarn . . . 320
paramDefVals . . . 321
paramGlobalDefVal . . . 322
pragma . . . 323
useRunDirNetlistsOnly . . . 324

processViewNames . . . 326
prohibitCompile . . . 327
runNcelab . . . 328
runNcsim . . . 329
scaddlglblopts . . . 330
scaddltranopts . . . 331
scale . . . 332
scalem . . . 333
scannotate . . . 334
scapprox . . . 335
scaudit . . . 336
sccheckstmt . . . 337
sccmin . . . 338
sccompatible . . . 339
scdebug . . . 340
scdiagnose . . . 341
scdigits . . . 342
scerror . . . 343
scerrpreset . . . 344
scfastbreak . . . 345
scglobalminr . . . 346
scgmin . . . 347
scgmincheck . . . 348
schomotopy . . . 349
sciabstol . . . 350
scic . . . 351
scicstmt . . . 352
scignshorts . . . 353
scinfo . . . 354
scinventory . . . 355
sclimit . . . 356
sclteratio . . . 357
scmacromod . . . 358
scmaxiters . . . 359
scmaxnotes . . . 360
scmaxnotestologfile . . . 361

scmaxrsd . . . 362
scmaxstep . . . 363
scmaxwarn . . . 364
scmaxwarntologfile . . . 365
scmethod . . . 366
scmodelevaltype . . . 367
scmosvres . . . 368
scnarrate . . . 369
scnotation . . . 370
scnote . . . 371
scopptcheck . . . 372
scpivabs . . . 373
scpivotdc . . . 374
scpivrel . . . 375
scquantities . . . 376
screadic . . . 377
screadns . . . 378
screlref . . . 379
screltol . . . 380
scrforce . . . 381
scscale . . . 382
scscalem . . . 383
scscfincfile . . . 384
scscftimestamp . . . 385
scscfusefileflag . . . 386
scskipcount . . . 387
scskipdc . . . 388
scskipstart . . . 389
scskipstop . . . 390
scspeed . . . 391
scspscflag . . . 393
scstats . . . 394
scstep . . . 395
scstop . . . 396
scstrobedelay . . . 397
scstrobeperiod . . . 398

sctemp . . . 399
sctempeffects . . . 400
sctitle . . . 401
sctnom . . . 402
sctopcheck . . . 403
scusemodeleval . . . 404
scvabstol . . . 405
scwarn . . . 406
scwrite . . . 407
scwritefinal . . . 408
simcompat . . . 409
simRunDirLoc . . . 410
simVisScriptFile . . . 411
status . . . 412
templateFile . . . 413
templateScript . . . 414
timescale . . . 415
update . . . 416
use5xForVHDL . . . 417
useDefparam . . . 418
useEffectiveCDF . . . 420
useNcelabNowarn . . . 421
useNcelabSdfCmdFile . . . 422
useNcsimNowarn . . . 423
useNowarn . . . 424
useProcessViewNamesOnly . . . 425
useScaddlglblopts . . . 426
useScaddltranopts . . . 427
useScic . . . 428
useScreadic . . . 429
useScreadns . . . 430
useScscfincfile . . . 431
useScwrite . . . 432
useScwritefinal . . . 433
useSimVisScriptFile . . . 434
usimAbstoli through usimWFTres . . . 435

verboseUpdate 438
vlogGroundSigs 439
vloglinedebug 440
vlogSupply0Sigs 441
vlogSupply1Sigs 442
wfDefaultDatabase 443
wfDefInstCSaveAll 444
wfDefInstCSaveLvl 445
wfDefInstSaveCurrents 446
wfDefInstSaveVoltages 447
wfDefInstVSaveAll 448
wfDefInstVSaveLvl 449
wfDefInstVSaveObjects 450
wfFilter 451
wfFilterSpec 452

Preface

Virtuoso Analog Design Environment L allows you to set up and run analog simulations using different simulators. After running a simulation, you can view and analyze simulation results.

This manual describes how to use the Virtuoso® Analog Design Environment to simulate analog designs. The information presented in this manual is intended for integrated circuit designers and assumes that you are familiar with analog design and simulation.

The preface discusses 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 and methodologies (for example, ICADVM20.1) releases.

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

Licensing Requirements

The license number required for ADE L is `Analog_Design_Environment_L`. One ADE L feature license is required for one User, Display and Host (UHD) session of ADE L.

You can also set ADE L to be your default application by selecting *File - Set Default Application* and ensuring that ADE L is set as the default for the listed scenario options. For more information on setting a default application see, Virtuoso Design Environment User Guide.

For more information on licensing and related information, see:

- Obtaining Licences in Virtuoso Design Environment User Guide
- Cadence Workspaces in Virtuoso Design Environment User Guide
- *Virtuoso Software Licensing and Configuration User Guide*

Related Documentation

What's New and KPNS

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

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Virtuoso Design Environment SKILL Reference*
- *Cadence Application Infrastructure User Guide*
- *Virtuoso® Spectre® Circuit Simulator Reference*

Technology Information

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

- *Virtuoso Technology Data SKILL Reference*.

Virtuoso Tools

- *Virtuoso Analog Design Environment XL User Guide*
- *Virtuoso Analog Design Environment GXL User Guide*
- *Virtuoso Visualization and Analysis XL User Guide*
- *Virtuoso Analog Distributed Processing Option User Guide*
- *Virtuoso Parasitic Estimation and Analysis User Guide*
- *Virtuoso Schematic Editor User Guide*
- *Spectre Circuit Simulator Reference*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis User Guide*
- *Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide*
- *Virtuoso Design Environment Adoption Guide* **(IC6.1.8 Only)**
- *Virtuoso UltraSim Simulator 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 L:

- Virtuoso Schematic Editor
- Virtuoso Analog Design Environment
- 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

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 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 public environment variables that control the characteristics of the Analog Design Environment (ADE). You can customize the operation and behavior of Analog Design Environment products by changing the value of a particular environment variable.

This appendix lists environment variables belonging to the following products:

- ADE Simulation Environment
- ADE XL
- AMS
- ams.envOpts
- Calculator
- Distributed Processing
- HspiceD
- Analysis
- Spectre
- Ultrasim

The appendix also provides you a list of deprecated environment variables.

ADE Simulation Environment

adelExitQuery

While closing the ADE L session, if this variable and the saveQuery are set to `t`, a dialog box is displayed asking whether you want to save the state of your environment if it has not been saved.

If the saveQuery variable is set to `nil`, a dialog box is displayed to confirm whether you want to close the ADE L session.

If the saveQuery and adelExitQuery are set to `nil`, the ADE L session closes without showing any dialog box.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "adelExitQuery" 'boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	--

adeMaestroCheckoutOrder

If you do not have the ADE L, XL or GXL licenses, but have either of the Virtuoso ADE Explorer or Virtuoso ADE Assembler licenses, then these licenses can also be used to run ADE L and XL. You can set this variable to one of the following values:

- `ADE`: Only ADE L, XL or GXL licenses are searched and used. In this case, ADE Explorer or ADE Assembler licenses will never be used to run ADE L or XL.
- `ADE_Maestro` (default): ADE L, XL or GXL licenses are searched first. If not available, ADE Explorer and ADE Assembler licenses are searched and used when found.
- `Maestro`: Only ADE Explorer and ADE Assembler license are searched and used. In this case, ADE Explorer or ADE Assembler licenses are always used to run ADE L and XL.

- `Maestro_ADE`: ADE Explorer and ADE Assembler licenses are searched and used first. If not available, ADE L or XL licenses are searched and used.
  - The checkout order for ADE L is: ADE Explorer – ADE Assembler – ADE L – ADE XL – ADE GXL
  - The checkout order for ADE XL is: ADE Assembler – ADE XL – ADE GXL

To set this variable in the .cdsinit file or CIW:

```
envSetVal("asimenv" "adeMaestroCheckoutOrder" 'cyclic "ADE_maestro")
```
or
```
envSetVal("license" "adeMaestroCheckoutOrder" 'string "Classic,
Maestro")
```

Variable Type	cyclic
Default Value	`"ADE_maestro"`
Acceptable Values	`"ADE"`, `"ADE_Maestro"`, `"Maestro"`, `"Maestro_ADE"`

Important Points to note:

- The `adeMaestroCheckoutOrder` environment variable can be set using `asimenv` and `license` tool partitions. However, the `asimenv` tool partition will be removed in a future release. It is recommended that you use the `license` tool partition to set this variable.
- Note the difference between variable type and values for different partitions.

adePanicStatePath

In case of an unexpected failure in Virtuoso, ADE saves the current simulation settings in the directory named `.ADE_Panic_State` at the path specified using this variable. In case `adePanicStatePath` is not set by the user, the `.ADE_Panic_State` directory is saved in the simulation directory.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "adePanicStatePath" 'string "dir1")
```

Variable Type	string

Default Value	`""`
Acceptable Values	name of a directory.

allowAdePanicStateSaving

In case of an unexpected failure in Virtuoso, there are chances to lose unsaved simulation settings. This variable allows ADE to save the settings in the panic state of unexpected failure. By default, the variable is set to t and ADE saves the settings.

To unset this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "allowAdePanicStateSaving" 'boolean nil)
```

Variable Type	boolean
Default Value	`t`
Acceptable Values	t, nil

allowInvalidObjectSelection

If this variable is set to `t`, external signals (nets and terminals) specified in the outputs are included for netlisting. This is useful when simulation includes an external text netlist or DSPF files in which internal nets and terminals are not mapped by ADE.

By default, signals specified in the outputs are validated by ADE and if the signals are not found in the design hierarchy, they are disabled. So you cannot select them for plotting and saving; hence they are not included in the simulation results.

Note: Since signals that are not found in the design hierarchy are not validated by ADE, you must ensure that the signals are valid.

To set this variable in the .cdsenv file, use the call:

```
auCore.selection allowInvalidObjectSelection boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("auCore.selection" "allowInvalidObjectSelection" 'boolean t)
```

Variable Type	boolean
Default Value	`nil`

Acceptable Values	`t`, `nil`

appendLibNameToProjectDir

Lets you specify if the library name should be appended to the simulation directory.

To set this variable in the .cdsenv file, use the call:

```
asimenv.startup appendLibNameToProjectDir 'boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.startup" "appendLibNameToProjectDir" 'boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

artistPlottingMode

Plots the entire plot set in the specified mode.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.plotting" "artistPlottingMode" 'string "New Win")
```

Variable Type	string
Default Value	Replace
Acceptable Values	Append / Replace / New Win / New SubWin
GUI Equivalent	*Plotting Mode drop-down listbox on the ADE L window*

waveformReaderThreadCount

Specifies the number of threads that are computed by SRR while reading a parametric waveform data. The purpose of this variable is to achieve maximum throughput during the loading of parametric waveform.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "waveformReaderThreadCount" 'string
"auto")
```

Variable Type	String
Default Value	auto
Acceptable Values	Any integer value
GUI Equivalent	--

The default value, `auto`, indicates that number of threads are computed automatically. When you specify an integer value, it directs the SRR to start the specified number of threads. When you specify `0`, it disables the SRR threading while reading the parametric data.

autoPlot

Plots the entire plot set (including waveform expressions) automatically when each simulation run is finished.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "autoPlot" 'cyclic "Refresh")
```

Variable Type	cyclic
Default Value	Auto
Acceptable Values	Auto, None, Refresh
GUI Equivalent	*Plot After Simulation drop-down list box on the ADE L window.*

awvResizeWindow

```
awvResizeWindow( w_windowId  l_bBox )
     => t | nil
```

Resizes a window to the size of a bounding box. The bounding box is specified as a list of 2 x:y points, that represent the lower left and upper right corners of the box.

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t/nil`
Example	`awvResizeWindow( window(4) list(391:288 1134:922) )`

browserCenterMode

To keep the most recently expanded node in the results browser in the center of the window, set this variable to true. If you do not want the most recently expanded node to move automatically to the center of the window, you can turn off the centering mode by setting this variable to `nil`.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.misc" "browserCenterMode" 'boolean nil)`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

createCDFdata

If this variable is set to `t`, the fields *netlistProcedure*, *namePrefix*, and *componentName* in the CDF `simInfo` section are reset to their default values while generating the CDF information for the cellview. If this variable is set to `nil`, none of these fields are included while generating the CDF information.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("auCore.misc" "createCDFdata" 'boolean nil)`

To set this variable in the `.cdsenv` file, use the call:
`auCore.misc createCDFdata boolean t`

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

createCDFtermOrder

While creating a new cellview from an existing cellview, the terminal order of the new cellview is automatically generated and added to the CDF simulation information (simInfo) property `termOrder` in the CDF. To disable the auto-creation of terminal order for cellview-from-cellview creation, you can set this variable to `nil`.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("auCore.misc" "createCDFtermOrder" 'boolean nil)`

To set this variable in the `.cdsenv` file, use the call:
`auCore.misc createCDFtermOrder boolean nil`

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

createComponentName

Adds the specified string to the *componentName* field in the CDF `simInfo` section. You can use this variable to specify the component name.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("auCore.misc" "createComponentName" 'string "subcircuit")`

To set this variable in the `.cdsenv` file, use the call:
`auCore.misc createComponentName string "subcircuit"`

Variable Type	String

Default Value	`"subcircuit"`
Acceptable Values	Any string value.

createNamePrefix

Adds the information related to the name prefix to the *namePrefix* field in the CDF `simInfo` section.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("auCore.misc" "createNamePrefix" 'boolean t)
```

To set this variable in the `.cdsenv` file, use the call:
```
auCore.misc createNamePrefix boolean t
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

createNetlistProc

Adds the information related to the netlisting procedure to the *netlistProcedure* field in CDF `simInfo` section.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("auCore.misc" "createNetlistProc" 'boolean t)
```

To set this variable in the `.cdsenv` file, use the call:
```
auCore.misc createNetlistProc boolean t
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

createOceanScriptBeforeSimulation

Creates an OCEAN script before running simulation and saves it in the netlist directory of the cellview. This increases the overall simulation time, therefore, the variable is set to `nil` by default. If you want to create an OCEAN script for debugging purposes, set this variable to `t` in the `.cdsinit` file.

To set this variable in the `.cdsinit` file, use the call:
```
envSetVal("asimenv" "createOceanScriptBeforeSimulation" 'boolean t)
```

Variable Type	Boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

nlcustomizedTermOrderList

The `nlcustomizedTermOrderList` variable is used to customize the terminal order in netlist. The value of this variable is a list that represents the possible combinations for portorder and defines the preference order. The available terminal orders are: CDF simInfo termOrder (`cdfTermOrder`), schematic portorder (`masterPortOrder`), name of the custom function for fetching terminal order (`customFunc`), and alphanumeric order given by OSS (`defaultTermOrder`).

To set this variable in the .cdsinit file or CIW, use the call:
```
nlcustomizedTermOrderList = list("cdfTermOrder" "masterPortOrder"
"customFunc" "defaultTermOrder")
```

Variable Type	list
Default Value	`none`
Acceptable Values	`cdfTermOrder`, `masterPortOrder`, `customFunc`, `defaultTermOrder`
GUI Equivalent	--

This variable indicates the preferences order to be used for cdf simInfo termOrder, schematic port order, name of custom function for fetching terminal order, and alpha numeric order given by OSS.

When you set `nlcustomizedTermOrderList = list("cdfTermOrder" "masterPortOrder")`, the preference is given to cdfTermOrder. If it is not present, schematic port order is used. If both are `nil`, the default order is used.

When you set `nlcustomizedTermOrderList = list("defaultTermOrder")`, the default alphanumeric order is used.

defaultHierSave

Sets the default selected value of the voltage, current, or power for all the subcircuit instances in the *Save By Subckt Instances* pane of the simulation window. By default, the check boxes for only the voltage are selected for all the subcircuit instances.

To set this environment variable in the `.cdsenv` file, use the following call:
`asimenv defaultHierSave 'cyclic "current"`

To set this environment variable in the `.cdsinit` file or CIW, use the following call:
`envSetVal("asimenv" "defaultHierSave" 'cyclic "current")`

Variable Type	cyclic
Default Value	`voltage`
Acceptable Values	`voltage`, `current`, `power`, `all`, `none`
GUI Equivalent	None

defaultTools

Set this variable to the list of simulators that need to be selected by default in the *toolFilter* form.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("auCore.toolFilter" "defaultTools" 'string "spectre")`

Variable Type	string
Default Value	string "spectre auCdl auLvs"
Acceptable Values	spectre auCdl auLvs

designEditMode

This variable lets you choose the default open mode for your designs. If you select `t`, your designs are opened in edit mode. If you select `nil`, your designs are opened in read-only mode.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv" "designEditMode" 'boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Default Design Open Mode*

designName

If true, displays the design name in the graph window.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "designName" 'boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Design Name*

designVarSetting

Specifies a list of design variables, with their respective values, to be added to an ADE session at the time of session creation or initialization. The list is specified as a string with space-separated name and value pairs. The design variables created with this variable are added in the *Design Variables* pane.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv" "designVarSetting" 'string "var1 10 var2 30")`

In this example, whenever an ADE session is created, two design variables, `var1` and `var2`, are set with values `10` and `30`, respectively.

Variable Type	string
Default Value	nil
Acceptable Values	Space-separated list of design variables and their values, nil
GUI Equivalent	-

digitalHost

The variable digitalHost lets you specify a path to the host computer for a digital remote simulation. You must specify a complete path.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.startup" "digitalHostMode" 'string "cds11939")`

Variable Type	string
Default Value	""
Acceptable Values	name of any machine in the network
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Digital Host*

digitalHostMode

The variable digitalHostMode lets you choose default local or remote digital simulation.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.startup" "digitalHostMode" 'string "remote")`

Variable Type	string

Default Value	"local"
Acceptable Values	local, remote
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Digital Host Mode*

digits

Number of significant digits with which the contributors are printed.

Variable Type	int
Default Value	6
Acceptable Values	Any integer. The maximum limit is the limit of an integer.
GUI Equivalent	--

directPlotPlottingMode

Plots the entire plot set in the specified mode.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.plotting" "directPlotPlottingMode" 'string "New Win")
```

Variable Type	string
Default Value	Append
Acceptable Values	Append / Replace / New Win / New SubWin
GUI Equivalent	*Cyclic is on the DirectPlot Main form*

displayPcellNameCustomFormat

Displays the parameters in the netlist header, in comments associated with the Pcell name that is customized using the variable pcellNameGenerationStyle.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.netlist" "displayPcellNameCustomFormat" 'boolean
t)
```

Variable Type	boolean
Default Value	`t`
Acceptable Values	`t, nil`
GUI Equivalent	-

displayPointNetlistTimeStamp

Displays the time stamp of the point netlist generated. The netlist header contains the following information.

```
// Point Netlist Generated on: Oct 21 10:16:52 2019 // Generated for: spectre //
Design Netlist Generated on: Oct 20 11:26:11 2019
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.netlist" "displayPointNetlistTimeStamp" 'boolean
t)
```

Variable Type	boolean
Default Value	`t`
Acceptable Values	`t`– Displays the time stamp on the point netlist header `nil`– Hides the time stamp on the point netlist header
GUI Equivalent	--

doNotOverwriteVarValuesWhileCopyingFromCV

If this variable is set to a non-nil value, design variables are copied from cellview without overwriting the existing variables' values and only copies the new variables and their values.

By default, *Copy from Cellview* overwrites all existing variables' values.

To set this variable in the .cdsenv file, add the line
```
asimenv doNotOverwriteVarValuesWhileCopyingFromCV boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv" "doNotOverwriteVarValuesWhileCopyingFromCV"
'boolean t)
```

For default behavior, use the call:
```
envSetVal("asimenv" "doNotOverwriteVarValuesWhileCopyingFromCV"
'boolean nil)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

drlBufferMemory

Allocates a buffer memory of the specified size for Virtuoso.

In cases when Virtuoso reads huge simulation data or evaluates memory-intensive expressions, the process might consume all the available memory. When allocation of more memory fails, Virtuoso might terminate abnormally or it might behave inconsistently. In such cases, Virtuoso can use this buffer memory to make the process exit gracefully. By default, memory of size 10 million bytes is allocated. You may increase the size of this memory for better results.

Note: Memory allocated using the drlBufferMemory variable helps in exiting Virtuoso gracefully in case of memory crunch. However, it may not be successful in all such scenarios.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "drlBufferMemory" 'int 10000000)
```

Variable Type	int
Default Value	`10000000`
Acceptable Values	10 to 25% of the total RAM size

findVariablesSkip

This environment variable can be used to ignore the design variables in the extracted cellviews or in the parasitic instances, thus improving the netlisting performance of extracted views.

To ignore the design variables in the parasitic instances,

- Add the following call in the `.cdsenv` file:

  ```
  auCore.misc findVariablesSkip cyclic "parasitic"
  ```

- Use the following call in the `.cdsinit` file or CIW:

  ```
  envSetVal("auCore.misc" "findVariablesSkip" 'cyclic "parasitic")
  ```

Important

The netlisting traversal in maestro cellviews uses a new traversal code that handles large iterated instances and extracted views more efficiently. This code identifies a cellview as `extracted`, if the *extractionCreatedBy* property is present in the cellview, but ignores the corresponding value. An external extracted view must have this property to benefit from the traversal code changes. The cellviews that have *findVariablesSkip* property set to `parasitic` or `extracted` are ignored during pre-simulation checks and the netlisting of Checks/Asserts to support performance improvement.

To ignore the design variables in the extracted cellviews,

- Add the following call in the `.cdsenv` file:

  ```
  auCore.misc findVariablesSkip cyclic "extracted"
  ```

- Use the following call in the `.cdsinit` file or CIW:

```
envSetVal("auCore.misc" "findVariablesSkip" 'cyclic "extracted")
```

Variable Type	Cyclic
Default Value	`None`
Acceptable Values	`None`, `parasitic`, `extracted`
GUI Equivalent	–

host

Lets you specify a path to the host computer for remote simulation. You must specify a complete path.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.startup" "host" 'string "cds11938")`

Variable Type	String
Default Value	""
Acceptable Values	name of any machine in the network
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Host*

hostMode

Lets you specify a default local, remote or distributed simulation.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.startup" "hostMode" 'string "remote")`

Variable Type	String
Default Value	Local
Acceptable Values	Local, remote, distributed
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Host Mode*

icons

This variable places icons in the graph window.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.plotting" "icons" 'boolean "t")`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Allow Icons*

ignoreDesignCheckout

If this variable is set to `t`, then while loading an ADE L state which is saved as a cellview using the Loading State form, the cellview is not checked out and is opened in read-only mode. By default, when an ADE L state which is saved as a cellview is loaded using the Loading State form, the Auto Checkout form that can be used to check out the cellview appears.

To set this variable in the .cdsenv file, use the call:
`asimenv.loadstate ignoreDesignCheckout  boolean t`

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.loadstate" "ignoreDesignCheckout" 'boolean t)`

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

ignoreSchModified

Set this variable to *t*, the schematic will not be modified. When this variable is set, you need not do a check and save after making changes to the *toolFilter* form.

To set this variable in the `.cdsenv` file or CIW, use the call:
`envSetVal("auCore.toolFilter" "ignoreSchModified" 'boolean nil)`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil

includeAsBundle

This variable specifies the default value of the *Include as bundle* check box that is displayed in the Select bits from bus form. If set to `t`, the *Include as bundle* check box is selected by default when you open the form.

To set this variable in the .cdsenv file, use the call:
`auCore.selection includeAsBundle boolean t nil`

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("auCore.selection" "includeAsBundle" 'boolean t)`

Variable Type	boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

innerSweepSwappableSizeLimit

Specifies the limit for the number of simulation points up to which the `swapSweep` function can be successfully run.

Note: The `swapSweep` function swaps the X-axis value with the specified sweep variable. If the number of points are more than the limit specified by this variable, you need to input a particular X-axis point to be used for swapping.

To set this variable in the .cdsinit file or CIW:
`envSetVal("asimenv.plotting" "innerSweepSwappableSizeLimit" 'int 500)`

Variable Type	integer
Default Value	500
Acceptable Values	Any integer value greater than 0

immediatePlot

This variable refers to the commands located in the *Direct Plot* menu. If true, the plot is drawn after each node is selected. If nil, none of the plots are drawn until all the nodes have been selected. You can select more than one node and click the `Escape` key when finished, and all the selected nodes are plotted at the same time.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.plotting" "immediatePlot" 'boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Direct Plots Done After*

immediatePrint

This variable refers to the commands located in the Print menu. If true, the results are printed after each node is selected. If nil, none of the nodes is printed until all the nodes have been selected.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.plotting" "immediatePrint" 'boolean "t")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Print After*

useHierarchicalSelection

Enables you to use hierarchical selection for various ADE operation. When set to `nil`, the behavior does not change, which means selection in the navigator is ignored. When set to `t`, the selection in the navigator can be used in many places, including output to be plotted or saved in ADE L and ADE XL, as well as the places that include a *Select on Design* button, such as expressions and the calculator of ADE XL.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("auCore.selection" "useHierarchicalSelection" 'boolean"t")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	--

inductorDevicesNames

Enables you to specify the initial conditions for the inductor current. You can define the list of space-separated inductor devices in the variable values. For example, if the value of this variable is set to `ind`, it enables the selection of pins of devices with names—`ind`, `ind1`, `ind2`, `inductor`, and so on.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal(auCore.selection inductorDeviceNames 'string "ind")
```

Variable Type	string
Default Value	string "ind"
Acceptable Values	Any string value

keepShellVarsInModelLibPath

This variable is used to retain the shell variables used in all file paths in the Model Library Setup form and the Simulation Files Setup form.

When set to `nil`, the file path is changed to an absolute path by expanding all the shell variables used in it. When set to `t`, the shell variables used in the file path are retained.

For example, If the following shell variables are available:

PROJECT=/projects/lrd_pe/users/user1/myblock
PDK=pdk_v1
tool=spectre

These variables are used in a model file path, as shown below.

```
$PROJECT/$PDK/models/$tool/models.scs
```

In this example, if you browse up to the `models` directory and choose another directory, the variables *$PROJECT* and *$PDK* will be retained, but, *$tool* will be removed from the path.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.misc" "keepShellVarsInModelLibPath" 'boolean "t")
```

Variable Type	boolean
Default Value	`t`
Acceptable Values	`nil, t`
GUI Equivalent	None

netlistAccess

This variable is used to specify the access permissions for the netlist. By default, this variable is set to `User`. Therefore, while netlisting, only the creator of the netlist will be able to run simulation on it. You can also set the values of the variable to `Group` and `All`. By setting the value to `Group`, all the users in the same group as the `User` will be able to run the netlist. If you set the value to `All`, anybody can run the netlist.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.netlist" "netlistAccess" 'string 'User)
```

Variable Type	string
Default Value	User
Acceptable Values	User, Group, and All

GUI Equivalent	--

includeSimLogInOCEAN

Controls whether the simulation log is to be included in the log for OCEAN script. By default, the log for an OCEAN run includes the simulator output. For large simulations, this can result in large log files. In such cases, before running a simulation, you can set this variable to `nil` to exclude the simulator output from the OCEAN job log.

To set this variable in the `.cdsenv` file, use the following call:

```
asimenv.misc includeSimLogInOCEAN 'boolean t
```

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv.misc" "includeSimLogInOCEAN" 'boolean t)
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

numberOfSavedRuns

Once set to value greater than `0`, ADE L will retain the simulation run data for the last *numberOfSavedRuns* simulations. In case of Parametric, a single run may include multiple simulations. At the end of a simulation run, ADE L will save the current run data under:

```
<simulation_dir>/<cell_name>/<simulator_name>/<view_name> to a
numbered directory under <simulation_dir>/<cell_name>/
<simulator_name>.
```

The number used is one higher than the highest numbered directory name or 1 if none exist. If the maximum number of *Saved Runs* is reached, ADE L will save the current run data, but delete the smallest numbered directory, thus keeping the number of Saved Runs equal to the value set in the variable.

Example:

numberOfSavedRuns is set to 2

Under `<simulation>/<ampTest>/<spectre>`

1. At the end of first simulation run

```
1/   schematic/
```

2. At the end of second simulation run

```
1/   2/ schematic/
```

3. At the end of third simulation run

```
2/   3/ schematic/
```

In all the three cases above the schematic directory would have the current simulation results.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.misc" "numberOfSavedRuns" 'int 2)`

Variable Type	int
Default Value	0

obsoleteWarnings

Number of warnings that are needed to be stored. By default, this variable is set to 1. Therefore, while netlisting, one error is shown at a time. If it is desired that more number of errors are shown then change this variable to a larger number.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.netlist" "obsoleteWarnings" 'int 2)`

Variable Type	int
Default Value	1
Acceptable Values	Any integer. The maximum limit is the limit of an integer.
GUI Equivalent	--

oceanScriptFile

Set this variable to specify the default location for saving OCEAN script files.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.misc" "oceanScriptFile" 'string "./myOcnScript")`

Variable Type	string
Default Value	"./oceanScript.ocn"

outputsImportExportVersion

If the value of this variable is greater than `1.0`, then ADE exports or imports the outputs from the CSV file. You can change the value of this variable to less than `1.0` if you want to export or import the outputs from the text file.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.misc" "outputsImportExportVersion" floatValue)`

Variable Type	float
Default Value	1.1
Acceptable Values	any floating value
GUI Equivalent	*Virtuoso Analog Design Environment – Outputs – Export* *Virtuoso Analog Design Environment – Outputs – Import*

paraplotUpdateSimulatorLog

When this variable is set to t, the simulator log appears in a new window, when a paramretric analysis is run.

Default Value	`nil`
Acceptable Values	`t/nil`
Variable Type	Boolean

pcellNameGenerationStyle

Customizes the hierarchical Pcell names using the Unique key. If this variable is set to `UniqueKey`, the Pcell name is generated in a long UUID format. On setting the value as `CustomFormat`, this variable generates a Pcell name based on a user-defined format.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.netlist" "pcellNameGenerationStyle" 'cyclic "Default")`

Variable Type	string
Default Value	`Default`
Acceptable Values	`Default`, `UniqueKey` and `CustomFormat`
GUI Equivalent	--

pcellNameCustomFormat

Customizes the format of hierarchical Pcell names when `CustomFormat` is set as the value for the variable `pcellNameGenerationStye`.

`"paramNme:Name-%VALUE,paramName:Name-%VALUE…"`

Here, `paramName` represents the parameter of the Pcell, `%NAME` specifies the name of the parameter and `%VALUE` specifies the value of the parameter.

For example,

Through this variable, set the Pcell name format as `"r1:%NAME-%VALUE, num:%NAME-%VALUE"`.

The following subcircuit will be generated:

```
subckt pres_pcell_r10_num11_13840648026460803914
<….>
ends pres_pcell_r10_num11_13840648026460803914
```

Here, the parameters `r1` and `num` and the values `0` and `11`, respectively, are added to the subcircuit name.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.netlist" "pcellNameCustomFormat" 'string "")`

Note: This variable also supports user-defined custom strings:

```
envSetVal("asimenv.netlist" "pcellNameCustomFormat" 'string
"r1:ResistorCustomName-%VALUE,num:NumCustomName-%VALUE")
```

Variable Type	string
Default Value	""
Acceptable Value	User-defined customized format for Pcell names.
GUI Equivalent	--

postSaveOceanScript

This procedure is executed after the ocean script is created when the *Save Ocean Script* option is clicked. You can add your own customized code here. Use the following syntax to specify the SKILL functions/procedures:

```
MYlastProc( session fp )
```

In this syntax, `session` is the ADE L session and `fp` is the file pointer to the OCEAN script. You do not need to set these; ADE L sets these for you. In this case, the value for the variable *`postSaveOceanScript`* will be `MylastProc`.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.misc" "postSaveOceanScript" 'string
"myPostSaveProc")
```

Variable Type	string
Default Value	""
Acceptable Values	name of a procedure
GUI Equivalent	--

preferTermOrderOverPinOrder

This variable is used to customize the termOrder printed in the netlist. As per the default order, if schematic pinOrder property is present, this pinOrder property is used. If schematic pinOrder property is not present, the default termOrder from CDF is used. When this variable is set to `t`, the CDF termOrder is used even if the schematic pinOrder property is present. When set to `nil`, the default order explained above is followed.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.netlist" "preferTermOrderOverPinOrder" 'boolean t)`

Variable Type	Boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

preSaveOceanScript

This procedure is executed before the ocean script is created, when the *Save Ocean Script* option is enabled. You can add your own customized code here. Use the following syntax to specify the SKILL functions/procedures:

```
MYfirstProc( session fp )
```

In this syntax, `session` is the ADE L session and `fp` is the file pointer to the OCEAN script. You do not need to set these. ADE L sets these for you. In this case, the value for the variable *`postSaveOceanScript`* will be `MyfirstProc`.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.misc" "preSaveOceanScript" 'string "myPreSaveProc")`

Variable Type	string
Default Value	""
Acceptable Values	name of a procedure
GUI Equivalent	--

printCommentChar

This variable sets the preferred comment character for the printvs data. The # sign is the default comment character. To set the preferred comment character, set the variable, *printCommentChar* to the character.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.userPref" "printCommentChar" 'string "+")`

Variable Type	string
Default Value	#
Acceptable Values	Any integer. The maximum limit is the limit of an integer
GUI Equivalent	--

printInlines

When this variable is set to `t`, data for all devices in a textual subcircuit will be printed. refer to chapter 10. Searching for devices in a textual subcircuit may take some time. If you want to disable this feature, set this variable to `nil`.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.plotting" "printInlines" 'boolean "nil")`

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t/nil`
Window Menu	*Results – Print – DC Operating Points* *Results – print – Transient Operating points* *Results – print – Model Parameters*

printNotation

It is used to specify how numbers are printed in the ADE L environment. This applies only to

Results – Print and *print/printvs* in the Calculator. Numbers are printed in the notation this variable is set to.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("auCore.userPref" "printNotation" 'cyclic "scientific")
```

Variable Type	cyclic
Default Value	suffix
Acceptable Values	engineering, scientific, suffix
GUI Equivalent	--

projectDir

Lets you specify the default simulation directory.

To set this variable in the .cdsenv file, use the call:

```
asimenv projectDir 'string "/tmp/simulation"
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.startup" "projectDir" 'string "/tmp/simulation")
```

Variable Type	string
Default Value	"~/simulation"
Acceptable Values	directory path
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Project Directory*

remoteDir

Lets you specify a path to the run directory for remote simulation. The remote directory name should be same as the local simulation directory name. You must specify a complete path.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.startup" "remoteDir" 'string "/net/cds11938/hm/
```

```
usr1/simulation”)
```

Variable Type	string
Default Value	""
Acceptable Values	Unix path
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Remote Directory*

resizeMode

Specifies how the Virtuoso Visualization and Analysis XL graph window will be resized.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "resizeMode" 'string "manual")
```

Variable Type	string
Default Value	manual
Acceptable Values	manual, auto
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Resize Mode*

retainDesignVarNotation

By default, the value of a design variable is converted using the engineering notation and then displayed in the *Design Variables* pane of ADE L. If this environment variable is set to `t`, the values of the design variables are displayed in the same format that you used in the *Editing Design Variables* form.

To set this environment variable in the `.cdsenv` file, use the following call:
```
asimenv retainDesignVarNotation 'boolean t
```

To set this environment variable in the `.cdsinit` file or CIW, use the following call:
```
envSetVal("asimenv" "retainDesignVarNotation" 'boolean t)
```

Variable Type	Boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	none

retainStateSettings

This variable enables you to retain/ignore the current design or simulator settings in the new design or simulator. Following are the acceptable values:

- `partial` – In case of simulator change, the analyses setup, simulator options, and stimuli information are retained. Other settings such as model files and switch view options are not retained. For these options, either the user-defined settings defined in the `.cdsenv` file are considered, if provided, or their default settings are used. In case of design change, the current design settings and simulator setup files are retained. This is the default value of the variable.
- `design` – Retains all the current settings only when the design is changed.
- `simulator` – Retains all the current settings only when the simulator is changed.
- `all` – Retains the current design settings and simulator setup files. This is same as option `yes`.
- `yes` – Retains the current design settings and simulator setup files.
- `none` – Ignores the current design settings and simulator setup files. This is same as option `no`.
- `no` – Ignores the current design settings and simulator setup files.

It is recommended to use the options `all` or `none` in place of `yes` or `no`, respectively. The options yes and no are being maintained only for backward compatibility.

Note: To change the design, choose *Setup – Design.* To change the simulator, choose *Setup – Simulator/Directory/Host.*

Verify the retained settings, for example the simulator-specific model library files, before running the simulation. This is because some of the retained settings may not apply to the new design or the new simulator, which can result in simulation errors or incorrect simulation results.

To set this variable in the .cdsenv file, use the call:

```
asimenv retainStateSettings cyclic "none"
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "retainStateSettings" 'cyclic "all")
```

Variable Type	cyclic
Default Value	`"partial"`
Acceptable Values	`"all"`, `"none"`, `"partial"`, `"design"`, `"simulator"`

saveAsCellview

This variable lets you save ADE state into a cellview. State is saved and loaded from the cellview.

The default value is `nil`. When set to `nil`, the states are saved and loaded from directories. When set to `t`, state is saved and loaded from the cellview.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv" "saveAsCellview" 'boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Saving State – Cellview* *Virtuoso Analog Design Environment – Loading State – Cellview*

saveDefaultsToOCEAN

When this variable is turned on, in addition to what is normally saved, it saves the following:

- All non-blank options
- All non-blank envOptions
- All enabled analyses and their options (as opposed to all analyses).
- All keep options (save all nets / currents... etc.)
- The model path(s)
- Temperature
- Simulator/analysis defaults to ocean scripts generated from ADE L.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.misc" "saveDefaultsToOCEAN" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

saveDir

This variable identifies the directory in which the saved state file is to be copied. By default, saved state files are to be kept in the `.artist_states` directory in the home directory. You can change this path to another directory as needed.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "saveDir" 'string "~/states/artist_states")
```

Variable Type	string
Default Value	~/.artist_states
Acceptable Values	dir path

saveStateWhenLibMarkedForDeletion

If this variable is set to `t`, you can save an ADE L state, which is saved as a cellview, to another cellview in the same or different library, even when the library containing the original cellview is marked for deletion. By default, when a library is marked for deletion and you try to save an ADE L state saved as a cellview in that library to another cellview in the same or different library, an error message is displayed and the ADE L state cannot be saved.

To set this variable in the `.cdsenv` file, use the call:

```
asimenv saveStateWhenLibMarkedForDeletion  boolean t
```

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv" "saveStateWhenLibMarkedForDeletion" 'boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

saveQuery

Lets you choose whether you want to be reminded to save the state of your environment before making a change. If the option is on, you are prompted to save the state before your environment is changed. If the option is off, you can save the state manually by choosing *Session - Save State*, but you will not be prompted to do so.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "saveQuery" 'boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Query to Save State*

scalarOutputs

If true, displays simulation results that evaluate to scalar values in the graph window.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "scalarOutputs" 'boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Scalar Outputs*

schematicBased

If this variable is set to true, it displays the Analog Design Environment menus on the Virtuoso Schematic window.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv" "schematicBased" 'boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Schematic Menus*

sevResolveSymLinks

When this variable is set to t, the model files are added after resolving symbolic links. If it is nil, symbolic links are not expanded. To prevent symbolic links from being expanded when browsing for model files using the Model Library Setup form, set this variable in the .cdsinit file as `nil`.

Default Value	`t`
Acceptable Values	`t/nil`
Variable Type	Boolean

showConvertNotifyDialog

If you load a state from a previous release, the following dialog box is displayed asking you to convert the state files for the current release:

When you set the value of the `showConvertNotifyDialog` environment variable to `nil`, the state files from the previous release are converted for the current release without displaying the above dialog box.

To set this environment variable in the `.cdinit` file or CIW, use the following call:

```
envSetVal("asimenv" "showConvertNotifyDialog" 'boolean nil)
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
asimenv showConvertNotifyDialog 'boolean nil)
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`
GUI Equivalent	*None*

showWhatsNew

Set this variable to the release number for which you do not want to see the What's New window. For example, set this variable to `5.0.0` if you do not want to see the What's New window for 5.0.0.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv" "showWhatsNew" 'string "5.1.2")`

Variable Type	string
Default Value	"yes"
Acceptable Values	Any existing release number
GUI Equivalent	--

simulationDate

If true, displays the simulation run date in the graph window.

To set this variable in the .cdsinit file or CIW, use the call:

`envSetVal("asimenv.plotting" "simulationDate" 'boolean "nil")`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Simulation Date*

simulator

Sets the default simulator for the Analog Design Environment.

To set this variable in the `.cdsinit` file or CIW, use the following call:

`envSetVal("asimenv.startup" "simulator" 'string "`*`simulatorName`*`|auto")`

When this variable is set to `auto`:

- AMS is set as the default simulator if the view is a config view and it includes Verilog views, which are any text views like `.vams`, `.sv`, `.v`, and so on.

- Spectre is set as the default simulator if the view is not a config view or it does not include Verilog views.

Variable Type	string
Default Value	spectre
Acceptable Values	spectre, ams, UltraSim
GUI Equivalent	*Virtuoso Analog Design Environment – Choosing Simulator/Directory/Host – Simulator*

Example

To set the default simulator as `ams`, use the following call:

```
envSetVal( "asimenv.startup" "simulator" 'string  "ams")
```

To set the default simulator automatically, use the following call:

```
envSetVal("asimenv.startup" "simulator" 'string "auto")
```

skipModelSectionsRetrieval

Allows you to skip parsing the model files in the Model Library Setup form. It is particularly helpful if you are working with large model files and parsing them takes significant time. If it is set as `t`, the Section drop-down list does not appear and you have to add sections manually.

By default, the model file specified in the Section drop-down list is parsed in ADE so that the sections are displayed.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv.misc" "skipModelSectionsRetrieval" 'boolean t).
```

Variable Type	Boolean
Default Value	`nil`
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Model Library Setup*

Example

To skip parsing the model files,

```
envSetVal("asimenv.misc" "skipModelSectionsRetrieval" 'boolean t)
```

smartViewDSPFDirectory

Specifies the location to save the DSPF file that is created by a smart view.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv.misc" "skipModelSectionsRetrieval" 'boolean t)
```

To set this variable in the .cdsenv file, use the following call:

```
envSetVal("asimenv.netlist" "smartViewDSPFDirectory" 'string "/home/
SV_TEST/")
```

Variable Type	String
Default Value	`./simulation/<Lib>/<Cell>/<View>/results/maestro/SmartViewDSPF/.tmpADEDir_<login>/<lib_cell_view>/<subckt>.dspf`
Acceptable Values	Any string value.
GUI Equivalent	None

Example

To save the DSPF generated by the smart view to a directory, `/home/SV_TEST/<subckt>.dspf`

```
envSetVal("asimenv.netlist" "smartViewDSPFDirectory" 'string "/home/SV_TEST/")
```

stateName

This variable specifies the existing `stateName` which would be automatically loaded whenever ADE session is setup. By default, user will need to load state from Load State form. In this case, the state will be searched in the default state directory location specified by the existing `saveDir` variable. The state will be picked from:

```
<saveDir>/<current Lib>/<current Cell>/<Current simulator>/
<stateName>
```

However, if the variable saveAsCellview is set, the state will be loaded as cellview from:

```
<current Lib>/<current Cell>/<stateName>
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "stateName" 'string "state1")
```

Variable Type	string
Default Value	`""`
Acceptable Values	dir path

stateOverWriteSkipList

When you overwrite an existing state in ADE L, all the files and directories in the state directory are deleted before the new state is saved. This variable lets you specify the files and directories that must not be deleted in the state directory when an existing state is overwritten.

To set this variable in the .cdsenv file, use the call:

```
asimenv stateOverWriteSkipList 'string "dir1 dir2 file1 file2"
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv" "stateOverWriteSkipList" 'string "dir1 dir2 file1 file2")
```

Variable Type	string
Default Value	`""`
Acceptable Values	names of files or directories in the state directory, separated by spaces.

strictCDFparseAsNumber

If this variable is set to `t`, only string CDF parameters are considered while retrieving design variables. If this variable is set to `nil`, all CDF parameters for which the *Parse As Number* option is set to `yes` in the Edit CDF form (or the equivalent attribute `parseAsNumber` is set to `yes`) are considered while retrieving design variables.

It is recommended that you set this variable to `t`, so that all incorrectly set up CDF parameters are ignored.

To set this variable in the `.cdsenv` file, use the call:
```
auCore.misc strictCDFparseAsNumber boolean t
```

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("auCore.misc" "strictCDFparseAsNumber" 'boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

temperature

If true, displays the temperature associated with the plotted results in the graph window.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "temperature" 'boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – temperature*

toolList

Set this variable to the list of simulators integrated into ADE. If a new simulator is integrated, it has to be added to this list.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("auCore.toolFilter" "toolList" 'string "spectre")
```

Variable Type	string
Default Value	string "spectre auCdl auLvs"
Acceptable Values	spectre auCdl auLvs

tryNFSSync

Specifies the maximum duration (in seconds) for which the file system must poll for data to make it visible to the host system at a remote location. This prevents plotting failures that occur due to simulation data not being available. These issues often arise because of file system slowdowns across systems.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "tryNFSSync" 'int 30)
```

To set this variable in the .cdsenv file, add:
```
asimenv.distributed tryNFSSync int 30
```

Variable Type	int
Default Value	`30`
Acceptable Values	Any integer value

tryNFSSyncLocal

Specifies the maximum duration (in seconds) for which the file system must poll for data to make it visible to the local host system. This prevents plotting failures that occur due to simulation data not being available. These issues often arise because of file system slowdowns across systems.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("asimenv.startup" "tryNFSSyncLocal" 'int 0)
```

To set this variable in the `.cdsenv` file, add:
`asimenv.startup tryNFSSyncLocal int 0`

Variable Type	int
Default Value	`0`
Acceptable Values	Any integer value

updateCDFtermOrder

If this variable is set to true, it allows updating of the CDF termOrder after a symbol update. The default setting is nil. The CDF updating only affects the termOrder information. Before any updating of the CDF occurs, a dialog box appears and confirms if it is OK to update the base cell CDF termOrder data. The dialog box displays the simulators whose termOrder it will update, and the new termOrder that will be set for each simulator.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("auCore.misc" "updateCDFtermOrder" 'boolean "t")`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

updateCDFtermOrder

If set to `t`, ADE L will automatically update the CDF termOrder when symbol changes that affect the terminal order are made. This will display additional dialog boxes asking you to accept or reject the change to the CDF termOrder.

Variable Type	boolen
Default Value	nil
Acceptable Values	t, nil

useDisplayDrf

Specifies whether the display settings for trace signals and plotting graphs should be applied from the `display.drf` file or from the Virtuoso Visualization and Analysis XL color bank.

When set to `t`, the Analog Design Environment (ADE) uses the display settings from the `display.drf` file for color, style and thickness of trace signals, color highlighting of schematic probes, and plotting waveforms in Virtuoso Visualization and Analysis XL.

Important Points to Note:

- For traces plotted on the graph by using the results browser or calculator, the style, thickness and color bank from the Virtuoso Visualization and Analysis XL are used.
- The display settings from the `display.drf` files are not considered while running sweeps, corners, and parametric analysis in ADE.

When this environment variable is set to `nil`, all the display definitions from Virtuoso Visualization and Analysis XL are used.

For more information, see the following:

- *How do I change the default style, color, and thickness of a waveform or trace in Virtuoso Visualization and Analysis XL?* section in the *Virtuoso Visualization and Analysis XL Frequently Asked Questions*.
- *Graph Environment Variables* section in the *Virtuoso Visualization and Analysis XL User Guide*.

To set this variable in the `.cdsinit` file or CIW to use the display settings from `display.drf` file, use the following call:

```
envSetVal("asimenv.plotting" "useDisplayDrf" 'boolean t)
```

To set this variable in the `.cdsenv` file to use the display settings from `display.drf` file, use the following call:

```
asimenv.plotting useDisplayDrf boolean t
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`
GUI Equivalent	*None*

useNamePrefix

Specifies whether the instance names in the netlist should have the `namePrefix` from the auCdl CDF simulation information or not.

When this environment variable is set to `t`, the `namePrefix` from the auCdl CDF siminfo is prefixed to the name of the instance. As a result, the name of the instance is same in the netlist generated from Spectre and auCdl CDF simulation information.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("asimenv" "useNamePrefix" 'boolean t)
```

To set this variable in the `.cdsenv` file, use the following call:

```
asimenv useNamePrefix boolean t
```

Variable Type	Boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	*None*

variables

If true, displays the names and values of design variables in the graph window.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.plotting" "variables" 'boolean "t")
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Design Variables*

vaTemplateScript

Specifies the name of the template script, which is a shell script, used to customize the header information for new Verilog-A cellviews. The content returned by the template script

is used to generate headers for new Verilog-A cellviews. The header generated by the default script has the following form:

```
//VerilogA for libname, cellname viewname
```

But you can customize this header by using a user-defined template script.

Note: The template file must have executable permission.

To set this environment variable in the .cdsenv file, use the following call:
`auCore.misc vaTemplateScript string "<absolute-path-to-the-template-script>"`

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("auCore.misc" "vaTemplateScript" 'string "<absolute-path-to-the-template-script>")`

Variable Type	string
Default Value	""
Acceptable Values	Any string value

Example

Assume that vaTemplateScript is set to myTemplateScript by using the following call:

```
envSetVal("auCore.misc" "vaTemplateScript" 'string "/home/myTemplateScript")
```

And the specified template script, myTemplateScript, contains the following:

```
#!/bin/sh
echo "// Library: $1"
echo "// Cell: $2"
echo "// View: $3"
echo "// Date: `date`"
```

Now, assume that a new view testView is created in a cell called test, in the library va_test_lib. A new Verilog-A cell is generated with the following information:

```
// Library: va_test_lib
// Cell: test
// View: testView
// Date: Wed Nov 11 11:34:57 GMT 2015
```

windowBased

This variable lets you choose the way the Virtuoso® analog design software starts up your session. If it is true, open the Simulation window. Else do not open the simulation window.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv" "windowBased" 'boolean "nil")
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Simulation Window*

x

Lets you set the horizontal position of the left side of the Simulation window. A selection of 1 (the default) positions the window flush with the left side of your screen. Higher numbers move the Simulation window further to the right.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.window" "x" 'int 200)
```

Variable Type	int
Default Value	1
Acceptable Values	Any number between 0 and 1200
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Window X Location*

x

Enables you to set the horizontal position of the left side of the Virtuoso Visualization and Analysis XL graph window.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.plotting" "x" 'int 500)
```

Variable Type	int
Default Value	577
Acceptable Values	Between 0 and 1200.
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – X Location*

y

Lets you set the vertical position of the top of the Simulation window. A selection of 1, positions the top of the window flush with the top of your screen. Higher numbers move the Simulation window further down the screen. the default positioning places the window about one third of the way down the screen.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.window" "y" 'int 200)
```

Variable Type	int
Default Value	317
Acceptable Values	Any number between 0 and 1000
GUI Equivalent	*Virtuoso Analog Design Environment – Editing Session Options – Window Y Location*

y

Enables you to set the vertical position of the top of the Virtuoso Visualization and Analysis XL graph window.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.plotting" "y" 'int 500)
```

Variable Type	int
Default Value	373

Acceptable Values	Between 0 and 1000.
GUI Equivalent	*Virtuoso Analog Design Environment – Setting Plotting Options – Y Location*

ADE XL

defaultLibName

Use this variable to specify name of the library in which the new view is to be created while opening ADE XL or ADE GXL from ADE L. By default, this variable is not set. You can set a library name by using this variable.

Note: This library name is used only when the showOpenViewDialog variable is set to `nil`.

To set this variable in the `.cdsinit` file or form CIW, use the call:

```
envSetVal("adexl.launchFromTest" "defaultLibName" 'string
"myDefaultLib")
```

To set it in the .cdsenv file, use the call:

```
adexl.launchFromTest defaultLibName ‘string "myDefaultLib"
```

showMenu

Use this variable to show or hide the *Launch* menu that is used to open ADE XL or ADE GXL from ADE L. By default, this variable is set to `t` and the *Launch* menu is visible. To hide the menu, set this variable to `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "showMenu" 'boolean nil)
```

To set it in the .cdsenv file, use the call:

```
adexl.launchFromTest showMenu 'boolean nil
```

showOpenViewDialog

Use this variable to stop the *Launch ADE (G)XL* form from appearing when you open ADE XL or ADE GXL from ADE L. By default, this variable is set to `t` and the *Launch ADE (G)XL* form appears. Using this form, you can either specify the choice to create new adexl view or to open an existing view. If you want to always create a new view when you run ADE XL/GXL from ADE L, set this variable to `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "showOpenViewDialog" 'boolean nil)
```

To set it in the .cdsenv file, use the call:

```
adexl.launchFromTest showOpenViewDialog 'boolean nil
```

viewNamePrefix

Use this variable to add the default prefix to the names of new `adexl` views that are created when you run ADE XL or ADE GXL from ADE L. By default, this variable is set to `adexl_imported_from_adel` and all the view names are prefixed with this string. You can set a different prefix using this variable.

Note: This prefix value is used only when the showOpenViewDialog variable is set as `nil`.

To set this variable in the `.cdsinit` file or from CIW, use the call:

```
envSetVal("adexl.launchFromTest" "viewNamePrefix" 'string "XView")
```

To set it in the .cdsenv file, use the call:

```
adexl.launchFromTest viewNamePrefix ‘string "XView"
```

AMS

ac_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for AC analysis.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("ams.checkOpts" "ac_severity" 'string "None")`

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

AMS_IGNORE_IGNORE

Specifies that the AMS netlister should ignore the `nlAction=ignore` property set on terminals. Those terminals will be included in either the instance port list or the module port list (depending on whether the terminal in question is on an instance or on the cellview being netlisted). For more information about the `nlAction=ignore` property, see the *Virtuoso NC Verilog Environment User Guide*.

To set this environment variable, enter the following command in the UNIX terminal:

```
setenv AMS_IGNORE_IGNORE true
```

Note: Unset this environment variable if you do not want the AMS netlister to ignore the `nlAction=ignore` property set on terminals.

For a comprehensive list of the AMS variables, Appendix C, "Variables for ams.env Files,"

assert_severity_default

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when the device checks are violated.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("ams.checkOpts" "assert_severity_default" 'string "None")`

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification*

dc_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC sweep analysis.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("ams.checkOpts" "dc_severity" 'string "None")`

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

dcOp_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC analysis.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("ams.checkOpts" "dcOp_severity" 'string "None")`

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

disallowRedefinition

If this variable is set to `t`, UNL does not print the `-allowredefinition` option in the `xrunArgs` file.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("ams.ncverilogOpts" "disallowRedefinition" 'boolean t)
```

To set it in the .cdsenv file, add:
```
ams.ncverilogOpts disallowRedefinition boolean t
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	*Virtuoso Analog Design Environment – AMS Options – Disable -allowredefinition option*

dresolution

If this variable is set to `t`, the elaborator uses the detailed discipline resolution method to determine the discipline of nets that do not otherwise have defined disciplines. It applies the `-dresolution` option on the `xmelab` command in the `runSimulation` file, if you are using AMS Cell Based Netlister, or the `xrunArgs` file, if you are using the or AMS Unified Netlister.

For more information on elaboration , see the chapter Elaborating in the *Spectre AMS Designer and Xcelium Simulator Mixed-Signal User Guide*.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("ams.elabOpts" "dresolution " 'boolean t)
```

To set it in the .cdsenv file, add:
```
ams.elabOpts  dresolution  'boolean t
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t, nil`
GUI Equivalent	*Virtuoso Analog Design Environment – AMS Options – Use detailed discipline resolution*

enablePkgImport

This variable is used to set the default selection of field *Enable package importing in AMS Unified Netlister* on the *Netlister* tab of AMS Options form.

To set this variable in the .cdsenv file, add the line
```
ams.netlisterOpts enablePkgImport boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("ams.netlisterOpts" "enablePkgImport" 'boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

fglobalSignals

Adds the signals to the *Global Signals* form for the AMS simulator.

The *Global Signals* form is auto populated after the AMS netlister extracts the signals from the schematic. AMS netlister cannot extract the signals when the signals are part of text views, or if some hierarchy inside the HED is not visible to AMS netlister. In such cases you need to add inaccessible signals to the *Global Signals* form.

The syntax for adding the signal is

```
;[origin];globalSignalName;namespace;wireType;[discipline];isGround;F;extractedWi
     reType;
```

where

`origin`	(optional) Specifies the origin of the signal. If the signal is extracted from the schematic, the value is `D`. If you are adding the signal manually, leave this as blank.
`globalSignalName`	Specifies the name of the signal.
`namespace`	Specifies the namespace in which the signal was created. Acceptable values: `CDBA`, `Spectre`, `Spice`, and `Verilog-AMS`.
`wireType`	Specifies the wire type of the signal. Acceptable values: `wire`, `supply0`, `supply1`, `wreal`, `wor`, `wand`, `tri`, `tri0`, `tri1`, `triand`, `trior`, and `trireg`.
`discipline`	(optional) Specifies the discipline of the signal.
`isGround`	Specifies if the global signal is used as a ground reference. Acceptable values: `YES`, `NO`.
`F`	It is an internally managed flag, and its value should always be `F`.
`extractedWireType`	Specifies the wire type extracted by the netlister. It is also an internally managed flag, and its value should be same as `wireType`.

To set this variable in the `.cdsinit` file or CIW for adding the global signal `gnd!`, use the following call:

```
envSetVal("ams.netlisterOpts" "globalSignals" 'string
     ";;gnd!;CDBA;wire;;YES;F;wire;")
```

To set this variable in the `.cdsinit` file or CIW for adding the global signals `gnd!` and `vdd12!`, use the following call:

```
envSetVal("ams.netlisterOpts" "globalSignals" 'string
     ";;gnd!;CDBA;wire;;YES;F;wire; ;;vdd12!;CDBA;wire;;NO;F;wire;")
```

Important Points to Note:

- Add a space if you are adding more than one signal. In the above example, a space is present after the values of `gnd!` signal.

- If you do not want to specify any value for the optional fields, do not replace them with a space. In the above example, the value for `discipline` is not specified.

To set this variable in the `.cdsenv` file for adding the global signal `gnd!`, use the following call:

```
ams.netlisterOpts globalSignals 'string ";;gnd!;CDBA;wire;;YES;F;wire;"
    |
```

Variable Type	string
Default Value	`""`
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister tab – Global Signals*

hnlVerilogIgnoreTermNameList

Can be used to specify the terminals that the AMS UNL Environment must ignore when generating the logical netlist of a schematic. This variable eliminates the need to modify the schematic by specifying the signal type.

To set this variable in the `.cdsinit` file, in CIW or in the `. simrc` file, use the call:
`hnlVerilogIgnoreTermNameList=(list "ignoreTermName1" "ignoreTermName2" ...)`

Variable Type	list of strings
Default Value	`nil`
Acceptable Values	List of terminals names
GUI Equivalent	--

ignorePrintNettype

Controls the printing of property `netType` from the design schematic to the `netlist.vams` file.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:
`envSetVal( "ams.netlisterOpts" "ignorePrintNettype" 'boolean t)`

To set this environment variable in the `.cdsenv` file, use the following call:
`ams.netlisterOpts ignorePrintNettype boolean t`

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t, nil`
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister Tab*

lTnetlistMaxWarn

Use the `netlistMaxWarn` environment variable to specify the maximum number of warning messages issued by the netlister before it stops processing the design.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:
`envSetVal("ams.netlisterOpts" "netlistMaxWarn" 'string "3")`

To set this environment variable in the `.cdsenv` file, use the following call:
`ams.netlisterOpts netlistMaxWarn 'string "3"`

Variable Type	string
Default Value	-
Acceptable Values	Any integer
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister Tab*

netlistBlackboxWithExtHdl

If set to `t`, the netlist uses a symbol view to generate an instance line for a blackbox in an extracted view. This black box must have the external HDL properties set in the *HED* (*Hierarchy Editor*).

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

`envSetVal("ams.netlisterOpts" "netlistBlackboxWithExtHdl" 'boolean nil)`

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts netlistBlackboxWithExtHdl 'boolean nil
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	*Simulation – Options – AMS Simulator – AMS Options form – Use symbols for blackboxes bound to external HDL*

netlistNoWarn

You can suppress the information or warning messages issued by the netlister while processing the design by specifying a space-separated list of message IDs, such as `AMS-2000 AMS-2171 AMS-2174`, to the `netlistNoWarn` environment variable.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "netlistNoWarn" 'string "AMS-2000 AMS-2171 AMS-2174")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts netlistNoWarn 'string "AMS-2000 AMS-2171 AMS-2174"
```

Variable Type	string
Default Value	-
Acceptable Values	A space-separated list of warning message IDs
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister Tab*

print_control_vars

Enables the AMS netlister to print a list of control variables for which the default AMS value has been changed in the `changedVarsSummary` file. This file is saved in the netlist directory.

The syntax of this variable is as follows:

```
envSetVal("ams.netlisterOpts" "print_control_vars" 'boolean {t|nil})
```

The default value of this variable is `t`. If you do not want the `changedVarsSummary` file to be created, unset the variable in `.cdsinit` or `.cdsenv`.

To unset this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "print_control_vars" 'boolean nil)
```

To unset this variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts print_control_vars 'boolean nil
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`

shortInstanceList

You can use this environment variable to specify a list of libraries or cells in which devices are to be shorted. When you netlist the design, devices in the specified libraries or cells that meet the following criteria appear shorted in the generated netlist:

- The device must only have two terminals.
- It must be an analog instance.
- It must be a stopping device.

Note: This environment variable is supported only for the AMS UNL netlister.

To set this variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "shortInstanceList" 'string
"(library_name1)(library_name2 cell_name1)(library_name3)")
```

To set this variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts shortInstanceList 'string
"(library_name1)(library_name2 cell_name1)(library_name3)"
```

Variable Type	String
Default Value	`""`

Acceptable Values	List of library names or library-cell pairs, each enclosed within parentheses.

simOutputFormat

Use this variable to specify the format of output results. Setting the value to `sst2` causes AMS to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("ams.outputs" "simOutputFormat" 'string "wdf")
```

To set it in the .cdsenv file, add:

```
ams.outputs simOutputFormat 'string "wdf"
```

Variable Type	string
Default Value	`sst2`
Acceptable Values	`sst2, fsdb, wdf`

tran_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the transient analysis.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("ams.checkOpts" "tran_severity" 'string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

upgradeMsgSevError

Changes the severity of the specified warning messages to error while processing a design. The message IDs should be provided as a space-separated list, such as `AMS-2171 AMS-2174`.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "upradeMsgSevError" 'string "AMS-2171 AMS-2174")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts upradeMsgSevError 'string "AMS-2171 AMS-2174"
```

Variable Type	string
Default Value	-
Acceptable Values	A space-separated list of warning message IDs
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister Tab*

upgradeMsgSevWarn

Changes the severity of the specified information messages to warning while processing a design. The message IDs should be provided as a space-separated list, such as `AMS-1244 AMS-1246`.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("ams.netlisterOpts" "upradeMsgSevWarn" 'string "AMS-1244 AMS-1246")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
ams.netlisterOpts upradeMsgSevWarn 'string "AMS-1244 AMS-1246"
```

Variable Type	string
Default Value	-
Acceptable Values	A space-separated list of information message IDs
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation – Options – AMS Simulator – Netlister Tab*

useEffectiveCDF

When this variable is set to t, AMS in ADE uses effective CDF while netlisting. If set to `nil`, the base cell CDF will be used. To set this variable in the .cdsenv file or CIW, use the call:

```
envSetVal("ams.netlisterOpts" "useEffectiveCDF" 'boolean nil)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t/nil`

ams.envOpts

- amsIEsList
- autoConfigNameForConfigCreation
- connectRulesList
- constraintListForConfigCreation
- defaultVsupForVAR
- disableCompileVAAsVAMS
- disableRunModeInDP
- filesOnIrunCmdLineHDL
- ipExportHDLFiles
- irunIncDirHDL
- libDirsHDL
- libFilesHDL
- libraryListForConfigCreation
- maxNumSnapShots
- optsFileHDL
- saveAllSnapShots
- snapShotBaseName
- snapShotNameTimeUnit
- snapShotSaveMode
- stopListForConfigCreation
- strobeTime
- useIeSetup
- viewListForConfigCreation

Calculator

dstack

This field is set to display the contents of the stack. This is available only for the RPN mode.

To set this variable in the .cdsenv file, add the line:
```
calculator dstack ‘boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("calculator" "dstack" ‘boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
GUI Equivalent	*Calculator*

eval

This field is set to evaluate the contents of a calculator buffer automatically. This is available only for the RPN mode.

To set this variable in the .cdsenv file, add the line:
```
calculator eval ‘boolean t
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("calculator" "eval" ‘boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	*Calculator*

mode

This variable sets the mode for creating expressions.

To set this variable in the .cdsenv add the line:
```
calculator mode cyclic "algebraic"
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("calculator" "mode" 'cyclic "algebraic")
```

Variable Type	cyclic
Default Value	RPN
Acceptable Values	{RPN, algebraic}
GUI Equivalent	*Calculator – Options*

uimode

This variable sets the mode of operation for the calculator.

To set this variable in the .cdsenv file, add the line:
```
calculator uimode cyclic "RF"
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("calculator" "uimode" 'cyclic "RF")
```

Variable Type	cyclic
Default Value	standard
Acceptable Values	{standard, RF}
GUI Equivalent	*Calculator*

Distributed Processing

autoJobSubmit

If this variable is set to a non-nil value, the *Job Setup* form is not displayed at job submit time.

To set this variable in the .cdsenv file, add the line:
```
asimenv.distributed autoJobSubmit 'boolean nil
```

To set this variable in the .cdsinit file or CIW, use the call
```
envSetVal("asimenv.distributed" "autoJobSubmit" 'boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
GUI Equivalent	*Setup – Simulator/Directory/Host*

block

If this variable is set to a non-nil value, the process is blocked until the job has completed.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "block" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

copyMode

If this variable is set to a non-nil value, the input data for the job is copied to `/tmp` on the execution host, the job is run there locally (without network read/write), and the output data is copied back to the submission host.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.distributed" "copyMode" 'boolean t)`

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

copyModeDir

Specifies the directory relative to the execution host, that will be used for setting up the working directory of a copy mode job.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.distributed" "copyModeDir" 'string "dirname")`

Variable Type	string
Default Value	`/tmp`
Acceptable Values	Any string value
GUI Equivalent	none

copyNetlist

Specifies whether the netlist directory needs to be copied from the execution host to the submission host. This may be required if during simulation, some files are generated under the netlist directory.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.distributed" "copyNetlist" 'boolean t)`

Variable Type	boolean
Default Value	nil

Acceptable Values	{nil, t}
GUI Equivalent	none

daysBeforeExpire

Specifies the number of days after which terminated jobs will be deleted from the job server.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "daysBeforeExpire" 'int 6)
```

Variable Type	int
Default Value	3
Acceptable Values	Any String Value
GUI Equivalent	none

drmsCommandList

Enables you to specify multiple DRMS commands in the *command* drop-down list box of the *Virtuoso Analog Distributed Processing option Job Submit* form.

The DRMS command specified using the `drmsCommand` environment variable is set as the default value for the *command* drop-down list box. If the `drmsCommand` environment variable is not set, then the first command specified using the `drmsCommandList` variable is set as default.

For more information, see *Submitting Jobs using the Command Mode*.

To set this variable in the `.cdsinit` file or CIW for specifying multiple DRMS commands, use the following call:

```
(envSetVal "asimenv.distributed" "drmsCommandList" 'string "bsub -R
\"OSNAME==Linux && OSREL==EE80\";bsub -R \"OSNAME==Linux && OSREL==EE60\";bsub -R
\"OSNAME==Linux && OSREL==EE50 && SFIARCH==EM64T && OSBIT==64\"")
```

To set this variable in the `.cdsenv` file for specifying multiple DRMS commands, use the following call:

```
asimenv.distributed drmsCommandList string "bsub -R \"OSNAME==Linux &&
OSREL==EE80\";bsub -R \"OSNAME==Linux && OSREL==EE60\";bsub -R \"OSNAME==Linux &&
OSREL==EE50 && SFIARCH==EM64T && OSBIT==64\""
```

Important points to note:

- The DRMS commands specified in the example above are the resource strings generated from the farm. You can define multiple resource strings using this environment variable with each string separated by a semi-colon (`;`).
- Only the first ten commands specified using the `drmsCommandList` variable are added to the *command* drop-down list box.

Variable Type	string
Default Value	`""`
Acceptable Values	DRMS commands in correct syntax
GUI Equivalent	None

emailNotify

If this variable is set to a non-nil value, an e-mail notification is provided, following job termination. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "emailNotify" 'boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
GUI Equivalent	*Job Submit Form*

expDay

This variable sets the default expiration day for a job. If the expiration day is set as `today`, then the job will always run on the same day it is submitted. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.distributed" "expDay" 'cyclic "Friday")`

Variable Type	cyclic
Default Value	`today`
Acceptable Values	{`today, Sunday, Monday, Tuesday Wednesday, Thursday, Friday, Saturday` }
GUI Equivalent	*Job Submit Form*

expTime

This variable sets the default expiration time for a job (in 24 hour format). If unspecified, the expiration time is based on the value of the `timeLimit` variable. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.distributed" "expTime" 'string "00:43")`

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
GUI Equivalent	*Job Submit Form*

externalServer

If this variable is set to a non-nil value, the job server is started remotely.
If this variable is set to a non-nil value, the job server is started remotely.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("asimenv.distributed" "externalServer" 'boolean nil)`

Variable Type	boolean

Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

hostName

This variable sets the default host name. If unspecified, the host is selected automatically. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsenv file, add the line:

```
asimenv.distributed hostName 'string "host"
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "hostName" 'string "host")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
GUI Equivalent	*Job Submit Form*

jobArgsInOceanScript

Indicates job arguments that should be added to run commands when an OCEAN script is generated.

```
envSetVal("asimenv.distributed" "jobArgsInOceanScript" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}
GUI Equivalent	none

loginShell

Specifies the login shell for the job. If it is specified as `none` then the users local environment is copied over to the execution host and used as the jobs environment.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "loginShell" 'cyclic "csh")
```

Variable Type	cyclic
Default Value	`none`
Acceptable Values	{`none, csh, ksh, sh`}
GUI Equivalent	none

logsInEmail

If this variable is set to a non-nil value, `stdout` and `stderr` logs will be included in the termination E-mail.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "logsInEmail" 'boolean t)
```

Variable Type	boolean
Default Value	t
Acceptable Values	{t, nil}
GUI Equivalent	none

mailAllLogs

Sends out a mail after completion of all the tasks and each individual task (when set to `t)`.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "mailAllLogs" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil, t}

mailTo

This variable sets the default list of users who will receive job termination notification e-mail. If unspecified and if *emailNotify* is `t`, then the default value is the user's ID. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "mailTo" 'string
"userId123@cadence.com")
```

Variable Type	string
Default Value	""
Acceptable Values	Any Valid Id String Value.
GUI Equivalent	*Job Submit Form*

numOfTasks

Specifies the default number of tasks a job should be broken into. This is used by the Monte Carlo tool. If zero, then the number of tasks is based on queue and/or host settings.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "numOfTasks" 'int 5)
```

Variable Type	int
Default Value	0
Acceptable Values	Any integer value
GUI Equivalent	none

puttogetherqueue

Specifies the queue to be used for the Put Together Job.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "puttogetherqueue" 'string
"queuename")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
GUI Equivalent	none

queueName

The variable sets the default queue name. If unspecified, the system default is used. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsenv file, add the line:
```
asimenv.distributed queueName 'string "myqueue"
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "queueName" 'string "myqueue")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value
GUI Equivalent	*Job Submit Form*

setupFunction

Triggers the user-defined SKILL function to update the values in the *Virtuoso Analog Distributed Processing option Job Submit* form.

The SKILL function should return the values in a list of key-value format, such as `?drmsCommand command ?queueName myQueue`.

The user-defined SKILL function can return the following values:

Value	Data Type	Description
queue	string	The queue in which the job should be submitted.
host	string	The name of the machine on which the job should be launched.
drmsCommand	string	The DRMS command to submit the job.
mail	string	The e-mail addresses to which the e-mail should be sent after the job terminates.
dependentOn	string	The job dependency list.
block	Boolean	The blocking mode. Specifies whether the analog design environment or OCEAN should be blocked until all the jobs have completed.
tasks	integer	The number of tasks.
lsfResourceStr	string	The additional resource requirements for the job, such as `mem` (available memory), `swp` (available swap space), and `pg` (paging rate).
lsfLicenseProject	string	The name of the LSF license project.
lsfNoOfProcessors	string	The number of parallel processors to be used to run the submitted job in LSF.
sgeNoOfProcessors	string	The number of parallel processors to be used to run the submitted job in SGE.
sgeSoftResourceStr	string	The requirements for soft resources in SGE.
sgeHardResourceStr	string	The requirements for hard resources in SGE.
sgePriority	string	The priority for the job submitted in SGE.
sgeParallelEnvName	string	The name of a parallel environment in SGE.

startTime	string	The start time of the job in `?time` and `?day` format. `?time` is in 24-hour format. For example, `05:40`, `15:30`, or `17:25`. `?day` has the following acceptable values: `today`, `Sunday`, `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, and `Saturday`.
termTime	string	The time after which the submitted job should be terminated in the `?time` and `?day` format. `?time` is in 24-hour format. For example, `05:40`, `15:30`, or `17:25`. `?day` has the following acceptable values: `today`, `Sunday`, `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, and `Saturday`.
jobName	string	The name of the job to be submitted.

You can create the user-defined function named `myCustomSetup` as follows:

```
procedure(myCustomSetup()
list(?drmsCommand "bsub -R \"OSNAME==Linux && OSREL==EE50\"" ?jobName "myJobName")
)
```

You can dynamically redefine the `myCustomSetup` function in CIW as follows:

```
myCustomSetup
procedure(myCustomSetup()
list(?drmsCommand "bsub -R \"OSNAME==Linux && OSREL==EE50\"" ?startTime "12:00"
?jobName "myJobNameStart")
)
=>function myCustomSetup redefined
myCustomSetup
```

To set this variable in the `.cdsenv` file to trigger the above function, use the following call:

```
asimenv.distributed setupFunction 'string "myCustomSetup"
```

To set this variable in the `.cdsinit` file or CIW to trigger the function, use the following call:

```
envSetVal("asimenv.distributed" "setupFunction" 'string "myCustomSetup")
```

Variable Type	string
Default Value	" "
Acceptable Values	SKILL function name
GUI Equivalent	None

showMessages

If this variable is set to a non-nil value, a message is displayed in the CIW or OCEAN terminal on the completion of a job.

To set this variable in the .cdsenv file, add the line:
```
asimenv.distributed showMessages 'boolean nil
```

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("asimenv.distributed" "showMessages" 'boolean nil)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	{nil t}
GUI Equivalent	none

startDay

This variable sets the default start day for a job. If the start day is set as `today`, then the job will always run on the same day it is submitted.

To set this variable in the .cdsinit file or CIW, use the call
```
envSetVal("asimenv.distributed" "startDay" 'cyclic "Wednesday")
```

Variable Type	cyclic
Default Value	`today`

Acceptable Values	{`today, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday` }
GUI Equivalent	*Job Submit Form*

startTime

This variable sets the default start time for a job (in 24hour format). If unspecified, the job executes immediately. For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "startTime" ‘string "23:11")
```

Variable Type	string
Default Value	""
Acceptable Values	Any String Value (HH:MM)
GUI Equivalent	*Job Submit Form*

stateFile

This variable sets the filename containing the job server’s state.

This variable sets the filename containing the job server’s state.
To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "stateFile" ’string "myStateFile")
```

Variable Type	string
Default Value	`~/.adpState`
Acceptable Values	Any String Value
GUI Equivalent	none

timeLimit

This variable sets the default time limit for a job. If the time limit is set to `none`, then no time limit is imposed. If unspecified, then expiration time is based on value of *`expTime`* and *`expDay`* variables.For details, refer to the *Submitting a Job* section, of Chapter 2 of the *Virtuoso Analog Distributed Processing Option User Guide*.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("asimenv.distributed" "timeLimit" 'cyclic "5 minutes")
```

Variable Type	cyclic
Default Value	none
Acceptable Values	{`unspecified, none, 5 minutes, 15 minutes, 30 minutes, 1 hour, 3 hours, 6 hours, 12 hours, 1 day, 2 days, 3 days, 5 days, 10 days`}
GUI Equivalent	*Job Submit Form*

HspiceD

checkOutLicenseDuringNetlistAndRun

If set to `t`, the license `32760` “Virtuoso Analog HSPICE Interface Option” only remains checked out during netlisting and simulation. After these tasks are completed, the license gets checked back in to the license server.

The default value is `nil`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "checkOutLicenseDuringNetlistAndRun"
'boolean nil)
```

To set it in `.cdsenv`, use the call:

```
hspiceD.envOpts checkOutLicenseDuringNetlistAndRun 'boolean nil
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`

hspiceMaxLineLength

Controls the maximum limit on the number of characters to be printed in a line of netlist output from the default. The maximum limit value can be increased or decreased as required. This variable overrides the OSS variable, `maxLineLength`.

Note: The `hspiceSoftLineLength` value can never be greater than `hspiceMaxLineLength` value.

To set this variable in the .cdsinit file or CIW, use the call:

```
hspiceMaxLineLength=1024
```

Variable Type	int
Default Value	`1024`

Acceptable Values	Any integer. The maximum limit is the limit of an integer.

For more information, see the *Customizing the Hierarchical Netlister (HNL)* chapter of the *Open Simulation System Reference*.

hspiceSoftLineLength

This variable controls the maximum number of characters in a line of netlist output after which the line is automatically split into multiple lines for reading effortlessly. This variable overrides the OSS variable, `softLineLength`, which is set to 1024 characters by default.

To set this variable in the .cdsinit file or CIW, use the call:

```
hspiceSoftLineLength=80
```

Variable Type	int
Default Value	`1024`
Acceptable Values	`Any integer less than or equal to 1024`

mapGndNetToZero

Use this variable to control mapping of `gnd!` nets. By default, its value is set to `t` which implies `gnd!`nets are mapped to 0. To stop the mapping of `gnd!` nets to 0, set the variable to `nil`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "mapGndNetToZero" 'boolean nil)
```

To set it in `.cdsenv`, add:

```
hspiceD.envOpts mapGndNetToZero 'boolean nil
```

Variable Type	boolean
Default Value	`t`

Acceptable Values	`t/nil`

netlistModelFileFirst

Use this variable to control sequence of netlisting of design variables and include files. By default, the design variables are netlisted before the include files. To netlist the include files before the design variables, set the value of `netlistModelFileFirst` variable to `t`.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "netlistModelFileFirst" 'boolean t)
```

To set it in `.cdsenv`, add:

```
hspiceD.envOpts netlistModelFileFirst 'boolean t
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t/nil`

setTopLevelAsSubckt

This variable controls whether the top-level schematic should be netlisted as a subcircuit or not. If it is set to `t`, the top-level schematic is netlisted as a subcircuit; otherwise, it is not netlisted as a subcircuit.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal( “hspiceD.envOpts” “setTopLevelAsSubckt” ‘boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t,nil`

userCmdLineOption

Use this variable to pass command-line options in hspice. For example, to set multi-threading to 3, you need to pass `-mt 3` as the value.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("hspiceD.envOpts" "userCmdLineOption" 'string "-mt 3")
```

To set this variable in `.cdsenv` file, add:

```
hspiceD.envOpts userCmdLineOption 'string "-mt 3"
```

Variable Type	string
Default Value	""
Acceptable Values	Any string value

Analysis

relxpert_gradual_aging

If this variable is set, the *Gradual Aging* tab is removed form the *Simulation – – Setup* form.

To set this variable in the .cdsinit file or CIW, use the call
`putprop _amsUISimFeatures nil 'relxpert_gradual_aging`

Variable Type	boolean
Default Value	t
Acceptable Values	{nil t}
GUI Equivalent	*Simulation – – Setup*

spectre_analysis_

If this variable is set, the *RelXpert* option is set as default Simulation Mode instead of *Spectre Native* in the *Simulation – – Setup* form.

To set this variable in the .cdsinit file or CIW, use the call
`putprop _amsUISimFeatures nil'spectre_analysis_`

Variable Type	boolean
Default Value	t
Acceptable Values	{nil t}
GUI Equivalent	*Simulation – – Setup*

Spectre

AC Match Analysis Environment Variables

The table below lists all the environment variables that you can set in the `.cdsenv` file or CIW to modify the setup and options for the AC Match analysis.

Note: The values displayed in the variable syntax are the default values.

Variable	Description and Syntax	possible Values
`dec`	Sets points per decade. `envSetVal(spectre.acmatch dec string "")`	
`lin`	Sets the number of steps for a linear sweep. `envSetVal(spectre.acmatch lin string "")`	
`log`	Sets the number of steps for a log sweep. `envSetVal(spectre.acmatch log string "")`	
`mth`	Sets the threshold of the relative contribution to be exported in output file. `envSetVal(spectre.acmatch mth string "")`	
`oprobe`	Specifies the mismatch variations of the current signal, based on this, component is chosen as the output. `envSetVal(spectre.acmatch oprobe string "")`	`vsource`, `inductor`, `switch`, `tline`, `iprobe`, `ccvs`, `vcvs`, `pccvs`, and `pvcvs`.
`annotate`	Sets the degree of annotation. `envSetVal(spectre.acmatchOpts annotate string "status")`	`no`, `title`, `sweep`, `status`, and `steps`
`force`	Sets the set of initial conditions to be used. `envSetVal(spectre.acmatchOpts force string "none")`	`none`, `node`, `dev`, and `all`.

Variable	Description and Syntax	possible Values
groupby	Determines how to group total sigma. The total sigma of each contributor can be grouped either by statistics parameters or by subckt instances. envSetVal(spectre.acmatchOpts groupby string "param")	param and inst
oppoint	Determines whether operating point information should be computed. If yes, specifies where should it be printed (screen or file). Operating point information is not printed if the operating point computed in the previous analysis remains unchanged. envSetVal(spectre.acmatchOpts oppoint string "no")	no, screen, logfile, and rawfile
prevoppoint	Determines whether to use the operating point computed in the previous analysis. envSetVal(spectre.acmatchOpts prevoppoint string "no")	yes and no
readns	Specified the file that contains an estimate of DC solution (nodeset). envSetVal(spectre.acmatchOpts readns string "")	
restart	Restarts the DC solution from scratch if any condition has changed. If not, use the previous solution as an initial guess. envSetVal(spectre.acmatchOpts restart string "yes")	yes and no
save	Specified the signal levels to be saved in output. envSetVal(spectre.acmatchOpts save string "")	all, lvl, allpub, lvlpub, selected, none, and nooutput
skipdc	Skips DC analysis. envSetVal(spectre.acmatchOpts skipdc string "no")	yes and no
useprevic	If set to yes or ns, use the converged initial condition from the previous analysis as ic or ns. envSetVal(spectre.acmatchOpts useprevic string "no")	no, yes and ns

Variable	Description and Syntax	possible Values
where	Specifies where the results should be printed. envSetVal(spectre.acmatchOpts where string "screen")	screen, logfile, file, and rawfile
additionalParams	Specifies the additional parameters. envSetVal(spectre.acmatchOpts additionalParams string "")	
incrType	Sets the sweep type for AC Match. envSetVal(spectre.acmatch incrType string "Automatic")	Automatic, Linear, Logarithmic
outType	Sets the output type. envSetVal(spectre.acmatch outType string "voltage")	voltage and probe
rangeType	Sets the frequency sweep range. envSetVal(spectre.acmatch rangeType string "Start-Stop")	Start-Stop, Center-Span, Single-Point
span	Sets the sweep limit span. envSetVal(spectre.acmatch span string "")	
center	Specifies the center of the sweep. envSetVal(spectre.acmatch center string "")	
start	Specifies the start value for the sweep limit. envSetVal(spectre.acmatch start string "")	
stop	Specifies the stop value for the sweep limit. envSetVal(spectre.acmatch stop string "")	
step	Sets the step size for the linear sweep. envSetVal(spectre.acmatch step string "")	
useDiscrete	Determines whether to add specific points. envSetVal(spectre.acmatch useDiscrete boolean nil)	t/nil

Variable	Description and Syntax	possible Values
values	Specifies the value of specific points. envSetVal(spectre.acmatch values string "")	
P	Specifies the value of positive output node. envSetVal(spectre.acmatch p string "")	
n	Specifies the value of negative output node. envSetVal(spectre.acmatch n string "")	
stepTypeLin	Specifies the step type for Linear. envSetVal(spectre.acmatch stepTypeLin string "Step Size")	Step Size, Number of Steps
stepTypeLog	Specifies the step type for Logarithmic. envSetVal(spectre.acmatch stepTypeLog string "Points Per Decade")	Points Per Decade, Number of Steps

ac_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for AC analysis.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.checkOpts" "ac_severity" 'string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

analysisOrder

Determines the order in which the analyses would be run by the simulator.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.envOpts" "analysisOrder" 'string "tran ac dc")
```

Variable Type	string
Default Value	""
Acceptable Values	Names of analysis in the order desired
GUI Equivalent	*Virtuoso Analog Design Environment* – Environment Options – *Analysis Order*

apsplus

This variable controls the value of the *Use ++aps* check box that is displayed in the High Performance Simulation Options form. If set to `t`, it enables Fast APS mode for Spectre simulator.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.turboOpts" "apsplus" 'boolean t)
```

To set it in .cdsenv, add:
```
spectre.turboOpts  apsplus 'boolean t
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment – Setup – High-Performance Simulation – APS – Use ++aps*

assert_severity_default

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when the device checks are violated.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.checkOpts" "assert_severity_default" 'string
"None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification*

autoDisplay

This variable is used to set/reset the *Automatic output log* option. When on, the output log opens and displays simulator messages as they are generated.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.envOpts" "autoDisplay" 'boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment –* Environment Options *– Automatic output log*

autoDisplayBBox

This variable is used to control the size of the `spectre.out` window.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.envOpts" "autoDisplayBBox" 'string "10 10 525
800")
```

Variable Type	string
Default Value	"0 0 515 700"
Acceptable Values	window coordinates
GUI Equivalent	--

checkpoint

Y runs spectre with the `+checkpoint` option, which turns on the checkpoint capability. N runs spectre with the `-checkpoint` option, which turns off the checkpoint capability.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "checkpoint" 'string "Y")`

Variable Type	string
Default Value	""
Acceptable Values	Y, N
GUI Equivalent	*Virtuoso Analog Design Environment* – Environment Options – *Create Checkpoint File(cp)*

controlMode

Used to run Spectre in batch or interactive modes depending on the value of the variable.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "controlMode" 'string batch)`

Variable Type	string
Default Value	interactive
Acceptable Values	interactive, batch
GUI Equivalent	--

Note:

- All the Spectre simulator options are documented under `spectre -h` options and there is one-to-one correspondence between `spectre.<optName>` and `<optName>`
- All the analysis options are documented under `spectre -h <analysisName>`
- The following variables are deprecated:

  `spectre.init remoteDir 'string ""`

```
spectre.init hostmode 'string "local"
spectre.init host 'string ""
spectre.init settableResultsDirectory 'boolean t
spectre.init processPriority int 0 0 20
```

currents

The currents parameter of the options statement computes and saves terminal currents. Use it to create settings for currents that apply to all terminals in the netlist.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.outputs" "currents" 'string "nonlinear")`

Variable Type	string
Default Value	""
Acceptable Values	selected, all, nonlinear
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Select device currents (currents)*

dc_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC sweep analysis.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.checkOpts" "dc_severity" 'string "None")`

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

dcOp_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the DC analysis.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.checkOpts" "dcOp_severity" 'string "None")`

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

definitionFiles

Type the full UNIX path or the name of one or more files. A definitions file contains function definitions and definitions of parameters that are not displayed in the Design Variables section of the simulation window.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "definitionFiles" 'string "./file1")`

To disable a definition file, use a hash sign (#) as shown in the example below:
`envSetVal("spectre.envOpts" "dspfFile" 'string "#;./file1")`

Here, the file `file1` will be disabled.

Variable Type	string
Default Value	""
Acceptable Values	unix path or name of one or more files.
GUI Equivalent	*Virtuoso Analog Design Environment –* Simulation Files Setup *– Definition Files*

displayToolTip

Controls whether the tooltips for various fields are displayed on the Options forms for various analyses supported by Spectre. For example, AC Options, DC Options, Periodic Steady State Options and so on.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("spectre.envOpts" "displayToolTip" 'boolean t)`

To set this variable in `.cdsenv` file, add:
`spectre.envOpts displayToolTip 'boolean t`

Variable Type	boolean
Default Value	`t`
Acceptable Values	`t`, `nil`
GUI Equivalent	––

dochecklimit

When set to `yes`, enables device checking without selecting the check box in form *Simulation – Device Checking – Enable Device checking*.

To set this option in `.cdsenv` or `CIW`, use the call:

`envSetVal("spectre.opts" "dochecklimit" 'string "yes")`

Variable Type	string
Default Value	yes
Acceptable Values	yes,no
GUI Equivalent	*Virtuoso Analog Design Environment – Simulation/Device Checking – Use Enable Device Checking*

You can alternatively choose *Simulation – Options – Analog(Spectre) – Check – Device Checking Options.*

dspfFile

This variable is used to set the path of the parasitic DSPF file.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "dspfFile" 'string "./dspfFile1.dspf")`

To disable a DSPF File, use a hash sign (#) as shown in the example below:
`envSetVal("spectre.envOpts" "dspfFile" 'string "#;./dspfFile1.dspf")`

Here, the file `dspfFile1.dspf` will be disabled.

Variable Type	string
Default Value	""
Acceptable Values	unix path
GUI Equivalent	*Virtuoso Analog Design Environment L– Simulation Files Setup – Parasitics File (DSPF)*

emLayerMap

Specifies the path of the Layer Map file to be used for EM/IR Analysis.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "emLayerMap" 'string "./file1")`

To set this variable in the .cdsenv file or CIW, add:
`spectre.envOpts emLayerMap string "./file1"`

Variable Type	string
Default Value	nil
Acceptable Values	unix path
GUI Equivalent	*Setup – EM/IR Analysis – Layer Map File*

emirEnable

Enables the EMIR analysis.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "emirEnable" 'boolean t)
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts emirSumList 'boolean t
```

Variable Type	Boolean
Default Value	`t`
Acceptable Values	`t`, `nil`
GUI Equivalent	*None*

emirLCVDisable

Disables the *Library*, *Cell*, and *Views* drop-down list boxes that appear in the Spectre EMIR / Voltus-Fi XL Analysis Setup form when the *TMI Selfheating* check box is selected.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("spectre.envOpts" "emirLCVDisable" 'boolean t)
```

To set this variable in `.cdsenv` file, add:

```
spectre.envOpts emirLCVDisable 'boolean t
```

Variable Type	Boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	*None*

emirSumList

Defines the default values for the options in the *EMIR Analysis Setup* form.

For a list of values and their options that can be set using this environment variable, see the *EMIR Analysis* section in *Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "emirSumList" 'string "")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts emirSumList string ""
```

Variable Type	string
Default Value	`""`
Acceptable Values	A list of `(option value)` pair with each option separated by space
GUI Equivalent	--

emTechFile

Specifies the technology file containing the EM current limits per layer or via.

To set this environment variable in the `.cdsinit` file or CIW, use the following call:

```
envSetVal("spectre.envOpts" "emTechFile" 'string "")
```

To set this environment variable in the `.cdsenv` file, use the following call:

```
spectre.envOpts emirSumList string ""
```

Variable Type	string
Default Value	`""`
Acceptable Values	Filename containing the EM current limits per layer. The formats supported are `emdatafile`, `qrctechfile`, and `ictfile`.
GUI Equivalent	*None*

enableArclength

When this variable is set to true, the homotopy convergence option is visible, else this is not visible.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "enableArclength" 'boolean t)`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

fastViewOption

Enables the fast waveform viewing format for PSF output.

Using the PSF output in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

Note: Use this environment variable if the value of the simOutputFormat environment variable is set to `psf`, `psf with floats`, or `psfxl`.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.outputs" "fastViewOption" 'boolean t)`

To set it in `.cdsenv`, add:
`spectre.outputs fastViewOption 'boolean t`

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	*Outputs – Save All – Use Fast Viewing Extensions*

finalTimeOp

If this variable to `t`, the info statements for final operating point are generated in the control file, `amsControlSpectre.scs`, and the related data is generated in the psf file, `finalTimeOP.info`. If you do not want the results to be saved, set this variable to `nil`. If this variable is set to `nil`, no final operating point data is generated.

Note: This environment variable is supported for both Spectre and AMS simulators. Replace `spectre.outputs` with `ams.outputs` in the syntax given below to set this variable for AMS simulator.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("spectre.tranOpts" " finalTimeOp" 'boolean t)`

To set it in the .cdsenv file, add:
`spectre.tranOpts finalTimeOp 'boolean t`

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t, nil`
GUI Equivalent	*Virtuoso Analog Design Environment – Transient Options – Save Final Op Pt*

firstRun

Set this variable to false if you do not want the *Welcome to Spectre* window to pop up when you run a simulation.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "firstRun" 'boolean "nil")`

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	--

ignorePortOrderMismatch

When set to t, the netlister will not generate any warning when a portOrder mismatch occurs.

When set to the default value nil, any portOrder mismatch will result in following warning:

`"Mismatch was found between the terminals in the cellView and those on the pin order property on the schematic, or on the termOrder`

property on the CDF. The internal default order is being used. Eliminate the mismatch if any of the above properties must be used for netlisting."

Variable Type	boolean
Default Value	nil
Acceptable Values	t/nil
GUI Equivalent	--

infoOptions

Loads the specified parameters in the *Save circuit information analysis* table in the Save Options form, along with the specified values of the relared fields that are displayed in the table. You can add multiple parameters to the *Save circuit information analysis* table using this variable. The same parameters and field values are loaded in the form when you click the *Defaults* button in the form.

Note: While specifying the value of this variable, ensure that there is no newline character in between the parameters or the related fields as it may cause errors.

Note: This environment variable is supported for both Spectre and AMS simulators. Replace `spectre.outputs` with `ams.outputs` in the syntax given below to set this variable for AMS simulator.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.outputs" "infoOptions" 'string
"modelParameter;models;rawfile;;;;true element;inst;rawfile;;;;true
outputParameter;output;rawfile;;;;true
designParamVals;parameters;rawfile;;;;true
primitives;primitives;rawfile;;;;true
subckts;subckts;rawfile;;;;true asserts;assert;rawfile;;;;false
extremeinfo;all;logfile;;yes;;false
<Click_To_Add>;none;rawfile;;;;false")
```

To set this variable in the .cdsenv file, add:

```
spectre.outputs infoOptions string
"modelParameter;models;rawfile;;;;true element;inst;rawfile;;;;true
outputParameter;output;rawfile;;;;true
designParamVals;parameters;rawfile;;;;true
primitives;primitives;rawfile;;;;true
subckts;subckts;rawfile;;;;true asserts;assert;rawfile;;;;false
```

```
extremeinfo;all;logfile;;yes;;false
<Click_To_Add>;none;rawfile;;;;false"
```

Variable Type string

Default Value For Spectre:

```
"modelParameter;models;rawfile;;;;true
element;inst;rawfile;;;;true
outputParameter;output;rawfile;;;;true
designParamVals;parameters;rawfile;;;;t
rue
primitives;primitives;rawfile;;;;true
subckts;subckts;rawfile;;;;true
asserts;assert;rawfile;;;;false
extremeinfo;all;logfile;;yes;;false
<Click_To_Add>;none;rawfile;;;;false"
```

For AMS:

```
"modelParameter;models;rawfile;;;;true
element;inst;rawfile;;;;true
outputParameter;output;rawfile;;;;true
designParamVals;parameters;rawfile;;;;f
alse
primitives;primitives;rawfile;;;;false
subckts;subckts;rawfile;;;;false
asserts;assert;rawfile;;;;false
extremeinfo;all;logfile;;yes;;false
<Click_To_Add>;none;rawfile;;;;false"
```

Acceptable Values

List of lists that specify the values of the following fields in the *Save circuit information analysis* table:

- *Name*—Name of the parameter.
- *What*—The parameters that are to be saved. possible values are `none`, `inst`, `models`, `input`, `output`, `nodes`, `all`, `terminals`, `oppoint`, `captab`, `parameters`, `primitives`, `subckts`, `assert`, `allparameters`, `netlist`, `options`, and `dumpall`.
- *Where*—File in which the parameters are to be saved.

Note: Asserts can be saved only in a rawfile. possible values are `nowhere`, `file`, `logfile`, and `rawfile`.

- *File*—When Where is set to file, this field is used to specify the name of the file in which parameters are to be saved.
- *Extremes*—Specifies whether the minimum or maximum values are to be saved. possible values are `no`, `yes`, and `only`.
- *Others*—When the What field is set to captab, the options located at the bottom of the table become active and the values specified in the Sort and Threshold fields are populated in the *Others* field
- *Enabled*—Select this check box to save the corresponding parameter.

GUI Equivalent

Virtuoso Analog Design Environment – Save Options – Save circuit information analysis

liclog

If this variable is set to `t`, the check-in/check-out information of the license is added to the log file.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("spectre.envOpts" "liclog" 'boolean t)`

To set this variable in the `.cdsenv` file, add:
`spectre.envOpts liclog  boolean t`

Variable Type	boolean
Default Value	`nil`
GUI Equivalent	*Setup – Environment – Trace License Check-in/ Check-out*

licQueueSleep

This variable specifies the sleep time between two attempts to check out a license when queuing. Setting the value to a positive number overrides the default sleep time of 30 seconds.The option `'+lqsleep <value>'` is not passed to Spectre unless given a value. If it is not passed to *Spectre*, *Spectre* uses a default value of 30.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "licQueueSleep" 'string "20")`

Variable Type	string
Default Value	""
GUI Equivalent	--

licQueueTimeOut

This variable enables queuing for a license. You have to set the time required to wait for a license (in seconds). The option `'+lqtimeout <value>'` is always passed to the simulator, unless set to `0`. It is passed with a value of `900` or any other value that is specified.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "licQueueTimeOut" 'string "1000")`

To set this variable in the .cdsenv file, add:
```
spectre.envOpts licQueueTimeOut string  "900"
```

Variable Type	string
Default Value	`"900"`(15 min)
GUI Equivalent	*Setup – Environment – License Queue Time Out*

licQueueToken

Registers a token request with the license server and creates a queue for a license.

When set to `t`, Spectre registers the token request with the license server and waits for authorization. Spectre ignores all non-token licenses during the wait time because only token licenses are queued.

To set this variable in the `.cdsinit` file or CIW, use the following call:
```
envSetVal("spectre.envOpts" "licQueueToken" 'boolean t)
```

To set this variable in the `.cdsenv` file, use the following call:
```
spectre.envOpts licQueueToken boolean t
```

Variable Type	Boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	None

includePath

Use this field for relative filenames. The simulator resolves a relative filename by first searching in a directory where the file is located. Subsequently, it searches for the file in each of the directories specified by the include path, from left to right.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "includePath" 'string "./dir1 ../dir2")`

Variable Type	string
Default Value	`""`
Acceptable Values	unix directory names, separated with spaces
GUI Equivalent	*Virtuoso Analog Design Environment* – Simulation Files Setup – *include Path*

modelFiles

Specifies the default model files. It can aslo be used to disable model file.

Note: This variable is supported for both Spectre and AMS simulators. Replace `spectre.envOpts` in the code given below to set this variable for AMS simulator.

To set this variable,

- In the .cdsinit file or CIW, use the call:
  ```
  envSetVal("spectre.envOpts" "modelFiles" 'string "./models/
  model1.scs ./models/model2.scs")
  ```
- In the .cdsenv file, use the call:

  ```
  spectre.envOpts modelFiles 'string "./models/model1.scs ./models/
  model2.scs"
  ```

To disable model files,

- In the .cdsinit file CIW, use the call:

  ```
  envSetVal("spectre.envOpts" "modelFiles" 'string "#;./models/
  model1.scs ./models/model2.scs")
  ```
- In the .cdsenv file, use the call:

  ```
  spectre.envOpts modelFiles 'string "#;./models/model1.scs ./
  models/model2.scs"
  ```

Here, the file `model1.scs` will be disabled.

Variable Type	string
Default Value	""
Acceptable Values	list of paths to model files.
GUI Equivalent	*Virtuoso Analog Design Environment* - Model File Setup

modelParamInfo

This variable sets/resets the *Save model parameters Info* option.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.outputs" "modelParamInfo" 'boolean nil)
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Save model parameters info*

nestlvl

This variable is used to save groups of signals as results and when signals are saved in subcircuits. The nestlvl parameter also specifies how many levels deep into the subcircuit hierarchy you want to save signals.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.outputs" "nestlvl" 'string "2")
```

Variable Type	string
Default Value	""
Acceptable Values	--

GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Set level of subcircuit to output (nestlvl)*

netlistBBox

This variable is used to control the size of the netlist window.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "netlistBBox" 'string "10 10 525 800")`

Variable Type	string
Default Value	"0 0 515 700"
Acceptable Values	window coordinates.
GUI Equivalent	--

nport_default_interp

Specifies the global defaults for `interp`. Possible values for `nport_default_interp` are `spline`, `linear`, `bbspice`, and `auto_switch`. For more information, see Interpolation Method.

- If `nport_default_interp` is set to `auto_switch`, `nport` automatically switches the interpolation method based on the analysis. It chooses *bbspice* for pss shooting Newton analysis, and *linear* for analyses, such as `ac`, `dc`, and `sp`. See `spectre -h nport` for information on how `nport_default_interp` works for your particular version of Spectre.

  All `nport` elements in the netlist that do not have `interp` set will have `interp` set to the value specified in the global option `nport_default_interp`. If an `nport` instance has the `interp` option explicitly specified, the instance option takes priority over the global option.

- When ***linear*** is selected as the interpolation method, linear interpolation is used to get a data point needed in the sample that is not directly in the S-parameter file.

- ***Spline*** uses a cubic spline algorithm. Cubic spline can occasionally introduce errors when there are rapid changes in the transfer functions defined in the S-parameter file near the sample point.

- ***Bbspice*** is used to do the rational fit. Bbspice uses a rational model to represent the S-parameter data.

The default is `"auto_switch"`.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.opts" "nport_default_interp" 'string "")`

Variable Type	string
Default Value	"auto_switch"
Acceptable Values	`auto-switch`, `linear`, `Spline`, `Bbspice`.
GUI Equivalent	--

nportirfiledir

This environment variable controls the location of the cached impulse response files. By default, these files are saved in following directory: `/home/<username>/.cadence/mmsim`. When you specify a directory path with this environment variable or in the *nportirfiledir* field on the GUI, the impulse response files are cached in that directory.

Note: You can specify a shared directory only if all users who can access it have write permissions on it.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.opts" "nportirfiledir" 'string "filePath")
```

Variable Type	string
Default Value	""
Acceptable Values	path of cached impulse response file
GUI Equivalent	*Virtuoso Analog Design Environment* – Simulator Options

outputParamInfo

This variable sets/reset the *Save output Parameters Info* option.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.outputs" "outputParamInfo" 'boolean nil )
```

Variable Type	boolean
Default Value	t
Acceptable Values	t, nil
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Save output parameters info*

osc_accuracy

Specifies the accuracy control level for small-signal analyses, such as PAC, PNoise, PXF, PSTB, and PSP when `osc_version` is set to `dts`. The higher the value, the more iterations the Generalized Minimal Residual Method (GMRES) solver takes to generate more accurate results.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.pspOpts" "osc_accuracy" 'string "0")
```

To set it in .cdsenv file, add:
```
spectre.pspOpts osc_accuracy    string     "0"
```

Variable Type	string
Default Value	""
Acceptable Values	`0`, `1`, `2`, `3`, `4`, `5`

GUI Equivalent

- *Virtuoso Analog Design Environment - Choosing Analyses - PAC Options - osc_accuracy*
- *Virtuoso Analog Design Environment - Choosing Analyses - PNoise Options - osc_accuracy*
- *Virtuoso Analog Design Environment - Choosing Analyses - PXF Options - osc_accuracy*
- *Virtuoso Analog Design Environment - Choosing Analyses - PSTB Options - osc_accuracy*
- *Virtuoso Analog Design Environment - Choosing Analyses - PSP Options - osc_accuracy*

Note: This option is available only when the Shooting engine is selected and Oscillator mode is enabled in the Choosing Analyses form for Periodic Steady State analysis.

osc_version

Specifies the method to use for small-signal analysis such as PAC, PNoise, PXF, PSTB, and PSP.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.pspOpts" "osc_version" 'string "dts")
```

To set it in .cdsenv file, add:

```
spectre.pspOpts osc_version     string     "dts"
```

Variable Type	string
Default Value	""

Acceptable Values

- `floquet`: Can be used when the perturbation frequency is close to the oscillating frequency, and when the phase deviation is dominant and the amplitude variation can be ignored. When the perturbation frequency gets higher, the accuracy may be lost.
- `augmented`: Can be used when the oscillators have long time constants, which cannot be accurately predicted by the `floquet` method.
- `dts`: Can be used for any perturbation frequency. It generates results as accurate as the HB engine on all perturbation frequencies.

GUI Equivalent

- *Virtuoso Analog Design Environment - Choosing Analyses - PAC Options - TD Noise Algorithm Version*
- *Virtuoso Analog Design Environment - Choosing Analyses - PNoise Options - TD Noise Algorithm Version*
- *Virtuoso Analog Design Environment - Choosing Analyses - PXF Options - TD Noise Algorithm Version*
- *Virtuoso Analog Design Environment - Choosing Analyses - PSTB Options - TD Noise Algorithm Version*
- *Virtuoso Analog Design Environment - Choosing Analyses - PSP Options - TD Noise Algorithm Version*

Note: This option is available only when the Shooting engine is selected and Oscillator mode is enabled in the Choosing Analyses form for Periodic Steady State analysis.

paramRangeCheckFile

Specifies the path to a file that contains the correct ranges for component parameters. If this file is present, the simulator checks the values of all component parameters in the circuit against the specified parameter range checking file and prints a warning if any parameter value is out of range.

Note: This variable is supported for both Spectre and AMS simulators. For AMS it is supported only when `Spectre` is selected as the solver. Replace `spectre.envOpts` by `ams.envOpts` in the code given below to set this variable for AMS simulator.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("spectre.envOpts" "paramRangeCheckFile" 'string "./
param.file")
```

To set this environment variable in the `.cdsenv` file, use the following call:
```
spectre.envOpts paramRangeCheckFile 'string "./param.file"
```

Variable Type	string
Default Value	""
Acceptable Values	path of the file
GUI Equivalent	*Virtuoso Analog Design Environment* – Environment Options – *Parameter Range Checking File*

preserveSubcktTermNamesByOrder

Enables the generation of save statements with the instance ports in name format instead of the default index format.

To set this variable in the `.cdsinit` file or CIW, use the call:
```
envSetVal("spectre.envOpts" "preserveSubcktTermNamesByOrder"
'boolean nil)
```

To set this variable in the `.cdsenv` file, add:
```
spectre.envOpts preserveSubcktNamesByOrder boolean nil
```

Variable Type	boolean
Default Value	`nil`
GUI Equivalent	*Setup – Environment – Preserve Subckt Terminal by Names*

printComments

Prints the name mapping of the terminals in the netlist as comments.

You can print the mapping of the schematic terminal name with the netlist terminal name for the subcircuits and the mapping of schematic device names with simulator devices names by setting the value of the first toggle to `t`.

For each subcircuit, you can print the connection of each terminal with the net it is connected to by setting the second toggle to `t`.

To set this variable in the `.cdsinit` file or CIW for printing comments for name mapping and subcircuit port connection, use the following call:

```
envSetVal("spectre.envOpts" "printComments" 'toggle '(t t)
```

pwr

This variable is used to select the power signals to output.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.outputs" "pwr" 'string "all")
```

Variable Type	string
Default Value	""
Acceptable Values	none, total, devices, subckts, all
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Select power signals to output (pwr*)

recover

Y runs spectre with the `+recover` option, which restarts the simulation from the checkpoint file, if it exists. N runs spectre with the `-recover` option, which does not restart the simulation, even if a checkpoint file exists.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.envOpts" "recover" 'string "Y")
```

Variable Type	string
Default Value	""
Acceptable Values	Y, N

GUI Equivalent	*Virtuoso Analog Design Environment – Environment Options – Start from Checkpoint File(rec)*

resetSpectreApsSharedOptions

When you change the simulation performance mode from *Spectre* to *APS* or vice versa, this environment variable resets the values of options `reltol`, `vabstol`, and `iabstol` in the Simulator Options form to their default values. These options are common for both simulation performance modes and the default values of these options change according to the specified simulation performance mode. By default, when you change the Simulation Performance Mode, the values of these shared options that you specified do not change.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("asimenv.misc"  "resetSpectreApsSharedOptions" "reltol vabstol iabstol")
```

To set this variable in the `.cdsenv` file, add:

```
asimenv.misc  resetSpectreApsSharedOptions  string  "reltol vabstol iabstol"
```

Variable Type	string
Default Value	nil
Acceptable Value	`"reltol vabstol iabstol"`
GUI Equivalent	--

save

This variable selects signals to be saved.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.outputs” “save” ‘string “all”)
```

Variable Type	string
Default Value	allpub
Acceptable Values	none,selected,lvlpub,allpub,all

GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Select signals to output (save)*

saveahdlvars

If you want to save all the `ahdl` variables belonging to all the `ahdl` instances in the design, set the *saveahdlvars* option to `all` using a Spectre options command. For example:

```
Saveahdl options saveahdlvars=all
```

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.outputs" "saveahdlvars" 'string "all")
```

Variable Type	string
Default Value	""
Acceptable Values	selected, all
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Save AHDL variables (saveahdlvars)*

setEngNotation

Specifies how the instance parameters in the netlist are printed. If set to `t`, the instance parameters in the netlist are printed in pure engineering notation. If set to `nil`, the instance parameters are printed in their default notation.

To set this variable in the .cdsinit file or CIW, use the call:

```
envSetVal("spectre.envOpts" "setEngNotation" 'boolean t)
```

Variable Type	boolean
Default Value	nil
Acceptable Values	t, nil
GUI Equivalent	--

simExecName

This variable can be set to point to the path of the desired spectre executable.

Note: When distributing jobs in ADE XL, do not set the `simExecName` cdsenv variable. Instead, set the Distribution method on the Job Policy Setup form to `LBS`, and specify resource strings in the *Hard Resource Req* and *Soft Resource Req* fields.

Caution

Change this variable with caution. It is advisable not to change this variable unless required.

To set this variable in the `.cdsinit` file or CIW, use the call:
`envSetVal("spectre.envOpts" "simExecName" 'string "<path-to-the-spectre-executable>")`

Or

To set this variable in the `.cdsinit` file or CIW, use the call:

`envSetVal("spectre.envOpts" "simExecName" 'string "<name-of-the-spectre-executable>")`

Here, it is assumed that the executable is in the PATH variable.

To set this variable in `.cdsenv`, add:
`spectre.envOpts simExecName 'string "<path-to-the-spectre-executable>"`

Variable Type	string
Default Value	spectre
Acceptable Values	path or name of the spectre executable
GUI Equivalent	--

simOutputFormat

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, Spectre generates an error. The `psf with floats` format is a single-precision format that uses only half the disk space that `psf` uses. Setting the value to `sst2` causes Spectre to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.outputs" "simOutputFormat" 'string "psf")`

To set it in `.cdsenv`, add:
`spectre.outputs simOutputFormat 'string "psf"`

Variable Type	string
Default Value	psfxl
Acceptable Values	`psf`, `psf with floats`, `sst2`, `psfxl`
GUI Equivalent	*Outputs – Save All – Output Format*

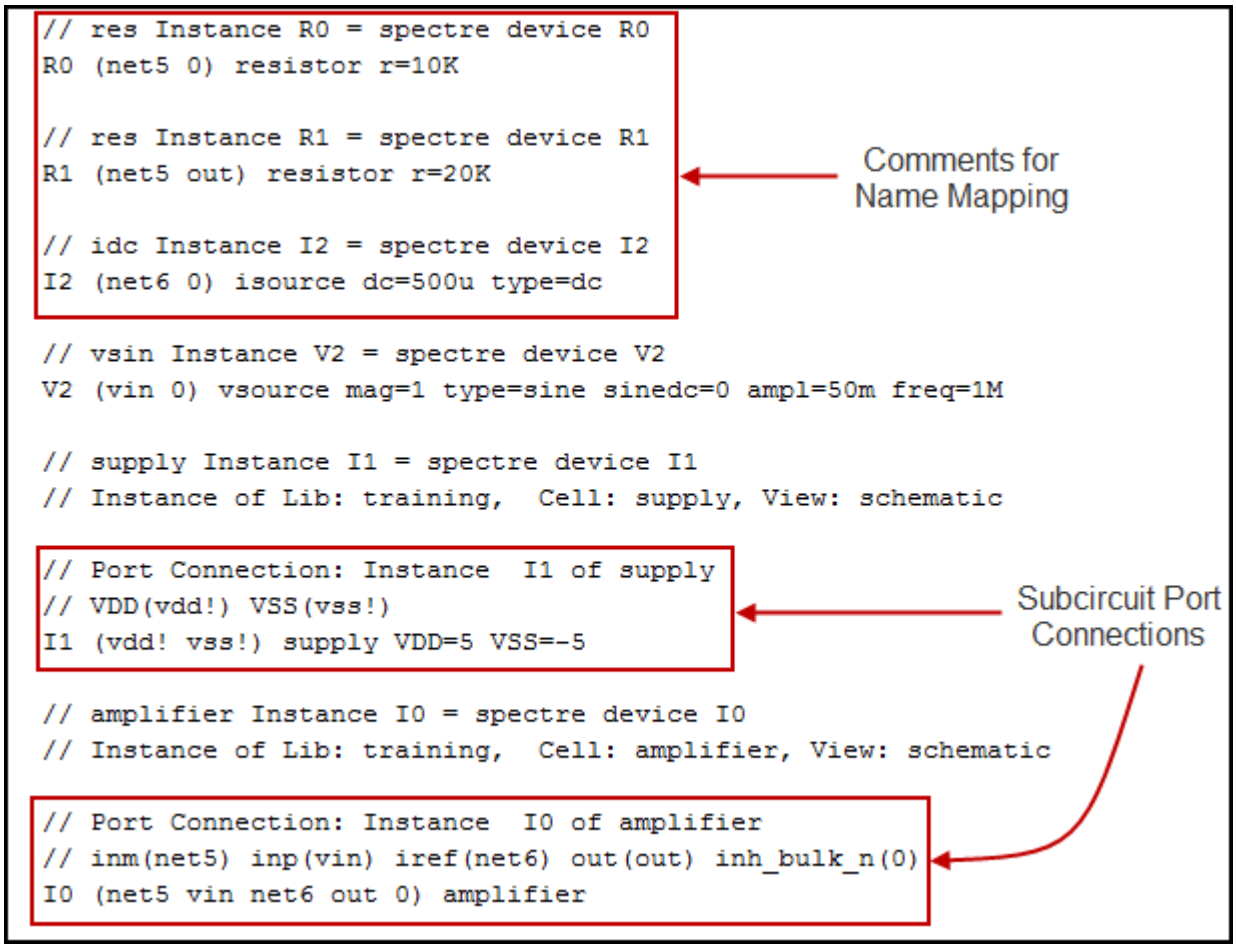
stimulusFile

This variable is used to set the path for stimulus file.

To set this variable in the .cdsinit file or CIW, use the call:
`envSetVal("spectre.envOpts" "stimulusFile" 'string "./file")`

Variable Type	string
Default Value	""
Acceptable Values	unix path
GUI Equivalent	*Virtuoso Analog Design Environment* – Simulation Files Setup – *Stimulus File*

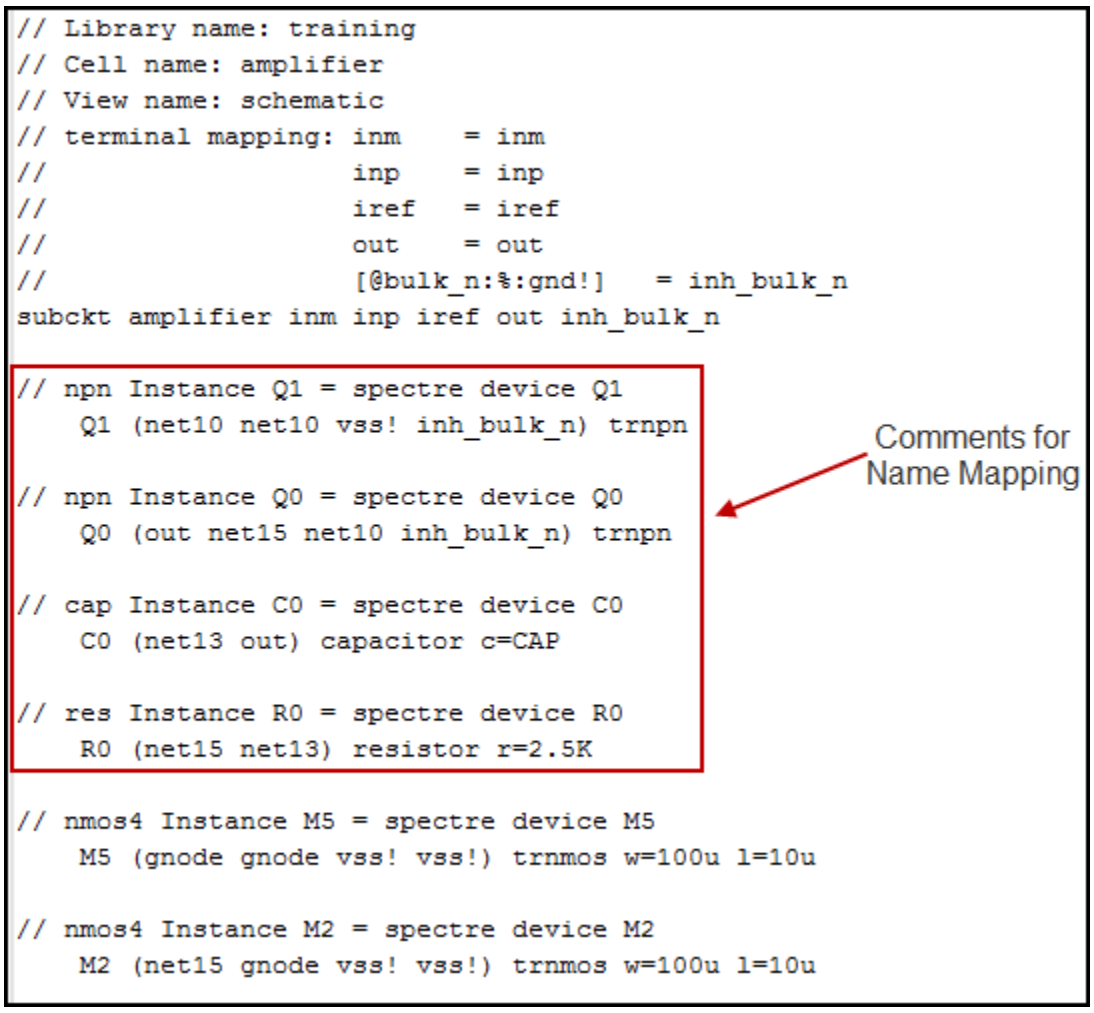
The following figure shows the comments for name mapping and the subcircuit port connections printed in the netlist.



To set this variable in the `.cdsinit` file or CIW for printing comments for only name mapping, use the following call:

```
envSetVal("spectre.envOpts" "printComments" 'toggle '(t nil))
```

The following figure shows the comments for name mapping in the netlist.



To set this variable in the `.cdsenv` file for printing comments for name mapping and subcircuit port connections, use the following call:

```
spectre.envOpts printComments 'toggle '(t t)
```

Variable Type	toggle
Default Value	`nil nil`
Acceptable Values	`t t`, `t nil`, `nil t`, `nil nil`
GUI Equivalent	*Virtuoso Analog Design Environment* – Environment Options – *Print Comments*

stopViewList

Specifies the default stop view list. It is a list of views that identify the stopping view to be netlisted. The same stop view list is displayed in the *Stop View List* field in the Environment Options form.

Note: This variable is supported for both Spectre and AMS simulators. Replace `spectre.envOpts` by `ams.envOpts` in the code given below to set this variable for AMS simulator.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.envOpts" "stopViewList" 'string "spectre verilog")
```

To set this variable in the .cdsenv file, use the call:
```
spectre.envOpts stopViewList 'string "spectre verilog"
```

Variable Type	string
Default Value	`spectre`
Acceptable Values	view names separated with spaces.
GUI Equivalent	*Virtuoso Analog Design Environment* – Environment Options – *Stop View List*

subcktprobelvl

This variable is used to control the calculation of terminal currents for subcircuits. Current probes are added to the terminals of each subcircuit (up to subcktprobelvl deep).

Variable Type	string
Default Value	""
Acceptable Values	--
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Set subcircuit probe level (subcktprobelvl)*

switchViewList

Specifies the default view list. It is a list of the views for ADE to switch when searching for design variables and when netlisting. The same view list is displayed in the *Switch View List* field on the Environment Options form.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.envOpts" "switchViewList" 'string "schematic spectre")
```

To set this variable in the .cdsenv file, use the call:
```
spectre.envOpts switchViewList 'string "schematic spectre"
```

Variable Type	string
Default Value	`spectre cmos_sch cmos.sch schematic veriloga`
Acceptable Values	View names, separated by spaces.
GUI Equivalent	*Virtuoso Analog Design Environment* – Environment Options – *Switch View List.*

tran_severity

This variable changes the default severity of the messages displayed in the simulation log file and the Violations Display form, when device checks are violated and reported for the transient analysis.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.checkOpts" "tran_severity" 'string "None")
```

Variable Type	string
Default Value	Warning
Acceptable Values	None, Error, Warning, Notice, Fatal
GUI Equivalent	*Virtuoso Analog Design Environment – Device Check Specification – Device Checking Options*

uniMode

This variable is used to set the simulation performance modes for Spectre. The available modes are Spectre (default), APS, and XPS MS.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.turboOpts" "uniMode" 'string  "Spectre")
```

Variable Type	string
Default Value	Spectre
Acceptable Values	Spectre, APS, XPS MS
GUI Equivalent	*Virtuoso Analog Design Environment L* – Running a Simulation – Specifying Performance and Post-Layout Optimization Options for the Spectre Simulator

useprobes

This variable is used to set the *Select AC terminal currents (useprobes)* option.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("spectre.outputs" "useprobes" 'string "no")
```

Variable Type	string
Default Value	""
Acceptable Values	yes, no
GUI Equivalent	*Virtuoso Analog Design Environment* – Save Options – *Select AC terminal currents (useprobes)*

OSS

Generating Customized Cellnames

Netlister changes cellname during netlisting in the following cases:

- When cellname contains an illegal character
- When cellname length exceeds maximum simulator supported length
- When multiple instances of cell are present in design with different library or viewname
- When multiple instances of same cell are present with occurrence binding

By default cell names are generated in the following order.

- `Cellname`
- `Cellname_viewname`
- `Libname_cellname_viewname`
- `_sub0`
- `_sub1`
- `_sub2`

After generating a cell name, a check has been made to ensure that the generated cellname is unique, which means it has not been generated previously and the generated cellname is legal. If the cellname is not unique or contains illegal characters, a new cellname is generated as per the order explained above.

For example, if the first cellname is generated as `cellname` and if it is not unique or is illegal, then the next cellname that gets generated is `cellname_viewname`. Similarly, the sequence is followed.

Other than the default cellnames, you can also generate the customized cellnames in the following order.

- `cellname`
- `cellname_viewname`
- `libname_cellname_viewname`
- `customizecellname0`

- customizecellname1
- customizecellname2

To generate these customized names, you need to set the following variables in the .simrc or .cdsinit file:

- hnlGenerateCustomizedCellName
- hnlCustomizedModulePrefix

hnlCustomizedModulePrefix

Set this variable to specify the customized cellname that you want to use for the subcircuit. It contains the value in the following format:

```
hnlCustomizedModulePrefix = myPrefix%L%C%V%N%mypostPreifx
```

If % is followed by L, C, V, N (specified only in uppercase), then L/C/V/N is replaced by lib/cell/view/number. Otherwise, % is treated as a separator and a "_" is inserted. The number is generated by the netlister in a sequential order starting from 0.

It is not necessary to use % as separator. Alternatively, you can also insert an underscore character "_" .

Consider the case where library name is 70WSM, cellname is RES, and viewname=schematic. The table below shows the corresponding cellnames that are generated when you specify a particular prefix:

Prefixes	Generated CellNames
yoursub0	yoursub01
your%sub0	your_sub02
abc%L%C%V%N%XYZ	abc_70WSM_RES_schematic_3_XYZ
abc%L%C%V%NXYZ	abc_70WSM_RES_schematic_4XYZ
abc%L%C%V%N_XYZ	abc_70WSM_RES_schematic_5_XYZ
abc%l%c%v%n_XYZ	abc_l_c_v_n_XYZ6
NPDK%L%C%V%N%NPD	NPDK_70WSM_RES_schematic_2_2PD

If you want that N does not replace with a number in the NPD prefix, use _ instead of %. For example:

`NPDK%L%C%V%N_NPD` will generate `NPDK_70WSM_RES_schematic_2_NPD`.

When the `hnlCustomizedModulePrefix` variable is not defined, the default cellnames are used.

If an invalid name is specified in the `hnlCustomizedModulePrefix` variable, an error message is displayed and netlisting is stopped.

hnlGenerateCustomizedCellName

Set this variable to `true` if you want to generate a cellname of your choice. Before generating a cellname, a check is done to find whether this variable is enabled. If it is enabled, it indicates that the naming convention defined in the `hnlCustomizedModulePrefix` variable is used for generating the cellnames instead of the default names.

If the `hnlGenerateCustomizedCellName` variable is set to `nil` or `hnlCustomizedModulePrefix` is not set, the default cellnames are used.

Ultrasim

fastViewOptionl

Enables the fast waveform viewing format for PSF output.

Using the PSF output in the fast waveform viewing format, Virtuoso Visualization and Analysis XL can render extremely large datasets (where signals have a large number of data points, for example 10 million) within seconds.

Note: Use this environment variable if the value of the wf_format environment variable is set to `psf`.

To set this variable in the .cdsinit file or CIW, use the call:
```
envSetVal("UltraSim.outputs" "fastViewOption" 'boolean t)
```

To set it in `.cdsenv`, add:
```
UltraSim.outputs fastViewOption 'boolean t
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t`, `nil`
GUI Equivalent	*Outputs – Save All – Use Fast Viewing Extensions*

useOtherOutputFormat

When this variable is set to t, PSF, PSFXL,SST2, FSDB, and WDF is displayed for Output Format field of Simulator Options form. If set to nil, PSF and SST2 is displayed as Output Format.To set this variable in the .cdsinit file , use the call:

```
envSetVal("UltraSim.opts" "useOtherOutputFormat" 'boolean t)
```

Variable Type	boolean
Default Value	`nil`
Acceptable Values	`t/nil`

For a comprehensive list of UltraSim variables, refer to *Appendix A* of the *Virtuoso AMS Environment User Guide*.

wf_format

Use this variable to specify the format of output results. If you specify values other than those supported by ADE, UltraSim generates an error. Setting the value to `sst2` causes UltraSim to generate the output for transient analyses in the SignalScan Turbo 2 (SST2) format. Non-transient data cannot be generated in the SST2 format and is generated in parameter storage format (PSF) format.

To set this variable in the `.cdsinit` file or CIW, use the call:

```
envSetVal("UltraSim.outputs" "wf_format" 'string "psf")
```

To set it in the .cdsenv file, add:

```
UltraSim.outputs wf_format 'string "psf"
```

Variable Type	string
Default Value	`sst2`
Acceptable Values	`psf,psfxl,sst2`
GUI Equivalent	*Outputs – Save All – Output Format*

B

Environment Variables for Spectre Simulator Options Form

This section describes the environment variables that you can set in the `.cdsenv` file to specify default values for Spectre simulator options or analysis options.

Important

The default values specified for the variables in this document indicate the default values specified for these variables in the $CDSHOME/tools/dfII/etc/tools/spectre/.cdsenv file.

If the default value of a variable is specified as an empty string value(`""`) or as `"default"`, Spectre uses the in-built default value for that variable.

For example, the default value for `spectre.hb noiseout` is `""`. If you do not modify this value in Virtuoso, while running the simulation, Spectre internally sets it to the in-built default value `"usb"`.

For more details, refer *Spectre Circuit Simulator Reference Guide*.

You can set or customize the Spectre simulation options using the *Simulator Options* form in ADE. This appendix provides details of all available options on the following tabs of the *Simulator Options* form and their respective environment variables:

- Simulator Options - Main Tab
- Simulator Options - Algorithm Tab
- Simulator Options - Component Tab
- Simulator Options - Check Tab
- Simulator Options - Annotation Tab
- Simulator Options - Miscellaneous Tab

spectre.opts

Simulator Options - Main Tab

- generalnoiseinst
- generalnoiseinstonoff
- highvoltage
- iabstol
- multithread
- noiseOffType
- noiseOnType
- nthreads
- reltol
- residualtol
- temp
- tempeffects
- tnom
- vabstol

Simulator Options - Algorithm Tab

- convdbg
- dc_pivot_check
- dptran_gmethod
- gmethod
- gmin
- gmin_check
- gmindc
- homotopy
- icpriority
- ilimit
- nonconv_topnum
- pivabs
- pivotdc
- pivrel
- preorder
- rabsclamp
- rabsshort
- rebuild_matrix
- rforce
- rthresh
- try_fast_op

Simulator Options - Component Tab

- approx
- auto_minductor
- vth_vdsmin
- ivthl
- ivthn
- ivthp
- ivthw
- imacromodels
- maxrsd
- nport_default_passivity

- nportcompress
- nportcompressfiledir
- nportirfiledir
- nportirreuse
- nportunusedportgmin
- nportunusedportrmin
- minr
- scale
- scalefactor
- scalem
- tmevthmod
- vthmod

Simulator Options - Check Tab

- checklimitdest
- checklimitfile
- checklimitskipfile
- checklimitskipsubs
- diagnose
- iccheck
- gnshorts
- redefinedparams
- opptcheck
- topcheck

Simulator Options - Annotation Tab

- audit
- cols
- colslog
- debug
- digits
- error
- info
- inventory
- maxnotes
- maxnotestologfile
- maxwarns
- maxwarnstologfile
- notation
- note
- narrate
- printstep
- simstat
- title
- warn

Simulator Options - Miscellaneous Tab

- additionalArgs
- ahdllint
- ahdllint_maxwarn
- ahdllint_on
- flow
- quantities
- sensbinparam
- sensfile

- sensfileonly
- senstype
- sensformat
- value1

Simulator Options - Main

generalnoiseinst

`spectre.opts generalnoiseinst string` *`any_string_value`*

Description

List of instances to be considered for noise contribution.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*Instance List:*

generalnoiseinstonoff

`spectre.opts generalnoiseinstonoff string` *`any_string_value`*

Description

Specify whether to enable ("on") or disable ("off") the noise contribution for the instance list given in the `generalnoiseinst` cdsenv.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*Noise Contribution*

highvoltage

`spectre.opts highvoltage string` *`any_string_value`*

Description

Enables optimized Spectre settings for high voltage designs including voltage, and current binning, excluding VerilogA and dangling nodes from convergence checks, and optimized large capacitance handling. Possible values are 'no' and 'yes'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*highvoltage*

iabstol

`spectre.opts iabstol string` *`any_string_value`*

Description

Convergence criterion for absolute current tolerance.

The default is `1e-12`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*iabstol*

multithread

`spectre.opts multithread string` *`any_string_value`*

Description

Enable or disable multithreading capability. When multithreading is enabled but the number of threads (nThreads) is not specified, Spectre will automatically detect the number of processors and select the proper number of threads to use.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*multithread*

noiseOffType

```
spectre.opts noiseOffType toggle { thermal | flicker | shot | ign | all }
```

Description

Disable specific noise sources for the list of instances given with the cdsenv `generalnoiseinst`.

The default is `(nil nil nil nil nil)`.

Example:

```
envSetVal("spectre.opts" "noiseOffType" 'toggle '(t nil t nil nil))
```

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*noiseoff_type*

noiseOnType

```
spectre.opts noiseOnType toggle { thermal | flicker | shot | ign | all }
```

Description

Enable specific noise sources for the list of instances given with the cdsenv `generalnoiseinst`.

The default is `(nil nil nil nil nil)`.

Example:

```
envSetVal("spectre.opts" "noiseOnType" 'toggle '(t nil t nil nil))
```

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*noiseon_type*

nthreads

`spectre.opts nthreads string` *`any_string_value`*

Description

Specifies the number of threads for multithreading.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*nthreads*

reltol

`spectre.opts reltol string` *`any_string_value`*

Description

Relative convergence criterion.

The default is `1.00E-03`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*reltol*

residualtol

`spectre.opts residualtol string` *`any_string_value`*

Description

Tolerance ratio for residual (multiplies reltol).

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*residualtol*

temp

`spectre.opts temp string` *`any_string_value`*

Description

Specifies the temperature.

The default is `"27"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*temp*

tempeffects

`spectre.opts tempeffects string` *`any_string_value`*

Description

Temperature effect selector. If tempeffect = vt, only thermal voltage varies with temperature; if tempeffect = tc, parameters that start with tc are active and thermal voltage is dependent on temperature; and if tempeffect = all, all built-in temperature models are enabled.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*tempeffects*

tnom

`spectre.opts tnom string` *`any_string_value`*

Description

Temperature measurement of the default component parameter

The default is `"27"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*tnom*

vabstol

`spectre.opts vabstol string` *`any_string_value`*

Description

Convergence criterion for absolute voltage tolerance.

The default is `1.00E-06`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Main*
Field	*vabstol*

Simulator Options - Algorithm

convdbg

`spectre.opts convdbg string` *`any_string_value`*

Description

Specifies the option to diagnose convergence issues and identify problem areas. Possible values are "none", "status" and "detailed".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*convdbg*

dc_pivot_check

`spectre.opts dc_pivot_check string` *`any_string_value`*

Description

During DC analysis, the numeric pivoting is only performed when bad pivot is detected.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*dc_pivot_check*

dptran_gmethod

`spectre.opts dptran_gmethod string` *`any_string_value`*

Description

Stamp gdev, gnode, or both in the dptran (homotopy) methods. Possible values are 'dev', 'node' and 'both'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*dptran_gmethod*

gmethod

`spectre.opts gmethod string` *`any_string_value`*

Description

Stamp gdev, gnode, or both in the homotopy methods (other than dptran).

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*gmethod*

gmin

`spectre.opts gmin string` *`any_string_value`*

Description

`gmin` (conductance) is added to each nonlinear branch of the device to prevent simulation non-convergence. Large `gmin` impacts accuracy of current probe, small `gmin` may cause circuit convergence issue. For circuit that is sensitive to leakage current, it is recommended to set `gmin` to a small value or zero.

The default is `"1e-12"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*gmin*

gmin_check

```
spectre.opts gmin_check string any_string_value
```

Description

Specifies that effect of `gmin` should be reported if significant. Possible values are 'no', 'max_v_only', 'max_only' and 'all'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options- Algorithm*
Field	*gmin_check*

gmindc

```
spectre.opts gmindc string any_string_value
```

Description

Minimum conductance across each non-linear device in DC analysis. If `gmindc` is not specified, the value of `gmindc` will be equal to `gmin`.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*gmindc*

homotopy

`spectre.opts homotopy string` *`any_string_value`*

Description

Method used when there is no convergence on initial attempt of DC analysis. Possibly values are 'none', 'gmin', 'source', 'dptran', 'ptran', 'all'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*homotopy*

icpriority

`spectre.opts icpriority string` *`any_string_value`*

Description

Set the ic priority order. If set to netlist, the order from lowest to highest is readNS, netlist NS, readIC, netlist IC. If set to file, the order from lowest to highest is netlist NS, readNS, netlist IC readIC.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*icpriority*

ilimit

`spectre.opts limit string` *`any_string_value`*

Description

Limiting algorithms to aid DC convergence. Possibly values are 'delta', 'log' and 'dev'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options- Algorithm*
Field	*limit*

nonconv_topnum

`spectre.opts nonconv_topnum string` *`any_string_value`*

Description

Top number of non-convergence nodes to be printed.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*nonconv_topnum*

pivabs

`spectre.opts pivabs string` *`any_string_value`*

Description

Specifies the absolute pivot threshold.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*pivabs*

pivotdc

```
spectre.opts pivotdc string any_string_value
```

Description

Use numeric pivoting on every iteration of DC analysis. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*pivotdc*

pivrel

```
spectre.opts pivrel string any_string_value
```

Description

Specifies the relative pivot threshold.

The default is `1.00E-03`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*pivrel*

preorder

```
spectre.opts preorder string any_string_value
```

Description

Try this option when simulation runs out of memory or if the simulation is unreasonably slow for the size of your design. It controls the amount of matrix pre-ordering that is done and may lead to much fewer matrix fill-ins in some cases. Known cases include designs with very large number of small resistors and large number of behavioral instances containing voltage based equations. Possible values are "partial" and "full".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*preorder*

rabsclamp

```
spectre.opts rabsclamp string any_string_value
```

Description

When rabsclamp=value is specified, all instance resistors with absolute R<value are clamped to value.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*rabsclamp*

rabsshort

`spectre.opts rabsshort string` *`any_string_value`*

Description

When this option is set, all fixed value resistors with absolute value of R<=rabsshort are shorted.

Default value is 0 for Spectre, and 1m for APS. Rabsshort can additionally be applied to variable resistors using the option 'short_cut_var_elem=yes'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*rabsshort*

rebuild_matrix

`spectre.opts rebuild_matrix string` *`any_string_value`*

Description

If set to yes, rebuild circuit matrix at the beginning of ac, dc, dcmatch, montecarlo, pz, stb, sweep, tdr, and tran analyses. This is to ensure consistent matrix ordering at the beginning of the analyses for consistent results. Notice that rebuild circuit matrix can result in performance overhead. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*rebuild_matrix*

rforce

```
spectre.opts rforce string any_string_value
```

Description

Resistance used when forcing nodesets and node-based initial conditions.

The default is `"1"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*rforce*

rthresh

```
spectre.opts rthresh string any_string_value
```

Description

All instance resistors that have resistance smaller than global rthresh will use resistance form, unless their instance parameter or model parameter overwrites it. Note that resistance form of any resistor is set at the beginning of simulation and cannot be changed later, so altering the value of rthresh is of no use. You will have to start a new run if you want a different rthresh for your circuit

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*rthresh*

try_fast_op

```
spectre.opts try_fast_op string any_string_value
```

Description

Speed up the DC solution. For hard-to-converge designs, this feature fails and other methods are applied. In corner cases, this feature may have negative effects. If the DC analysis is unusually slow, the memory usage of the processes keeps increasing, or if DC analysis gets stuck even before homotopy methods start, set this option to "no".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Algorithm*
Field	*try_fast_op*

Simulator Options - Component

approx

`spectre.opts approx string` *`any_string_value`*

Description

Specifies the use of approximate models. Difference between approximate and exact models. Possible values are 'yes' and 'no'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*approx*

auto_minductor

`spectre.opts auto_minductor string` *`any_string_value`*

Description

Automatic insertion of missing mutual inductor coupling. Possible values are 'yes' and 'no'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*auto_minductor*

vth_vdsmin

`spectre.opts ivth_vdsmin string` *`any_string_value`*

Description

Minimum Vds in constant current Vth calculation.

The default is "".

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*ivth_vdsmin*

ivthl

```
spectre.opts ivthl string any_string_value
```

Description

Length offset for constant current Vth.

The default is "".

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*ivthl*

ivthn

```
spectre.opts ivthn string any_string_value
```

Description

NMOS Vth current parameter.

The default is "".

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*ivthn*

ivthp

```
spectre.opts ivthp string any_string_value
```

Description

PMOS Vth current parameter.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*ivthp*

ivthw

```
spectre.opts ivthw string any_string_value
```

Description

Width offset for constant current Vth.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*ivthw*

imacromodels

```
spectre.opts macromodels string any_string_value
```

Description

Determine whether circuit contains macromodels; at times, setting this parameter to yes helps improve performance

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*macromodels*

maxrsd

`spectre.opts maxrsd string` *`any_string_value`*

Description

Use approximation for drain/source parasitic resistors which are less then maxrsd. Applies to bsim3v3, bsim4 mosfet models

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*maxrsd*

nport_default_passivity

`spectre.opts nport_default_passivity string` *`any_string_value`*

Description

Check and enforce passivity of S-parameter for all nport instances. Default is `"disable"`, which means this global option has no effect. If set to a value other than `"disable"`, all nport elements in the netlist without a value for 'passivity' explicitly set, will have their 'passivity' argument set to the same value as specified in this global option. If an nport instance already has the 'passivity' option specified, the instance option will take priority if both are present.

Possible values are "no", "check", "enforce", "fit_weak_enforce", "fit_enforce" and "disable".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nport_default_passivity*

nportcompress

`spectre.opts nportcompress string` *`any_string_value`*

Description

Nport compression improves the efficiency of S-parameter simulation of large nport files when a certain percentage of the ports is unused, i.e., open or short circuited. Nport compression does not impact simulation accuracy. This option turns off compression if set to no and attempts to force compression if set to yes. If left unspecified, compression is on if N>=10 and the ratio of used ports is less than or equal to 0.8. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nportcompress*

nportcompressfiledir

`spectre.opts nportcompressfiledir string` *`any_string_value`*

Description

The directory where the compressed nport S-parameter file is written to. If unspecified, it is stored in outdir.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nportcompressfiledir*

nportirfiledir

`spectre.opts nportirfiledir string` *`any_string_value`*

Description

The directory to which the nport impulse response file will be written. If it is not specified, the file will be written to /home/<username>/.cadence/mmsim/. If a relative path is specified, the path is relative to the current working directory

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nportirfiledir*

nportirreuse

`spectre.opts nportirreuse string` *`any_string_value`*

Description

Reuses impulse responses data for all `nport` instances. If set to "no", disables this feature. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nportirreuse*

nportunusedportgmin

```
spectre.opts nportunusedportgmin string any_string_value
```

Description

Default is 0, which leaves the port open-circuited. A small value loads open-circuited ports with a finite but large resistance. This introduces a small error in the response, but it induces losses which help obtain a passive response.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nportunusedportgmin*

nportunusedportrmin

```
spectre.opts nportunusedportrmin string any_string_value
```

Description

Default is 0, which leaves the port short-circuited. A small value will insert a small resistance in place of short circuited ports. This introduces a small error in the response, but it induces losses which help obtain a passive response.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*nportunusedportrmin*

minr

```
spectre.opts minr string any_string_value
```

Description

All parasitic resistors inside devices less than global minr will be removed. The order of checking devices is the follows: 1. Check if resistors are smaller than local minr. If yes, check if it is a MOSFET or BJT. If it is a MOSFET, drop the resistor, if it is BJT, clamp to the minr value, and give a warning message for both cases. 2. Check global minr, All Parasitic resistors less than global minr are removed and a warning message is issued. 3. If the resistor is not removed and is smaller than 0.001, issue a warning

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*minr*

scale

```
spectre.opts scale string any_string_value
```

Description

Device instance scaling factor

The default is `"1"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*scale*

scalefactor

```
spectre.opts scalefactor string any_string_value
```

Description

ScaleFactor for Device Model Technology Scaling. The options parameter scalefactor enables device model providers to scale device technology independent of the design dimension scaling done by circuit designers. The resulting device instance scaling is defined by 'scale * scalefactor'. If the foundry uses a technology scale factor of 0.9 (scalefactor=0.9), and the circuit designer uses a design scale factor of 1e-6 (scale=1e-6), then the compounded scaling of the device instance dimension is 0.9e-6. Unlike options parameter scale, scalefactor cannot be used as a netlist parameter and cannot be altered or used in sweep statements.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*scalefactor*

scalem

```
spectre.opts scalem string any_string_value
```

Description

Model scaling factor.

The default is `"1"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*scalem*

tmevthmod

```
spectre.opts tmevthmod string any_string_value
```

Description

TSMC constant vth calculation. By default it is not activated. Possible values are "0" and "1".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*tmevthmod*

vthmod

```
spectre.opts vthmod string any_string_value
```

Description

Vth output selector. 'std' outputs model equation Vth; 'vthcc' outputs constant current Vth and may impact simulation performance. Possible values are "std" and "vthcc".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Component*
Field	*vthmod*

Simulator Options - Check

checklimitdest

`spectre.opts checklimitdest string` *`any_string_value`*

Description

Destination(s) where violations are written. Possible values are 'file', 'psf' and 'both'.

The default is `"psf"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*checklimitdest*

checklimitfile

`spectre.opts checklimitfile string` *`any_string_value`*

Description

File to which assert violations are written.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*checklimitfile*

checklimitskipfile

`spectre.opts checklimitskipfile string` *`any_string_value`*

Description

Specifies the file which contains the subcircuit masters or subcircuit master patterns to be skipped in device checking. Patterns can have any wildcard symbols.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*checklimitskipfile*

checklimitskipsubs

`spectre.opts checklimitskipsubs string` *`any_string_value`*

Description

Specifies the array of subcircuit masters or subcircuit master patterns to be skipped in device checking. Patterns can have any wildcard symbols.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*checklimitskipsubs*

diagnose

`spectre.opts diagnose string` *`any_string_value`*

Description

Print additional information that might help diagnose accuracy and convergence problems. Possible values are 'no', 'yes' and 'detailed'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*diagnose*

iccheck

`spectre.opts iccheck string` *`any_string_value`*

Description

Check if nodes with initial conditions have capacitive path to ground or connected to the ground by vsource. IC for such node is treated as nodeset. Possibly values are 'no', 'vsource', 'cap' and 'all'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*iccheck*

gnshorts

`spectre.opts ignshorts string` *`any_string_value`*

Description

Silently ignore shorted components. Possibly values are 'no' and 'yes'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*ignshorts*

redefinedparams

```
spectre.opts redefinedparams string any_string_value
```

Description

Specify whether parameters can be redefined in the netlist. When set to warning or ignore, the simulator allows you to redefine parameters in the netlist. However, it honors only the last definition of the redefined parameter. Depending on the value set, the simulator displays warning messages for the redefined parameters or does not display any message. When set to error, the simulator does not allow you to redefine parameters in the netlist and displays an error message.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*redefinedparams*

opptcheck

```
spectre.opts opptcheck string any_string_value
```

Description

Checks operating point parameters against soft limits. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*opptcheck*

topcheck

```
spectre.opts topcheck string any_string_value
```

Description

Check circuit topology for errors. Possible values are "no", "min", "full", "fixall", "errmin" and "errfull".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Check*
Field	*topcheck*

Simulator Options - Annotation

audit

`spectre.opts audit string` *`any_string_value`*

Description

Print time required by various parts of the simulator. Possible values are 'no', 'brief' or 'detailed'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*audit*

cols

`spectre.opts cols string` *`any_string_value`*

Description

Width of screen in characters.

The default is `"80"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*cols*

colslog

`spectre.opts colslog string` *`any_string_value`*

Description

Width of log-file in characters.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*colslog*

debug

`spectre.opts debug string` *`any_string_value`*

Description

Give debugging messages. Possible values are 'no' and 'yes'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*debug*

digits

`spectre.opts digits string` *`any_string_value`*

Description

Number of digits used when printing numbers.

The default is `"5"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*digits*

error

`spectre.opts error string` *`any_string_value`*

Description

Generate error messages. Possible values are 'no' and 'yes'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*error*

info

`spectre.opts info string` *`any_string_value`*

Description

Give informational messages. Possibly values are 'no' and 'yes'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*info*

inventory

`spectre.opts inventory string` *`any_string_value`*

Description

Print summary of components used. Possibly values are 'brief', 'detailed' and 'no'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*inventory*

maxnotes

`spectre.opts maxnotes string` *`any_string_value`*

Description

Maximum number of times a notice is issued per analysis. Note that this option has no effect on notices issued as part of parsing the netlist. Use the -maxnotes command-line option to control the number of all notices issued

The default is `"5"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*maxnotes*

maxnotestologfile

`spectre.opts maxnotestologfile string` *`any_string_value`*

Description

Maximum number of times a notice message is printed to the log file per analysis. Note that this option has no effect on notices printed as part of parsing the netlist. Use the -maxnotestolog command-line option to control the number of all notices printed to the log file

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*maxnotestologfile*

maxwarns

```
spectre.opts maxwarns string any_string_value
```

Description

Maximum number of times a warning message is issued per analysis. Note that this option has no effect on warnings issued as part of parsing the netlist. Use the -maxwarns command-line option to control the number of all warnings issued

The default is `"5"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*maxwarns*

maxwarnstologfile

```
spectre.opts maxwarnstologfile string any_string_value
```

Description

Maximum number of times a warning message is printed to the log file per analysis. Note that this option has no effect on warnings printed as part of parsing the netlist. Use the -maxwarnstolog command-line option to control the number of all warnings printed to the log file

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*maxwarnstologfile*

notation

`spectre.opts notation string` *`any_string_value`*

Description

The notation to be used to display real numbers to the screen. Possibly values are 'eng', 'sci', 'float'.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*notation*

note

`spectre.opts note string` *`any_string_value`*

Description

Give notice messages

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*note*

narrate

`spectre.opts narrate string` *`any_string_value`*

Description

Narrate the simulation

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*narrate*

printstep

`spectre.opts printstep string` *`any_string_value`*

Description

Enables Spectre to print results by equal step defined in .tran statement.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*printstep*

simstat

`spectre.opts simstat string` *`any_string_value`*

Description

Print simulation phase statistics report. Possible values are "basic" or "detailed".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*print statistics report*

title

`spectre.opts title string` *`any_string_value`*

Description

Specifies the circuit title.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*title*

warn

`spectre.opts warn string` *`any_string_value`*

Description

Display warning messages. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Annotation*
Field	*warn*

Simulator Options - Miscellaneous

additionalArgs

`spectre.opts additionalArgs string` *`any_string_value`*

Description

Specifies a string that adds options that are not supported through the ADE GUI.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*Additional arguments*

ahdllint

`spectre.opts ahdllint string` *`any_string_value`*

Description

Specifies the Spectre command line argument to enable Verilog-A linter check. Possible values are 'warn', 'error', and 'force'. This option is not available in Spectre base. It is available only in APS or XPS.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*Warning type*

ahdllint_maxwarn

`spectre.opts ahdllint_maxwarn string` *`any_string_value`*

Description

Specifies the maximum number of Verilog-A linter warning messages to be reported by the simulator for each message ID.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*Max warning*

ahdllint_on

`spectre.opts ahdllint_on string` *`any_string_value`*

Description

When set to "on", enables the AHDL Linter feature.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*Linter check*

flow

`spectre.opts flow string` *`any_string_value`*

Description

Specifies the default flow quantity.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*flow*

quantities

`spectre.opts quantities string` *`any_string_value`*

Description

Print quantities. If quantities=min, the simulator prints out all defined quantities; if quantities=full, the simulator also prints a list of nodes and their quantities.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*quantities*

sensbinparam

`spectre.opts sensbinparam string` *`any_string_value`*

Description

Sensitivity for binning models. Possible values are "no", "uncorrelated" and "correlated"

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*sensbinparam*

sensfile

`spectre.opts sensfile string` *`any_string_value`*

Description

Output sensitivity data file name.

The default is `"../psf/sens.output"`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*sensfile*

sensfileonly

`spectre.opts sensfileonly string` *`any_string_value`*

Description

Enable or disable raw output of sensitivity results. Possible values are "no" and "yes".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*sensfileonly*

sensformat

`spectre.opts sensformat string` *`any_string_value`*

Description

Format of sensitivity data. Possible values are "tabular" or "list".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*sensformat*

senstype

`spectre.opts senstype string` *`any_string_value`*

Description

Type of sensitivity being calculated. Possible values are "partial" or "normalized".

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*senstype*

value1

`spectre.opts value1 string` *`any_string_value`*

Description

Specifies the default value quantity.

The default is `""`.

GUI Equivalent

Command	*Options - Analog - Simulator Options - Miscellaneous*
Field	*value*

C

Variables for ams.env Files

The Spectre AMS Designer environment creates a temporary `ams.env` file in the netlist directory. The variables and values in `ams.env` files specify the basic behavior of the AMS netlister and AMS Designer. In general, you have no reason to edit this file.

Important

Only the AMS Cell-based netlisting flow creates the temporary `ams.env` file in the netlist directory. The variables and values in the `ams.env` files specify the basic behavior of the AMS Cell-based netlisting flow only. The AMS UNL and AMS OSSN flows ignore these settings and the `ams.env` files.

This appendix contains the following sections:

- List of ams.env Variables on page 208
- Detailed Descriptions of ams.env Variables on page 214

List of ams.env Variables

The variables that you can use in `ams.env` are all included in the default `ams.env` file. In each entry of the `ams.env` file, the first column is the application, the second column is the variable, the third column is the data type, and the fourth column contains the value to be used. For additional information about individual variables, see "Detailed Descriptions of ams.env Variables" on page 214.

The default `ams.env` file contains the following entries.

```
amsDirect         amsCompMode              boolean   nil
amsDirect         amsDefinitionViews       string    ""
amsDirect         amsExcludeParams         string    ""
amsDirect         amsExpScalingFactor      cyclic    "no"
amsDirect         amsLSB_MSB               boolean   nil
amsDirect         amsMaxErrors             int       50
amsDirect         amsScalarInstances       boolean   t
amsDirect         amsVerbose               boolean   nil
amsDirect         artistStateDirectory     string    "~/.artist_states"
amsDirect         confirmADEStateImport    boolean   t
amsDirect         defaultRunDir            string    ""
amsDirect         hdlVarFile               string    ""
amsDirect         implicitTmpDir           string    ""
amsDirect         includeInstCdfParams     boolean   nil
amsDirect         initFile                 string    ""
amsDirect         logFileName              string    "ams_direct.log"
amsDirect         modifyParamScope         cyclic    "no"
amsDirect         netlistToRunDir          boolean   nil
amsDirect         useRunDirNetlistsOnly    boolean   t
amsDirect         useEffectiveCDF          boolean   nil
amsDirect         simRunDirLoc             string    ""
amsDirect.prep    allowUndefParams         boolean   t
amsDirect.prep    analogControlFile        string    ""
amsDirect.prep    cdsGlobalsLib            string    ""
amsDirect.prep    cdsGlobalsView           string    ""
amsDirect.prep    compileExcludeLibs       string    ""
amsDirect.prep    compileMode              cyclic    "incremental"
amsDirect.prep    connectRulesCell         string    "mixedsignal"
amsDirect.prep    connectRulesCell2        string    "ConnRules_5V_full"
amsDirect.prep    connectRulesLib          string    ""
amsDirect.prep    connectRulesView         string    ""
amsDirect.prep    detailedDisciplineRes    boolean   nil
amsDirect.prep    discipline               string    "logic"
amsDirect.prep    forceGlobalSync          boolean   nil
amsDirect.prep    language                 string    "verilog"
amsDirect.prep    ncelabArguments          string    ""
amsDirect.prep    ncsimArguments           string    ""
amsDirect.prep    ncsimGUI                 boolean   t
amsDirect.prep    ncsimTcl                 boolean   nil
amsDirect.prep    netlistMode              cyclic    "incremental"
amsDirect.prep    runNcelab                boolean   t
amsDirect.prep    runNcsim                 boolean   t
amsDirect.prep    simVisScriptFile         string    ""
amsDirect.prep    timescale                string    "1ns/1ns"
amsDirect.prep    use5xForVHDL             boolean   t
amsDirect.prep    useNcelabNowarn          boolean   t
amsDirect.prep    useNcelabSdfCmdFile      boolean   t
```

```
amsDirect.prep   useNcsimNowarn          boolean   t
amsDirect.prep   wfFilter                boolean   nil
amsDirect.prep   useSimVisScriptFile     boolean   t
amsDirect.prep   vlogGroundSigs          string    "gnd!"
amsDirect.prep   vlogSupply0Sigs         string    ""
amsDirect.prep   vlogSupply1Sigs         string    ""
amsDirect.prep   wfDefaultDatabase       string    "waves"
amsDirect.prep   wfDefInstCSaveAll       boolean   nil
amsDirect.prep   wfDefInstCSaveLvl       int       1
amsDirect.prep   wfDefInstSaveCurrents   boolean   nil
amsDirect.prep   wfDefInstSaveVoltages   boolean   t
amsDirect.prep   wfDefInstVSaveAll       boolean   nil
amsDirect.prep   wfDefInstVSaveLvl       int       1
amsDirect.prep   wfDefInstVSaveObjects   cyclic    "All_data"
amsDirect.prep   wfFilterSpec            cyclic    "none"
amsDirect.prep   ncelabAccess            cyclic    "Read"
amsDirect.prep   ncelabAfile             string    ""
amsDirect.prep   ncelabAnnoSimtime       boolean   nil
amsDirect.prep   ncelabCoverage          boolean   nil
amsDirect.prep   ncelabDelayMode         cyclic    "None"
amsDirect.prep   ncelabDelayType         cyclic    "None"
amsDirect.prep   ncelabDisableenht       boolean   nil
amsDirect.prep   ncelabEpulseFiltering   cyclic    "None"
amsDirect.prep   ncelabEpulseNeg         boolean   nil
amsDirect.prep   ncelabExpand            boolean   nil
amsDirect.prep   ncelabExtendtcheckdatalimit  int  0
amsDirect.prep   ncelabExtendtcheckreferencelimit   int   0
amsDirect.prep   ncelabGenafile          string    ""
amsDirect.prep   ncelabIeee1634          boolean   nil
amsDirect.prep   ncelabInterconnmultisrc boolean   nil
amsDirect.prep   ncelabLibverbose        boolean   nil
amsDirect.prep   ncelabLoadpli1          string    ""
amsDirect.prep   ncelabLoadvpi           string    ""
amsDirect.prep   ncelabLogFileAction     cyclic    "Overwrite log file"
amsDirect.prep   ncelabMaxErrors         int       50
amsDirect.prep   ncelabMessages          boolean   nil
amsDirect.prep   ncelabMixEsc            boolean   nil
amsDirect.prep   ncelabModelFilePaths    string    ""
amsDirect.prep   ncelabmodelIncDirs      string    ""
amsDirect.prep   ncelabNeverwarn         boolean   nil
amsDirect.prep   ncelabNoautosdf         boolean   nil
amsDirect.prep   ncelabNocopyright       boolean   nil
amsDirect.prep   ncelabNoipd             boolean   nil
amsDirect.prep   ncelabNonegtchk         boolean   nil
amsDirect.prep   ncelabNonotifier        boolean   nil
amsDirect.prep   ncelabNosource          boolean   nil
amsDirect.prep   ncelabNostdout          boolean   nil
amsDirect.prep   ncelabNoTchkMsg         boolean   nil
amsDirect.prep   ncelabNoTchkXgen        boolean   nil
amsDirect.prep   ncelabNotimingchecks    boolean   nil
amsDirect.prep   ncelabNovitalaccl       boolean   t
amsDirect.prep   ncelabNoVpdmsg          boolean   nil
amsDirect.prep   ncelabNoVpdXgen         boolean   nil
amsDirect.prep   ncelabNowarn            string    ""
amsDirect.prep   ncelabNtcWarn           boolean   nil
amsDirect.prep   ncelabOmichecklvl       cyclic    "Standard"
amsDirect.prep   ncelabPathpulse         boolean   nil
amsDirect.prep   ncelabPlinooptwarn      boolean   nil
amsDirect.prep   ncelabPlinowarn         boolean   nil
amsDirect.prep   ncelabPresrvResFn       boolean   nil
amsDirect.prep   ncelabPulseE            int       100
```

```
amsDirect.prep   ncelabPulseIntE         int       100
amsDirect.prep   ncelabPulseIntR         int       100
amsDirect.prep   ncelabPulseR            int       100
amsDirect.prep   ncelabRelax             boolean   nil
amsDirect.prep   ncelabSdfCmdFile        string    ""
amsDirect.prep   ncelabSdfNocheckCelltype boolean  nil
amsDirect.prep   ncelabSdfNoHeader       boolean   nil
amsDirect.prep   ncelabSdfNoWarnings     boolean   nil
amsDirect.prep   ncelabSdfprecision      string    ""
amsDirect.prep   ncelabSdfverbose        boolean   nil
amsDirect.prep   ncelabSdfWorstcaseRounding boolean  nil
amsDirect.prep   ncelabsolverInfo        string    "Spectre"
amsDirect.prep   ncelabStatus            boolean   t
amsDirect.prep   ncelabTopLvlGeneric     string    ""
amsDirect.prep   ncelabUpdate            boolean   t
amsDirect.prep   ncelabUseAddArgs        boolean   nil
amsDirect.prep   ncelabUseAfile          boolean   nil
amsDirect.prep   ncelabUseExtendtcheckdatalimit        boolean   nil
amsDirect.prep   ncelabUseExtendtcheckreferencelimit   boolean   nil
amsDirect.prep   ncelabUseGenafile       boolean   nil
amsDirect.prep   ncelabUseGeneric        boolean   nil
amsDirect.prep   ncelabUsePulseE         boolean   nil
amsDirect.prep   ncelabUsePulseIntE      boolean   nil
amsDirect.prep   ncelabUsePulseIntR      boolean   nil
amsDirect.prep   ncelabUsePulseR         boolean   nil
amsDirect.prep   ncelabUseSdfprecision   boolean   nil
amsDirect.prep   ncelabV93               boolean   nil
amsDirect.prep   ncelabVipdelay          cyclic    "Typical"
amsDirect.prep   ncsimEpulseNoMsg        boolean   nil
amsDirect.prep   ncsimExtassertmsg       boolean   nil
amsDirect.prep   ncsimLoadvpi            string    ""
amsDirect.prep   ncsimLogFileAction      cyclic    "Overwrite log file"
amsDirect.prep   ncsimMaxErrors          int       50
amsDirect.prep   ncsimMessages           boolean   nil
amsDirect.prep   ncsimNeverwarn          boolean   nil
amsDirect.prep   ncsimNocifcheck         boolean   nil
amsDirect.prep   ncsimNosource           boolean   nil
amsDirect.prep   ncsimNostdout           boolean   nil
amsDirect.prep   ncsimNowarn             string    ""
amsDirect.prep   ncsimOmichecklvl        cyclic    "None"
amsDirect.prep   ncsimPlinooptwarn       boolean   nil
amsDirect.prep   ncsimPlinowarn          boolean   nil
amsDirect.prep   ncsimProfile            boolean   nil
amsDirect.prep   ncsimProfthread         boolean   nil
amsDirect.prep   ncsimRedmem             boolean   nil
amsDirect.prep   ncsimStatus             boolean   nil
amsDirect.prep   ncsimUnbuffered         boolean   nil
amsDirect.prep   ncsimUpdate             boolean   t
amsDirect.prep   ncsimUseAddArgs         boolean   nil
amsDirect.simcntl   dcop                 boolean   nil
amsDirect.simcntl   paramRangeCheckFile  string    ""
amsDirect.simcntl   scaddlglblopts       string    ""
amsDirect.simcntl   scaddltranopts       string    ""
amsDirect.simcntl   scglobalminr         string    "0.0"
amsDirect.simcntl   scannotate           cyclic    "status"
amsDirect.simcntl   scapprox             boolean   nil
amsDirect.simcntl   scaudit              cyclic    "detailed"
amsDirect.simcntl   sccheckstmt          cyclic    "all"
amsDirect.simcntl   sccmin               string    "0.0"
amsDirect.simcntl   sccompatible         cyclic    "spectre"
amsDirect.simcntl   scdebug              boolean   nil
```

amsDirect.simcntl	scdiagnose	boolean	nil
amsDirect.simcntl	scdigits	int	5
amsDirect.simcntl	scerror	boolean	t
amsDirect.simcntl	scerrpreset	cyclic	"moderate"
amsDirect.simcntl	scfastbreak	boolean	nil
amsDirect.simcntl	scgmin	string	"1e-12"
amsDirect.simcntl	scgmincheck	cyclic	"max_v_only"
amsDirect.simcntl	schomotopy	cyclic	"all"
amsDirect.simcntl	sciabstol	string	"1e-12"
amsDirect.simcntl	scic	cyclic	"all"
amsDirect.simcntl	scicstmt	string	""
amsDirect.simcntl	scignshorts	boolean	nil
amsDirect.simcntl	scinfo	boolean	t
amsDirect.simcntl	scinventory	cyclic	"detailed"
amsDirect.simcntl	sclimit	cyclic	"dev"
amsDirect.simcntl	sclteratio	string	""
amsDirect.simcntl	scmacromod	boolean	nil
amsDirect.simcntl	scmaxiters	int	5
amsDirect.simcntl	scmaxnotes	int	5
amsDirect.simcntl	scmaxnotestologfile	int	5
amsDirect.simcntl	scmaxrsd	string	""
amsDirect.simcntl	scmaxstep	string	""
amsDirect.simcntl	scmaxwarn	int	5
amsDirect.simcntl	scmaxwarntologfile	int	5
amsDirect.simcntl	scmethod	cyclic	"<Default value>"
amsDirect.simcntl	scnarrate	boolean	t
amsDirect.simcntl	scnotation	cyclic	"eng"
amsDirect.simcntl	scnote	boolean	t
amsDirect.simcntl	scopptcheck	boolean	t
amsDirect.simcntl	scpivabs	string	"0.0"
amsDirect.simcntl	scpivotdc	boolean	nil
amsDirect.simcntl	scpivrel	string	"1e-3"
amsDirect.simcntl	scquantities	cyclic	"no"
amsDirect.simcntl	screadic	string	""
amsDirect.simcntl	screadns	string	""
amsDirect.simcntl	screlref	cyclic	"<Default value>"
amsDirect.simcntl	screltol	string	""
amsDirect.simcntl	scrforce	string	"1.0"
amsDirect.simcntl	scscale	int	1
amsDirect.simcntl	scscalem	int	1
amsDirect.simcntl	scalem	string	"1.0"
amsDirect.simcntl	scale	string	"1.0"
amsDirect.simcntl	scmodelevaltype	cyclic	"s"
amsDirect.simcntl	scmosvres	string	"0.05"
amsDirect.simcntl	scscfincfile	string	""
amsDirect.simcntl	scscftimestamp	string	""
amsDirect.simcntl	scscfusefileflag	boolean	nil
amsDirect.simcntl	scskipcount	int	0
amsDirect.simcntl	scskipdc	cyclic	"no"
amsDirect.simcntl	scskipstart	string	"0.0"
amsDirect.simcntl	scskipstop	string	"0.0"
amsDirect.simcntl	scspeed	int	0
amsDirect.simcntl	scspscflag	boolean	nil
amsDirect.simcntl	scstats	boolean	nil
amsDirect.simcntl	scstep	string	""
amsDirect.simcntl	scstop	string	"0.0"
amsDirect.simcntl	scstrobedelay	string	"0.0"
amsDirect.simcntl	scstrobeperiod	string	"0.0"
amsDirect.simcntl	sctemp	string	"27.0"
amsDirect.simcntl	sctempeffects	cyclic	"all"
amsDirect.simcntl	sctitle	string	""

```
amsDirect.simcntl   sctnom               string    "27.0"
amsDirect.simcntl   sctopcheck           cyclic    "full"
amsDirect.simcntl   sctransave           cyclic    "allpub"
amsDirect.simcntl   scusemodeleval       boolean   nil
amsDirect.simcntl   scvabstol            string    "1e-6"
amsDirect.simcntl   scwarn               boolean   t
amsDirect.simcntl   scwrite              string    ""
amsDirect.simcntl   scwritefinal         string    ""
amsDirect.simcntl   simcompat            cyclic    "spectre"
amsDirect.simcntl   start                string    "0.0"
amsDirect.simcntl   useScaddlglblopts    boolean   t
amsDirect.simcntl   useScaddltranopts    boolean   t
amsDirect.simcntl   useScic              boolean   t
amsDirect.simcntl   useScreadic          boolean   t
amsDirect.simcntl   useScreadns          boolean   t
amsDirect.simcntl   useScscfincfile      boolean   t
amsDirect.simcntl   useScwrite           boolean   t
amsDirect.simcntl   useScwritefinal      boolean   t
amsDirect.simcntl   usimAbstoli          string    "1e-12"
amsDirect.simcntl   usimAbstolv          string    "1e-6"
amsDirect.simcntl   usimAddlOptions      string    ""
amsDirect.simcntl   usimAnalog           cyclic    "Default"
amsDirect.simcntl   usimCapFile          string    ""
amsDirect.simcntl   usimCgnd             string    "1e-20"
amsDirect.simcntl   usimCgndr            string    "0"
amsDirect.simcntl   usimDCMethod         cyclic    "Complete DC"
amsDirect.simcntl   usimDcut             boolean   nil
amsDirect.simcntl   usimDcutField        string    ""
amsDirect.simcntl   usimDiodeMethod      cyclic    "Analog table"
amsDirect.simcntl   usimDpfFile          string    ""
amsDirect.simcntl   usimDumpStep         string    ""
amsDirect.simcntl   usimenableNA         boolean   nil
amsDirect.simcntl   usimenablePA         boolean   nil
amsDirect.simcntl   usimenableRA         boolean   nil
amsDirect.simcntl   usimenableTA         boolean   nil
amsDirect.simcntl   usimLshort           string    "0.0"
amsDirect.simcntl   usimLvshort          string    "0.0"
amsDirect.simcntl   usimMaxstep          string    ""
amsDirect.simcntl   usimMaxstepStart     string    "0.0"
amsDirect.simcntl   usimMaxstepStop      string    "0.0"
amsDirect.simcntl   usimMaxstepSubckt    string    ""
amsDirect.simcntl   usimMosMethod        cyclic    "Analog/MS table"
amsDirect.simcntl   usimNALimit          string    "0"
amsDirect.simcntl   usimNAOutputSort     cyclic    "max_vo"
amsDirect.simcntl   usimNASortIs         cyclic    "inc"
amsDirect.simcntl   usimOutputStart      string    "0.0"
amsDirect.simcntl   usimPostl            cyclic    "No RCR"
amsDirect.simcntl   usimRAAgeDomain      cyclic    "loglog"
amsDirect.simcntl   usimRAAgeMethod      cyclic    "interp"
amsDirect.simcntl   usimRAAgeproc        string    ""
amsDirect.simcntl   usimRAAgingTime      string    "10y"
amsDirect.simcntl   usimRADeltaD         string    "0.1"
amsDirect.simcntl   usimRADeltaDToggle   boolean    t
amsDirect.simcntl   usimRAMinAge         string    "0.0"
amsDirect.simcntl   usimRAMode           cyclic    "HCI only"
amsDirect.simcntl   usimRANBTIAgeproc    string    ""
amsDirect.simcntl   usimRcrfmax          string    "1e9"
amsDirect.simcntl   usimRshort           string    "1e-6"
amsDirect.simcntl   usimRvshort          string    "1e-6"
amsDirect.simcntl   usimSimMode          cyclic    "Mixed signal"
amsDirect.simcntl   usimSpeed            int       0
```

```
amsDirect.simcntl   usimSpefFile         string    ""
amsDirect.simcntl   usimSpfFile          string    ""
amsDirect.simcntl   usimTol              string    "0.01"
amsDirect.simcntl   usimTranAddlOptions  string    ""
amsDirect.simcntl   usimUseAddlOptions   boolean   t
amsDirect.simcntl   usimVcdFile          string    ""
amsDirect.simcntl   usimVcdInfoFile      string    ""
amsDirect.simcntl   usimVectorFile       string    ""
amsDirect.simcntl   usimWFAbstoli        string    "1e-12"
amsDirect.simcntl   usimWFAbstolv        string    "1e-6"
amsDirect.simcntl   usimWFFilter         boolean    t
amsDirect.simcntl   usimWFReltol         string    "0.0"
amsDirect.simcntl   usimWFTres           string    "1e-12"
amsDirect.vlog   allowDeviantBuses       cyclic    "no"
amsDirect.vlog   allowIllegalIdentifiers cyclic    "warn"
amsDirect.vlog   allowNameCollisions     cyclic    "warn"
amsDirect.vlog   allowSparseBuses        cyclic    "warn"
amsDirect.vlog   amsEligibleViewTypes    string    "schematic"
amsDirect.vlog   checkAndNetlist         boolean   nil
amsDirect.vlog   checkOnly               boolean   nil
amsDirect.vlog   compileAsAMS            boolean   t
amsDirect.vlog   excludeViewNames        string    ""
amsDirect.vlog   headerText              cyclic    "none"
amsDirect.vlog   ifdefLanguageExtensions boolean   nil
amsDirect.vlog   includeFiles            string    "(disciplines.vams)"
amsDirect.vlog   ncvlogArguments         string    ""
amsDirect.vlog   netlistAfterCdfChange   boolean   nil
amsDirect.vlog   paramDefVals            string    ""
amsDirect.vlog   paramGlobalDefVal       string    "0"
amsDirect.vlog   processViewNames        string    ""
amsDirect.vlog   prohibitCompile         boolean   nil
amsDirect.vlog   templateFile            string    ""
amsDirect.vlog   templateScript          string    ""
amsDirect.vlog   useDefparam             boolean   nil
amsDirect.vlog   useNowarn               boolean   t
amsDirect.vlog   useProcessViewNamesOnly boolean   nil
amsDirect.vlog   verboseUpdate           boolean   t
amsDirect.vlog   checktasks              boolean   nil
amsDirect.vlog   errOutInconsistentMasters    boolean   nil
amsDirect.vlog   ieee1364                boolean   nil
amsDirect.vlog   ignoreIllegalCDFParams  boolean   nil
amsDirect.vlog   noline                  boolean   nil
amsDirect.vlog   incdir                  string    ""
amsDirect.vlog   lexpragma               boolean   nil
amsDirect.vlog   logFileAction           cyclic    "Overwrite log file"
amsDirect.vlog   macro                   string    ""
amsDirect.vlog   markcelldefines         boolean   nil
amsDirect.vlog   netlistUDFAsMacro       boolean   nil
amsDirect.vlog   bindCdsAliasLib         boolean   t
amsDirect.vlog   bindCdsAliasView        boolean   t
amsDirect.vlog   maxErrors               int       50
amsDirect.vlog   messages                boolean   nil
amsDirect.vlog   neverwarn               boolean   nil
amsDirect.vlog   nomempack               boolean   nil
amsDirect.vlog   nopragmawarn            boolean   nil
amsDirect.vlog   nostdout                boolean   nil
amsDirect.vlog   nowarn                  string    ""
amsDirect.vlog   pragma                  boolean   nil
amsDirect.vlog   status                  boolean   nil
amsDirect.vlog   update                  boolean   t
amsDirect.vlog   vloglinedebug           boolean   nil
```

```
amsDirect.vlog   ncvlogUseAddArgs        boolean   nil
amsDirect.vlog   iterInstExpFormat       string    "%b_%i"
amsDirect.vlog   netClashFormat          string    "%b_netclash"
amsDirect.vlog   instClashFormat         string    "%b_instclash"
amsDirect.vlog   aliasInstFormat         string    "ams_alias_inst_%i"
```

Detailed Descriptions of ams.env Variables

Details for these `ams.env` file variables appear alphabetically, by variable name, in the sections that follow.

aliasInstFormat

Specifies the format to be used to create instances of the `cds_alias` module.

Syntax

```
amsDirect.vlog aliasInstFormat string "format"
```

Value

format	All characters, except those listed below, are printed as included in *format*. The following characters have the indicated special meanings. `%i` Index number of the current `cds_alias` instance `%%` Prints the `%` character The default value of *format* is `ams_alias_inst_%i`, which produces names such as `ams_alias_inst_1`, `ams_alias_inst_2`, and so on.

If the resulting name is illegal in Verilog-AMS, the name is mapped. If the mapped name clashes with the name of another object, the name undergoes collision mapping.

Example

```
amsDirect.vlog aliasInstFormat string "cds_alias_%i"
```

Tells AMS netlister to create instance names with a suffixed index number. In this example, instances of the `cds_alias` module are given names like

```
cds_alias_1
cds_alias_2
cds_alias_3
```

allowDeviantBuses

Controls the netlisting of bus specifications when there are conflicting bus ranges. Bus ranges conflict when, in references to the same bus, the indexes sometimes go from smaller to larger and other times go from larger to smaller.

Syntax

```
amsDirect.vlog allowDeviantBuses cyclic "no" | "warn" | "yes
```

Values

`no`	Netlisting halts immediately when the AMS netlister encounters conflicting bus ranges. This is the default. This value corresponds to the *No – Print Errors* value used in the graphical user interface (GUI).
`warn`	Netlisting continues when the AMS netlister encounters conflicting bus ranges if it is possible to create a valid netlist. The AMS netlister tells you how the non-compliant bus data is transformed. The generated netlist is likely to be less readable than one created from compliant bus data. This value corresponds to the *Yes – Print Warnings* value used in the GUI.
`yes`	Netlisting continues when the AMS netlister encounters conflicting bus ranges if it is possible to create a valid netlist. The AMS netlister does not issue a warning. This value corresponds to the *Yes – Silently* value used in the GUI.

Example

Here is an example of conflicting bus ranges:

```
a<0:7>
a<7:6>
a<5:0>
a<2:4>
```

Here is the same example in Verilog-AMS:

```
a[0:7]
{a[7],a[6]}
{a[5],a[4],a[3],a[2],a[1],a[0]}
a{2:4}
```

Using the variable

```
amsDirect.vlog allowDeviantBuses cyclic "yes"
```

tells the AMS netlister to handle conflicting bus ranges whenever possible, without issuing a warning. This example sets the netlisting behavior for data netlisted into the Verilog®-AMS language.

allowIllegalIdentifiers

Controls the netlisting of non-compliant identifiers.

Syntax

```
amsDirect.vlog allowIllegalIdentifiers cyclic "no" | "warn" | "yes
```

Values

`no`	Netlisting halts immediately when the AMS netlister encounters a non-compliant identifier. This value corresponds to the *No – Print Errors* value used in the graphical user interface (GUI).
`warn`	Maps non-compliant identifiers to names that are legal in the target language and issues a warning telling you how the name is mapped. This is the default. This value corresponds to the *Yes – Print Warnings* value used in the GUI.
`yes`	Maps non-compliant identifiers to names that are legal in the target language. The AMS netlister does not issue a warning. This value corresponds to the *Yes – Silently* value used in the GUI.

Description

If you specify `warn` or `yes`, the AMS netlister maps non-compliant identifiers to the target language. However, mapping identifiers results in a less readable netlist.

Identifiers are non-compliant if one or more of the following situations applies:

- Identifiers do not follow the syntax required by the netlist language you plan to use
- Identifiers are reserved words in the netlist language

  For a list of Verilog-AMS reserved words, see the “Verilog-AMS Keywords” appendix in the *Cadence Verilog-AMS Language Reference*.
- Identifiers do not map cleanly to the netlist language
- Identifiers are not unique within the design

Because the determination of non-compliance depends on the target netlist language, it is possible to have identifiers that are compliant for one target language and non-compliant for another. To ensure that identifiers are compliant for every target netlist language, use the following syntax.

```
basic_identifier ::=
      letter {[_] letter_or_digit}

letter_or_digit ::=
      a-z | 0-9
```

For example, the following identifiers are compliant for every target language.

```
an_identifier_name
a_2nd_name
a_name2
```

The following identifiers, because they do not use the suggested syntax, might be non-compliant for some target languages.

```
2identifier     // Should begin with a letter.
My_identifer    // Should not use uppercase letters.
an_identifier_  // Should end with a letter or digit.
a&b             // Should not use characters other than a-z, 0-9, and underscore.
```

allowNameCollisions

Controls the netlisting of names that do not comply with Spectre AMS Designer environment guidelines because they are not unique.

Syntax

```
amsDirect.vlog allowNameCollisions cyclic "no" | "warn" | "yes
```

Values

`no`	Netlisting halts immediately when the AMS netlister encounters a non-unique name. This value corresponds to the *No – Print Errors* value used in the graphical user interface (GUI).
`warn`	Maps non-unique names to system-generated names that are legal in the target language, and issues a warning. This is the default. This value corresponds to the *Yes – Print Warnings* value used in the GUI.
`yes`	Maps non-unique names to system-generated names that are legal in the target language. The AMS netlister does not issue a warning.This value corresponds to the *Yes – Silently* value used in the GUI.

Description

To comply with AMS Designer environment guidelines, each instance, cell, terminal, parameter, and net in your design must have a unique name. If the names of these components are not unique, the AMS netlister acts as shown in the table below.

How Verilog-AMS Handles Non-Unique Identifiers

Objects sharing a name	AMS netlister action
module terminal, cell	No mapping occurs, and netlisting proceeds normally
parameter, module terminal	Netlisting fails
instance terminal, parameter of the same instance	No mapping occurs, and a warning is issued.
parameter, cell	No mapping occurs, and netlisting proceeds normally

How Verilog-AMS Handles Non-Unique Identifiers, *continued*

Objects sharing a name	AMS netlister action
net, parameter	Net identifier maps to *netName*_netclash
net, module terminal	Net identifier maps to *netName*_netclash. (However, no mapping occurs when the net and module terminal are connected to each other.)
net, cell	Net identifier maps to *netName*_netclash
instance, net	Instance identifier maps to *instName*_instclash
instance, parameter	Instance identifier maps to *instName*_instclash
instance, module terminal	Instance identifier maps to *instName*_instclash
instance, cell	Instance identifier maps to *instName*_instclash

allowSparseBuses

Controls the netlisting of sparse buses.

Syntax

```
amsDirect.vlog allowSparseBuses cyclic "no" | "warn" | "yes"
```

Values

`no`	Netlisting halts immediately when the AMS netlister encounters a sparse bus. This value corresponds to the *No – Print Errors* value used in the graphical user interface (GUI).
`warn`	Overdeclares any sparse buses and issues a warning. This is the default. This value corresponds to the *Yes – Print Warnings* value used in the GUI.
`yes`	Overdeclares any sparse buses. The AMS netlister does not issue a warning. This value corresponds to the *Yes – Silently* value used in the GUI.

Description

Sparse buses do not comply with AMS Designer environment guidelines because you must declare buses as a contiguous vector of bits before they are used in Verilog-AMS. If you specify `warn` or `yes`, the AMS netlister overdeclares sparse buses so it can continue netlisting.

Example

Here is an example of a sparse bus:

```
b<5:0:2>
```

which is the same as

```
b<5>, b<3>, b<1>
```

Using the variable

```
amsDirect.vlog allowSparseBuses cyclic "yes"
```

tells the AMS netlister to handle sparse buses whenever possible, without issuing a warning. In this example, the AMS netlister overdeclares this bus in order to continue netlisting:

```
module XXX (.b({b[5],,b{3],,b[1]}), ...);
    input [5:1] b;

    ...
```

allowUndefParams

Controls whether undeclared parameters can be overridden.

Syntax

```
amsDirect.prep allowUndefParams boolean t | nil
```

Values

`t`	The elaborator allows undeclared parameters to be overridden. This is the default.
`nil`	The elaborator stops when it encounters a value override for an undeclared parameter.

Description

By default, the elaborator reports an error and stops when it encounters a value override for an undeclared parameter. Specifying `t` for the `allowUndefParams` variable tells the elaborator to allow undeclared parameters to be overridden.

Example

```
amsDirect.prep allowUndefParams boolean t
```

Tells the elaborator to permit overriding the values of undeclared parameters, such as by using a `defparam` statement or by overriding the value when an instance is declared.

amsCompMode

Controls whether the AMS Designer environment supports certain properties used in legacy VHDL modules. Note, however, that the `amsCompMode` variable is not supported in this release.

Syntax

```
amsDirect amsCompMode boolean t | nil
```

Values

`t`	Specifies that certain properties used in legacy VHDL modules are to be supported by the AMS Designer environment.
`nil`	Specifies that certain properties used in legacy VHDL modules are *not* to be supported by the AMS Designer environment.

Description

The following legacy properties are supported by the AMS Designer environment if the `amsCompMode` variable is set to `t`. If the variable is set to `nil`, the properties are ignored and omitted from the netlist.

- `vhdlAttributeDefList`
- `vhdlComponentDecl`
- `vhdlFormalPortFuncName`
- `vhdlPackageComponents`
- `vhdlPackageNames`

amsDefinitionViews

Specifies a list of views that can be used to determine the vectored terminal range direction and terminal order for cellviews being netlisted. This capability is useful when the cellview being netlisted needs to be netlisted in accordance with another view of the cell, such as the placed master. AMS Designer does not provide a graphical interface for setting this variable.

To use the `amsDefinitionViews` list, the netlister

1. Determines whether there is a `termOrder` property for the cellview being netlisted. If so, that property determines the vectored terminal range direction and terminal order and the `amsDefinitionViews` list has no effect.
2. Determines whether the first listed view exists. If it does, no more views are considered. If the first view does not exist, the search through the list continues until the netlister finds a view that exists or reaches the end of the list.
3. If the identified existing view has a `portOrder` property, uses that information to determine the vectored terminal range direction and terminal order of the cellview being netlisted. If the `portOrder` property does not exist, the netlister checks the view for vectored terminals used in their entirety and uses that ordering. If the ordering is still not determined for one or more terminals, the ordering specified by the `amsLSB_MSB` environment variable is used.
4. If none of the listed views exists, uses the `portOrder` property of the cellview being netlisted (if that cellview has a `portOrder` property) to determine the vectored terminal range direction and terminal order. If the `portOrder` property does not exist, the netlister checks the cellview being netlisted for vectored terminals used in their entirety and uses that ordering. If the ordering is still not determined for one or more terminals, the ordering specified by the `amsLSB_MSB` environment variable is used.

Syntax

```
amsDirect.vlog amsDefinitionViews string "list"
```

Value

list	A string of space-separated views to be consulted for terminal order and vectored terminal range directions. The view names are considered to be in the cellview namespace. Any included views that are created or imported by the CIW must be accompanied by a shadow cellview. The default value is an empty string.

Example

```
amsDirect.vlog amsDefinitionViews string "symbol verilog"
```

amsEligibleViewTypes

Specifies the cellview types that trigger netlisting.

Syntax

```
amsDirect.vlog amsEligibleViewTypes string "list"
```

Value

list	A list of one or more of the following cellview types: `schematic`, `symbolic`, `maskLayout` (extracted view only, based on the last extraction timestamp), and `netlist`. Cellview types must be separated by spaces in the list. If you do not specify a cellview for netlisting (by using the `amsdirect -view` option, for example), the AMS netlister generates netlists for each of the cellview types included in the list. The default for *list* is `schematic`.

Example

```
amsDirect.vlog amsEligibleViewTypes string "schematic symbolic"
```

Tells the AMS netlister to netlist `schematic` and `symbolic` cellviews (unless, for example, a view is specified by using the `amsdirect -view` option). This example sets the netlisting behavior for data netlisted into the Verilog-AMS language.

amsExcludeParams

Lists parameters to be omitted from the netlist.

Syntax

```
amsDirect amsExcludeParams string "list"
```

Value

list	A list of parameters that are not to be netlisted. *list* is a string of space-separated parameter names. The default is an empty string.

Example

```
amsDirect amsExcludeParams string "fix unfix"
```

Tells the AMS netlister not to netlist the parameters `fix` and `unfix` when they are found associated with components in this design.

Note: If a cell has valid information in the `ams` section of the CDF simInfo, the contents of the simInfo are always obeyed, regardless of the value of the `amsExcludeParams` variable. For example, for a cell `mycell`, if `param1` and `param2` are in the *instParameters* field of the simInfo and `param1` is also listed in the `amsExcludeParams` variable, then `amsExcludeParams` has no effect. When `mycell` (or any instance of `mycell`) is netlisted, `param1` is always printed.

You can use the *excludeParameters* simInfo field in conjunction with the `amsExcludeParams` `ams.env` variable and the `amsExcludeParams` CDF parameter to precisely specify parameters at the cell, design, and library levels that are not to be netlisted.

amsExpScalingFactor

Controls the expansion of scaling factors for parameter values.

Syntax

```
amsDirect amsExpScalingFactor cyclic "no" | "dec" | "sci"
```

Values

`no`	Includes scaling factor suffixes in netlists without expanding them. This is the default.
`dec`	Expands scaling factor suffixes in decimal notation.
`sci`	Expands scaling factor suffixes in scientific notation.

Description

Some simulators do not support scaling factors or support only a subset of the scaling factors used in designs. If the simulator you plan to use is one of these simulators, you can use the `amsExpScalingFactor` variable to expand scaling factors so the factors do not appear in netlists.

The following table shows the scaling factor suffixes and the target simulators that support them.

Scaling Factor Suffixes and Target Simulators

Suffix		Scaling Factor (e^x)	AEL	Verilog-AMS	Spectre	SKILL
`Y`	Yotta	10^{24}	See note below.			
`Z`	Zetta	10^{21}	See note below.			
`T`	Tera	10^{12}	yes	yes	yes	yes
`G`	Giga	10^{9}	yes	yes	yes	yes
`M`	Mega	10^{6}	yes	yes	yes	yes
`ME`	Mega	10^{6}	yes			yes
`K`	Kilo	10^{3}	yes	yes	yes	yes

Scaling Factor Suffixes and Target Simulators, *continued*

Suffix		Scaling Factor (e^x)	AEL	Verilog-AMS	Spectre	SKILL
k	kilo	10^3	yes		yes	yes
%	percent	10^{-2}	yes		yes	yes
c	percent	10^{-2}			yes	
m	milli	10^{-3}	yes	yes	yes	yes
u	micro	10^{-6}	yes	yes	yes	yes
n	nano	10^{-9}	yes	yes	yes	yes
p	pico	10^{-12}	yes	yes	yes	yes
f	femto	10^{-15}	yes	yes	yes	yes
a	atto	10^{-18}	yes	yes	yes	yes
z	zepto	10^{-21}	See note below.			
y	yocto	10^{-24}	See note below.			

Note: AMS Designer always expands the `Y`, `Z`, `z`, and `y` scaling factors, using scientific notation, regardless of the value of the `amsExpScalingFactor` variable.

Example

A few examples of expanded scaling factor suffixes are shown below.

5.46T = 5.46e12 = 5,460,000,000,000
5.46G = 5.46e9 = 5,460,000,000
5.46M = 5.46e6 = 5,460,000
5.46K = 5.46e3 = 5,460
5.46% = 5.46e-2 = 0.0546
5.46u = 5.46e-6 = 0.00000546

amsLSB_MSB

Controls the bit order used to netlist a bus when the following conditions are all true:

- The information derived from views listed by the `amsDefinitionViews` environment variable is insufficient to determine the bit order.
- The `portOrder` property of the cellview being netlisted is insufficient to determine the bit order.
- The bus is not used in its entirety anywhere in the cellview being netlisting.

To summarize, the `amsLSB_MSB` variable is used only when the bit order cannot be determined by using the `amsDefinitionViews` variable.

Syntax

```
amsDirect amsLSB_MSB boolean t | nil
```

Values

`t`	Orders the bits as [LSB : MSB] when constructing buses.
`nil`	Orders the bits as [MSB : LSB] when constructing buses. This is the default.

Description

By default, the AMS netlister orders the bits as follows:

[MSB : LSB]

which is most significant bit to least significant bit. Specifying the `t` value for this variable reverses the bit order.

amsMaxErrors

Halts the AMS netlister when it reaches a certain number of errors. If the netlister encounters any design error, it does not produce a netlist.

Syntax

```
amsDirect amsMaxErrors int maxErrors
```

Value

maxErrors	A positive integer. Halts netlisting after this number of errors occur. The default is 50.

Example

```
amsDirect amsMaxErrors int 12
```

Tells the AMS netlister to halt netlisting when it encounters 12 errors.

amsScalarInstances

Controls the netlisting of iterated instances.

Syntax

```
amsDirect amsScalarInstances boolean t | nil
```

Values

`t`	Scalarizes iterated instances. This is the default.
`nil`	Produces an array of instances.

Description

By default, the AMS netlister scalarizes iterated instances. You can use this variable to produce an array of instances in Verilog-AMS netlists instead.

amsVerbose

Controls whether the netlister issues informational messages.

Syntax

```
amsDirect amsVerbose boolean t | nil
```

Values

`t`	Places a checkmark next to the *Print informational messages* field on the *Netlister* pane of the AMS Options window. This tells the netlister to issue verbose messages.
`nil`	Removes the checkmark, indicating that verbose messages are not issued while netlisting. This is the default.

Example

```
amsDirect amsVerbose boolean t
```

Removes the checkmark next to the *Print informational messages* field. As a result, verbose messages are not issued during netlisting.

analogControlFile

Specifies the analog simulation control file to be used.

Syntax

```
amsDirect.prep analogControlFile string "file"
```

Value

file	The analog simulation control file to be used. If *file* is specified with an absolute path, the analog simulation control file is stored at that location. If *file* is specified with a relative path, the path is determined relative to the run directory (not to the current working directory). The default is an empty string, which means that the value that appears in the AMS Run Simulation form is *runDir*/*topLevelCell*.scs.

Example

```
amsDirect.prep analogControlFile string "sch.scs"
```

artistStateDirectory

Specifies the directory used to seed the *From ADE state directory* field in the Import from ADE State form. The specified directory is expected to be the top level of the saved states directory structure.

Syntax

```
amsDirect artistStateDirectory string "directory"
```

Value

directory	The path and directory to be used to seed the form. The default is `~/.artist_states`, which is the default directory used by the Analog Design Environment (ADE) for saved states.

Example

```
amsDirect artistStateDirectory string "~/.mystatesdir"
```

bindCdsAliasLib

Adds the `library_binding = "basic"` attribute to instances of the `cds_alias` module that the AMS netlister adds to netlists.

This attribute specifies the library binding for instances of the `cds_alias` module. This specification is necessary when the `basic` library is not included in the Virtuoso Hierarchy Editor *Library List*. Regardless of the setting of the `bindCdsAliasLib` variable, the `basic` library, which contains the `cds_alias` module, must be defined in the `cds.lib` file.

Syntax

```
amsDirect.vlog bindCdsAliasLib boolean t | nil
```

Values

`t`	Adds the `library_binding = "basic"` attribute to automatically inserted instances of the `cds_alias` module. This is the default.
`nil`	Does not add the `library_binding` attribute to instances of the `cds_alias` module.

Examples

- The variable

  ```
  amsDirect.vlog bindCdsAliasLib boolean t
  ```

  tells the AMS netlister to add the `library_binding` attribute to automatically inserted instances of the `cds_alias` module, producing a statement similar to the following:

  ```
  cds_alias #(.width(1)) (* integer library_binding = "basic"; *)
      ams_alias_inst_0 (net015, net014[0]);
  ```

- The variables

  ```
  amsDirect.vlog bindCdsAliasLib boolean t
  amsDirect.vlog bindCdsAliasView boolean t
  ```

  tell the AMS netlister to add the `library_binding` and `view_binding` attributes to automatically inserted instances of the `cds_alias` module, producing a statement similar to the following:

  ```
  cds_alias #(.width(1)) (* integer library_binding = "basic";
      integer view_binding = "functional"; *)
      ams_alias_inst_0 (net015,net014[0]);
  ```

bindCdsAliasView

Adds the `view_binding = "functional"` attribute to instances of the `cds_alias` module that the AMS netlister adds to netlists.

This attribute specifies the view binding for instances of the `cds_alias` module. This specification is necessary when the `functional` view is not included in the Virtuoso Hierarchy Editor *View List*. Regardless of the setting of the `bindCdsAliasView` variable, the `basic` library, which contains the `cds_alias` module, must be defined in the `cds.lib` file.

Syntax

```
amsDirect.vlog bindCdsAliasView boolean t | nil
```

Values

`t`	Adds the `view_binding = "functional"` attribute to automatically inserted instances of the `cds_alias` module. This is the default.
`nil`	Does not add the `view_binding` attribute to instances of the `cds_alias` module.

Examples

- amsDirect.vlog bindCdsAliasView boolean t

  Tells the AMS netlister to add the `view_binding` attribute to automatically inserted instances of the `cds_alias` module, producing a statement similar to the following:

  ```
  cds_alias #(.width(1)) (* integer view_binding = "functional"; *)
      ams_alias_inst_0 (net015, net014[0]);
  ```

- The variables

  ```
  amsDirect.vlog bindCdsAliasLib boolean t
  amsDirect.vlog bindCdsAliasView boolean t
  ```

  tell the AMS netlister to add the `library_binding` and `view_binding` attributes to automatically inserted instances of the `cds_alias` module, producing a statement similar to the following:

  ```
  cds_alias #(.width(1)) (* integer library_binding = "basic";
      integer view_binding = "functional"; *)
      ams_alias_inst_0 (net015,net014[0]);
  ```

cdsGlobalsLib

Specifies the library to hold the `cds_globals` module created by AMS Designer.

Syntax

```
amsDirect.prep cdsGlobalsLib string "lib_name"
```

Value

lib_name	The library to hold the `cds_globals` module. The default is an empty string.

Description

AMS Designer automatically generates the `cds_globals` module that contains the global signals. The cell name for the module is fixed, but you can use this variable to specify the library name.

Example

```
amsDirect.prep cdsGlobalsLib string "myglobelib"
```

Tells AMS Designer to store the `cds_globals` module in the `myglobelib` library.

cdsGlobalsView

Specifies the view for the `cds_globals` module created by AMS Designer.

Syntax

```
amsDirect.prep cdsGlobalsView string "view_name"
```

Value

view_name	The view to be used for the `cds_globals` module. The default is an empty string.

Description

AMS Designer automatically generates the `cds_globals` module that contains the global signals. The cell name for the module is always `cds_globals`, but you can use this variable to specify the view name to be used.

Example

```
amsDirect.prep cdsGlobalsLib string "myglobelib"
amsDirect.prep cdsGlobalsView string "globeview"
```

Tell AMS Designer to store the `cds_globals` module in the `myglobelib` library, in the `cds_globals` cell, and to use a view name of `globeview`. (For information about `cdsGlobalsLib`, see "cdsGlobalsLib" on page 240.)

checkAndNetlist

Checks cellview data for Verilog-AMS compatibility and generates a Verilog-AMS netlist if no errors are found.

Syntax

```
amsDirect.vlog checkAndNetlist boolean t | nil
```

Values

`t`	Generates a netlist for the cellview if it does not find any errors while checking cellview data.
`nil`	Does not check cellview data and does not generate a netlist. This is the default.

Description

You can use this variable to create error-dependent netlists in Verilog-AMS. The `checkAndNetlist` variable takes precedence over the `checkOnly` variable.

Example

```
amsDirect.vlog checkOnly boolean nil
amsDirect.vlog checkAndNetlist boolean t
```

Tell the AMS netlister to generate a Verilog-AMS netlist for the cellview if there are no errors. In this example, the `nil` value for the `checkOnly` variable is ignored because the `t` value for the `checkAndNetlist` variable takes precedence.

checkOnly

Checks cellview data for Verilog-AMS compatibility without generating a Verilog-AMS netlist.

Syntax

```
amsDirect.vlog checkOnly boolean t | nil
```

Values

`t`	Checks cellview data, but does not generate a netlist for the cellview.
`nil`	Does not check cellview data or generate a netlist. This is the default.

Description

You can use this variable to check cellviews in Verilog-AMS. The `checkAndNetlist` variable takes precedence over the `checkOnly` variable.

Example

```
amsDirect.vlog checkOnly boolean t
```

Tells the AMS netlister to check a cellview for Verilog-AMS compliance but not to create a netlist for the cellview.

checktasks

Checks for the presence of non-predefined system tasks or functions in the source code.

Syntax

```
amsDirect.vlog checktasks boolean  t | nil
```

Values

`t`	Checks for the presence of non-predefined system tasks or functions in the source code.
`nil`	Does not check for the presence of non-predefined system tasks or functions in the source code.This is the default.

Example

```
amsDirect.vlog checktasks boolean t
```

Tells AMS Designer to compile Verilog files with the `-checktasks` option. As a result, the generated command might look like this.

```
xmvlog -checktasks
```

compileAsAMS

Specifies whether a Verilog file is handled as a Verilog-AMS file during compilation.

Syntax

```
amsDirect.vlog compileAsAMS boolean t | nil
```

Values

`t`	All Verilog files are compiled with the `-ams` option. This is the default.
`nil`	Verilog files with the extensions `.vams` or `.va` are compiled with the `-ams` option. Verilog files with the extension `.v` are compiled without the `-ams` option. If a Verilog file is a link, the decision to use or omit the `-ams` option is based on the extension of the name of the physical file that is the target of the link.

Description

You can use this variable to specify that Verilog-D files are not to be compiled with the `-ams` option. You might need to avoid using the `-ams` option, for example, if the Verilog-D files that you are compiling contain identifiers that are keywords in Verilog-AMS.

AMS Designer assumes that any file (or target of a file that is a link) with a `.v` extension contains Verilog-D code.

Example

```
amsDirect.vlog compileAsAMS boolean t
```

Tells AMS Designer to compile Verilog-D files with the `-ams` option. As a result, the generated command might look like this.

```
xmvlog -ams
```

compileExcludeLibs

Specifies libraries to be excluded when AMS Designer runs in compile all mode.

Syntax

```
amsDirect.prep compileExcludeLibs string "list_of_libraries"
```

Value

list_of_libraries	A list of library names separated by white space. Libraries with these names are not considered when AMS Designer runs in compile all mode. The default is an empty string.

Description

In compile all mode (such as when the `compileMode` variable is set to `"all"`), the default behavior of AMS Designer is to compile every cell referenced in the design hierarchy. However, when you use the `compileExcludeLibs` variable, cells in the design hierarchy that belong to a library listed in *list_of_libraries* are not compiled.

Read-only cellviews are never compiled. Nevertheless, if your design uses many cells from read-only libraries, the compile all step might run faster if you include those read-only libraries in *list_of_libraries*.

Example

```
amsDirect.prep compileExcludeLibs string "compiledLib readOnlyLib"
```

Tells AMS Designer not to attempt to compile any cells that belong to either the `compiledLib` or the `readOnlyLib` library.

compileMode

Specifies the conditions under which AMS Designer (working through the AMS netlister) compiles modules. When a module to be compiled is a VHDL or VHDL-AMS module, both the entity and the architecture are compiled.

Syntax

```
amsDirect.prep compileMode cyclic "none" | "incremental" | "all"
```

Values

`none`	Specifies that nothing is to be compiled.
`incremental`	Specifies that only newly netlisted modules are to be compiled. This is the default.
`all`	Specifies that, for each cellview in the design configuration, the `xmvlog` or `xmvhdl` compiler is to compile the netlist specified by the `master.tag` file for the view. If the master netlist is a Verilog, Verilog-A, or Verilog-AMS file, the `xmvhdl` compiler also compiles the first netlist found in files named `vhdl.vhms` or `vhdl.vhd` (in that order). If the master netlist is a VHDL or VHDL-AMS file, the `xmvlog` compiler also compiles the first netlist found in files named `verilog.vams`, `verilog.va`, `verilog.v`, or `veriloga.va` (in that order), These compilations occur whether or not the cell is netlisted in this run. **Note:** AMS Designer issues an error if the netlist specified by the `master.tag` file is a VHDL-AMS file, but the installed simulator does not support VHDL-AMS. AMS Designer compiles VHDL (digital) and VHDL-AMS design units in an order that resolves compilation order dependencies. If there is no `master.tag` file for the cellview, the `xmvlog` compiler compiles the first netlist found in files named `verilog.vams`, `verilog.va`, `verilog.v`, or `veriloga.va` (in that order). Similarly, the `xmvhdl` compiler also compiles the first netlist found in files named `vhdl.vhms` or `vhdl.vhd` (in that order).

In summary, specifying `all` causes a maximum of two files to be compiled for each cellview: one Verilog, Verilog-A, or Verilog-AMS cellview to be compiled by `ncvlog`; one VHDL or VHDL-AMS cellview to be compiled by `xmvhdl`.

Example

```
amsDirect.prep compileMode cyclic "incremental"
```

Tells AMS Designer to compile only newly created netlists.

confirmADEStateImport

Determines whether a dialog box opens to caution users that importing an ADE state overwrites existing netlister and compiler settings.

Syntax

```
amsDirect confirmADEStateImport boolean t | nil
```

Values

`t`	A dialog box appears when an ADE state is imported. The text of the dialog is `This action will cause your current netlister and compiler settings to be overwritten. A backup copy will be saved, but your current settings may change. Continue with import?`
`nil`	The dialog box does not appear.

Example

```
amsDirect       confirmADEStateImport   boolean nil
```

This example turns off the confirmatory dialog box so that clicking *OK* in the Import from ADE State window immediately imports the selected state.

connectRulesCell

Specifies the cell that contains the `connectrules` module.

Syntax

```
amsDirect.prep connectRulesCell string "cell"
```

Value

cell	The cell that contains the `connectrules` module. The default is `mixedsignal`.

Description

Depending on the version of the simulator that you are using, either the `connectRulesCell` or the `connectRulesCell2` variable is effective. The effective member of the pair, in conjunction with the `connectRulesLib` and `connectRulesView` variables, specifies the `connectrules` module.

Example

```
amsDirect.prep connectRulesLib string "mylib"
amsDirect.prep connectRulesCell string "comparator"
amsDirect.prep connectRulesView string "connectrules"
```

When the `connectRulesCell` variable is effective, these examples tell the elaborator and simulator to use the following `connectrules` module.

```
mylib.comparator:connectrules
```

connectRulesCell2

Specifies the cell that contains the `connectrules` module.

Syntax

```
amsDirect.prep connectRulesCell2 string "cell"
```

Value

cell	The cell that contains the `connectrules` module. The default is `ConnRules_5V_full`.

Description

Depending on the version of the simulator that you are using, either the `connectRulesCell` or the `connectRulesCell2` variable is effective. The effective member of the pair, in conjunction with the `connectRulesLib` and `connectRulesView` variables, specifies the `connectrules` module.

Example

```
amsDirect.prep connectRulesLib string "mylib"
amsDirect.prep connectRulesCell2 string "compar2"
amsDirect.prep connectRulesView string "connectrules"
```

When the `connectRulesCell2` variable is effective, these variables tell the elaborator and simulator to use the following `connectrules` module.

```
mylib.compar2:connectrules
```

connectRulesLib

Specifies the library that contains the `connectrules` module.

Syntax

```
amsDirect.prep connectRulesLib string "lib"
```

Value

lib	The library that contains the `connectrules` module. The default is an empty string.

Description

This variable, in conjunction with the `connectRulesCell` and `connectRulesView` variables, specifies the `connectrules` module.

Example

```
amsDirect.prep connectRulesLib string "mylib"
amsDirect.prep connectRulesCell string "comparator"
amsDirect.prep connectRulesView string "connectrules"
```

Tell the elaborator and simulator to use the following `connectrules` module.

```
mylib.comparator:connectrules
```

connectRulesView

Specifies the cellview that contains the `connectrules` module.

Syntax

```
amsDirect.prep connectRulesView string "view"
```

Value

view	The cellview that contains the `connectrules` module. The default is an empty string.

Description

This variable, in conjunction with the `connectRulesCell` and `connectRulesLib` variables, specifies the `connectrules` module.

Example

```
amsDirect.prep connectRulesLib string "mylib"
amsDirect.prep connectRulesCell string "comparator"
amsDirect.prep connectRulesView string "connectrules"
```

Tell the elaborator and simulator to use the following `connectrules` module.

```
mylib.comparator:connectrules
```

defaultRunDir

Specifies a directory to be used as the current run directory when the AMS menu is installed or when the `amsdesigner` command is run.

Syntax

```
amsDirect defaultRunDir string "rundir"
```

Value

rundir	The directory to be used as the run directory. The default is an empty string.

Description

An empty string for this variable means that the run directory must be specified in some other way, either by using the graphical user interface or by associating a run directory with a configuration. If, in the AMS Run Directory form, you turn on *Always use this run directory for this configuration*, that specification takes precedence over the value set by the `defaultRunDir` ams.env variable.

Example

```
amsDirect defaultRunDir string "newrundir"
```

Tells AMS Designer to use the `newrundir` directory as the run directory, unless this designation is overridden in some other way.

detailedDisciplineRes

Specifies the kind of discipline resolution to be used.

Syntax

```
amsDirect.prep detailedDisciplineRes boolean t | nil
```

Values

`t`	AMS Designer uses the detailed method of discipline resolution.
`nil`	AMS Designer uses the default method of discipline resolution. This is the default.

Description

For a description of these methods, see the “Discipline Resolution Method” section of Chapter 11, in the *Cadence Verilog-AMS Language Reference*.

Example

```
amsDirect.prep detailedDisciplineRes boolean nil
```

Specifies that the default method of discipline resolution is to be used.

discipline

Specifies a default discipline for discrete nets for which a discipline is either not specified or cannot be determined through discipline resolution.

Syntax

```
amsDirect.prep discipline string "discipline"
```

Value

discipline	The discipline to be used for discrete nets of otherwise unknown discipline. The default is `logic`.

Example

```
amsDirect.prep discipline string "logic"
```

Specifies that the `logic` discipline is to be used for discrete nets that do not have a known discipline.

errOutInconsistentMasters

Controls whether the netlister terminates with an error when it encounters an unbound master cellview.

Syntax

```
amsDirect.vlog errOutInconsistentMasters boolean t | nil
```

Values

`t`	The netlister terminates with an error if it encounters any unbound master cellviews.
`nil`	The netlister does not terminate with an error when it encounters unbound master cellviews. This is the default value.

Example

```
amsDirect.vlog errOutInconsistentMasters boolean t
```

The netlister terminates with an error if it encounters any unbound master cellviews.

excludeViewNames

Specifies the names of cellviews that are not to be netlisted.

Syntax

```
amsDirect.vlog excludeViewNames string "list_of_view_names"
```

Value

list_of_view_names	A list of view names separated by white space. Cellviews with these names are not netlisted. The default is an empty string.

Description

Normally, changes to cellviews while netlisting is enabled or changes to the CDF of cells while the `netlistAfterCdfChange` variable is set to `t` trigger netlisting. However, cells whose names are included in *list_of_view_names* are not netlisted.

Example

```
amsDirect.vlog excludeViewNames string "sch[0-3]"
```

hdlVarFile

Specifies the name of the `hdl.var` file to be used with the `ncvlog`, `ncelab`, and `xmsim` commands.

Syntax

```
amsDirect.prep hdlVarFile string "file"
```

Value

file	An `hdl.var` file to be used with the `-hdlvar` option of the `ncvlog`, `ncelab`, and `xmsim` commands. If *file* is not specified, the `-hdlvar` option is not used with these commands. The default is an empty string.

Description

If *file* is an empty string, the `ncvlog`, `ncelab`, and `xmsim` commands run without the `-hdlvar` option. As a result, each application looks for an `hdl.var` file in the directory where that application started. If there is no `hdl.var` file in that location, the program issues a warning. Because `xmvlog` starts in the directory where you start the Cadence software and `xmelab` and `xmsim` start in the run directory, the programs are likely to use different `hdl.var` files if you do not specify them explicitly.

If you use a relative path, be aware that paths are relative to the directory where the program starts. The `xmvlog` program starts in the current working directory so the path is relative to that directory. However, the `xmelab` and `xmsim` programs start in the run directory so the path for them is relative to the run directory. As a consequence, the different programs are likely to use different `hdl.var` files.

To be sure that all the programs find the appropriate `hdl.var` file, use an absolute path.

Example

```
amsDirect.prep hdlVarFile string "prepvarfile"
```

Specifies that the `ncvlog`, `ncelab`, and `xmsim` commands generated by AMS Designer are to include the following option.

```
-hdlvar "prepvarfile"
```

headerText

Specifies the kind of header to be used at the beginning of netlists generated by AMS Designer.

Syntax

```
amsDirect.vlog headerText cyclic "none" | "file" | "script"
```

Values

`none`	Specifies that the default header is to be used.
`file`	Specifies that the header of the netlist is to consist of the default header followed by the text of a file. The name of the file containing the text is specified by the `templateFile` variable. For more information, see "templateFile" on page 413.
`script`	Specifies that the header of the netlist is to consist of the default header followed by the text generated by running a script. The name of the file containing the script is specified by the `templateScript` variable. For more information, see "templateScript" on page 414.

Example

```
amsDirect.vlog headerText cyclic "none"
```

Tells AMS Designer to insert the default header at the beginning of each generated netlist. As a result, each netlist begins with lines like the following.

```
// Verilog-AMS netlist generated by the AMS netlister, version 4.4.6.100.43.
// Cadence Design Systems, Inc.
```

ieee1364

Checks the source code for compatibility with the IEEE standard described in *IEEE-1364 Verilog Hardware Description Language Reference Manual*.

Syntax

```
amsDirect.vlog ieee1364 boolean  t | nil
```

Values

`t`	Checks the source code for compatibility with the IEEE standard described in *IEEE-1364 Verilog Hardware Description Language Reference Manual*.
`nil`	Does not check the source code for compatibility with the IEEE standard. This is the default.

Example

```
amsDirect.vlog ieee1364 boolean t
```

Tells AMS Designer to compile Verilog files with the `-ieee1364` option. As a result, the generated command might look like this.

```
xmvlog -ieee1364
```

ifdefLanguageExtensions

Controls the netlisting of attributes.

Syntax

```
amsDirect.vlog ifdefLanguageExtensions boolean t | nil
```

Values

`t`	Generates `` `ifdef INCA `` clauses in the netlist for attribute statements.
`nil`	Does not generate `` `ifdef INCA `` clauses. This is the default.

Description

If you plan to use a compiler that does not support the Cadence attribute statements, you can use this variable to enclose the statements in an `` `ifdef INCA `` clause.

Note: This clause produces a Verilog-AMS netlist that is more difficult to read.

Example

You need to copy your netlists to a different location where they will be used in a purely text based flow without using configurations and the Virtuoso® Hierarchy Editor. In this situation, the library bindings in the netlist need to be disabled.

With the `ifdefLanguageExtensions` variable set to `nil`, the netlist looks like this.

```
vsource #(.dc(3), .type("dc"))  (*
integer library_binding = "analogLib";  *) V0 ( cds_globals.\vdd! ,
cds_globals.„nd!  );
vsource #(.dc(-3), .type("dc"))  (*
integer library_binding = "analogLib";  *) V1 ( cds_globals.\vss! ,
cds_globals.„nd!  );
```

Setting the `ifdefLanguageExtensions` variable to `t` results in a netlist where the library bindings are enclosed in `` `ifdef INCA `` clauses, so that they can be turned off.

```
vsource #(.dc(3), .type("dc"))
`ifdef INCA (* integer library_binding = "analogLib";  *) `endif
V0 ( cds_globals.\vdd! , cds_globals.„nd!  );
vsource #(.dc(-3), .type("dc"))
`ifdef INCA (* integer library_binding = "analogLib";  *) `endif
V1 ( cds_globals.\vss! , cds_globals.„nd!  );
```

ignoreIllegalCDFParams

Specifies whether to ignore non-compliant CDF parameters when netlisting.

For example, CDF parameters such as `min`, `max`, and `abs` for the `vcvs` cell in the analogLib library are non-compliant because they are reserved keywords in the Verilog-AMS language. By default, the netlister issues a warning or error message (depending on the value of the `allowIllegalIdentifiers` variable). You can turn off this notification by setting the `ignoreIllegalCDFParams` variable to `t`.

Syntax

```
amsDirect.vlog ignoreIllegalCDFParams boolean t | nil
```

Values

`t`	Netlisting does not notify you about any non-compliant CDF parameter names.
`nil`	Netlisting notifies you (by warning or error message) about any non-compliant CDF parameter names. This is the default. **Note:** The type of notification depends on the value of the `allowIllegalIdentifiers` variable.

implicitTmpDir

Specifies the implicit temporary (TMP) directory. The AMS Designer environment ensures that the specified directory is the same as the run directory.

The `implicitTmpDir` variable has no effect when the `netlistToRunDir` variable is set to `nil` or when the `useRunDirNetlistsOnly` variable is set to `nil`.

Note: The AMS Designer environment sets this variable automatically, and you should not change the setting by hand. To control netlisting into temporary directories, you need to set only the `netlistToRunDir` variable, and, optionally, the `useRunDirNetlistsOnly` variable.

Syntax

```
amsDirect implicitTmpDir string "implicitTmpDir"
```

Value

implicitTmpDir	Automatically set, by the AMS Designer environment, to the run directory.

incdir

Specifies directories to be searched for files specified by the `` `include `` compiler directive.

Syntax

```
amsDirect.vlog incdir string "dirs_to_search"
```

Value

dirs_to_search	Directories to be searched for specified files. The format must be as illustrated in the following example. The default is an empty string.

Example

```
amsDirect.vlog incdir string "11-LevelOneDir11-LevelTwoDir"
```

Generates a command that includes two `-incdir` options.

```
ncvlog
    -incdir LevelOneDir
    -incdir LevelTwoDir
```

includeFiles

Specifies a list of files to be included with the `‘include` directive at the top of each Verilog-AMS netlist that is created by the AMS netlister.

Syntax

```
amsDirect.vlog includeFiles string "(file_to_include_1)
        { (file_to_include_N) }"
```

Value

file_to_include_N	Files to be included in the netlist by the `‘include` compiler directive. If you list more than a single file, separate the files with spaces. The default is `disciplines.vams`.

Example

```
amsDirect.vlog includeFiles string "(disciplines.vams) (func1.h) (func2.h)"
```

Tells AMS netlister to include the files `func1.h` and `func2.h` at the top of the netlist. As a result, the netlist contains the lines:

```
‘include "disciplines.vams"
‘include "func1.h"
‘include "func2.h"
```

includeInstCdfParams

Specifies how the AMS netlister handles CDF parameters.

Syntax

```
amsDirect.vlog includeInstCdfParams boolean t | nil
```

Values

`t`	For each instance, writes to the netlist all parameters found in the CDF for the instance master.
`nil`	For each instance, writes to the netlist only CDF parameters set on the instance and all CDF parameters containing pPar or atPar expressions. This is the default.

Description

This variable is ignored if the instance master has an `ams` section in the simulation information (simInfo) section of the CDF. In this case, the AMS netlister does only what the simInfo says to do.

initFile

Specifies a SKILL file to be loaded at startup. The function definitions and code in the file are used to override netlist procedures.

Syntax

```
amsDirect initFile string "path"
```

Value

path	The path and filename of a SKILL file. The path can contain shell environment variables (consisting of a `$` followed by alphanumeric characters). A path that begins with a `/` (slash) is considered an absolute path. A path that does not begin with a `/` is considered to be relative to the current working directory (CWD).

The file can contain SKILL function definitions and code to override netlist procedures. The user code can call

- Core SKILL language functions
- DB functions (which begin with `db`)
- DDPI functions (which begin with `dd`)
- CDF functions (which begin with `cdf`)
- AMS functions (which begin with `ams`)

The SKILL code must not assume that other contexts are loaded by default, though the code can load other contexts as necessary.

Example

```
amsDirect initFile string "$YOUR_INSTALL_DIR/tools/dfII/local/amsProcs.il"
```

Tells AMS netlister to load the `amsProcs.il` file.

instClashFormat

Specifies the format to be used to map the names of instances that collide with names of other netlist constructs.

Syntax

```
amsDirect.vlog instClashFormat string "format"
```

Value

format	All characters, except those listed below, are printed as included in *format*. The following characters have the indicated special meanings. `%b` Original name of the instance `%%` Prints the `%` character The default value of *format* is `%b_instclash`, which produces a mapped name like `origname_instclash` for an instance originally named `origname`.

If the resulting name is illegal in Verilog-AMS, the name is mapped. If the mapped name clashes with the name of another object, the name undergoes collision mapping.

Example

```
amsDirect.vlog instClashFormat string "%b_iclash"
```

Tells AMS netlister to map clashing instance names with a suffixed `_iclash`. For example, you have an instance `samp` with a name that clashes with a net named `samp`. The AMS netlister maps the instance to the system-generated name `samp_iclash`.

iterInstExpFormat

Specifies the format to be used for the names of constituent elements generated by the expansion of an iterated instance.

Syntax

```
amsDirect.vlog iterInstExpFormat string "format"
```

Value

format	All characters, except those listed below, are printed as included in *format*. The following characters have the indicated special meanings.

%b	Base name of the instance
%l (small L)	Left bound of the range
%r	Right bound of the range
%i	Index of the current iteration
%%	Prints the % character

The default value of *format* is %b_%i, which produces names like instbn_1, instbn_2, and so on, where instbn is the base name of the instance.

If a resulting name is illegal in Verilog-AMS, the name is mapped. If the mapped name clashes with the name of another object, the name undergoes collision mapping.

Example

```
amsDirect.vlog iterInstExpFormat string "%b_%l_%r_%i"
```

Tells AMS netlister to generate names that include the left and right bounds. For example, you have an iterated instance with the name scatstr. The names of the expanded instances are:

```
scatstr_1_3_1
scatstr_1_3_2
scatstr_1_3_3
```

language

Specifies the language to be used for netlists.

Syntax

```
amsDirect.prep language string "verilog"
```

Value

`verilog`	Specifies that the language to be used for netlists is Verilog-AMS. This is the default.

Description

In this release, Verilog-AMS is the only supported language.

lexpragma

Enables processing of lexical pragmas.

Syntax

```
amsDirect.vlog lexpragma boolean  t | nil
```

Values

`t`	Turns on processing of lexical pragmas.
`nil`	Turns of processing of lexical pragmas. This is the default

Description

Lexical pragmas are pragmas that can be associated with any Verilog or VHDL construct to indicate that translation/synthesis is turned off. The following pragmas are classified as lexical pragmas:

- cadence translate_off and cadence translate_on (also: synopsys translate_off and synopsys translate_on)
- cadence synthesis_off and cadence synthesis_on (also: synopsys synthesis_off and synopsys synthesis_on)
- rtl_synthesis off and rtl_synthesis on

If you compile with the `-lexpragma` option, any HDL constructs between a translate_off/ synthesis_off pragma and a translate_on/synthesis_on pragma are treated as comments. For example, if the source code contains the following pragmas, `'define CI2CLKP 10` is treated as a comment.

```
'define CI2CLKP 512
// cadence translate_off
'define CI2CLKP 10
// cadence translate_on
```

If you use both -pragma and -lexpragma, lexical pragmas are processed with -lexpragma.

Example

```
amsDirect.vlog lexpragma boolean t
```

Tells AMS Designer to compile Verilog files with the `-lexpragma` option. As a result, the generated command might look like this.

```
xmvlog -lexpragma
```

logFileAction

Controls the generation of log files.

Syntax

```
amsDirect.vlog logFileAction cyclic "Overwrite log file" | "Append
    log file" | "No log file"
```

Values

`Overwrite log file`	Overwrites the log file each time xmvlog runs. This is the default.
`Append log file`	Appends all xmvlog log information to a single file.
`No log file`	Specifies that no log file be created.

Examples

- The `ams.env` variable

  ```
  amsDirect.vlog logFileAction cyclic "Overwrite log file"
  ```

  generates an `xmvlog` command similar to the following.

  ```
  ncvlog
  -logfile /usr1/cds11752/alpha6/test8/SAR_A2D/tutorial_run/ncvlog.log
  ```

- The `ams.env` variable

  ```
  amsDirect.vlog logFileAction cyclic "Append log file"
  ```

  generates an `xmvlog` command similar to the following.

  ```
  ncvlog
      -logfile /usr1/cds11752/alpha6/test8/SAR_A2D/tutorial_run/ncvlog.log
      -append_log
  ```

- The `ams.env` variable

  ```
  amsDirect.vlog logFileAction cyclic "No log file"
  ```

  generates an `xmvlog` command similar to the following.

  ```
  ncvlog
      -nolog
  ```

logFileName

Sets the name of the log file.

Syntax

```
amsDirect logFileName string "logFileName"
```

Value

logFileName	Specifies the name of the log file. The default is ams_direct.log.

Description

When the AMS netlister processes a design, it creates a log file that contains errors, warnings, and informational messages about the design. You can use this variable to name the log file.

The *logFileName* that you specify with this variable interacts with the CDS_LOG_PATH environment variable to determine the log file name that is used.

- If *logFileName* is an absolute path, the log file is written to *logFileName*.
- If *logFileName* is a relative path and
  - CDS_LOG_PATH is null, *logFileName* is placed in the current directory.
  - CDS_LOG_PATH is non-null, the value of CDS_LOG_PATH is prepended to the *logFileName*.
- Setting both *logFileName* and the CDS_LOG_PATH to absolute paths causes a fatal error.

Note: The -LOg option of the amsdirect command takes precedence over the logFileName variable.

Examples

- The *logFileName* variable is not used and CDS_LOG_PATH environment variable is unset. The default *logFileName* is used and log data goes to ams_direct.log in the current directory.

- The *logFileName* variable is not used and CDS_LOG_PATH environment variable is set to the absolute path

  /usr1/dave/test7

  Log data goes to

  /usr1/dave/test7/ams_direct.log

- The *logFileName* variable is set to the absolute path

  /usr1/dave/test8/test8_log2

  The CDS_LOG_PATH environment variable is set to the absolute path

  /usr1/dave/test8

  The *logFileName* variable takes precedence and the log data is written to

  /usr1/dave/test8/test8_log2

- The *logFileName* variable is set to the relative path

  ./usr1/dave/test8/test8_log2

  The CDS_LOG_PATH environment variable is set to

  /usr1/dave/test8

  In this case, the CDS_LOG_PATH is prepended to the *logFileName* and log data goes to

  /usr1/dave/test8/usr1/dave/test8/test8_log2

macro

Defines macros for the `xmvlog` command.

Syntax

```
amsDirect.vlog macro string "macros"
```

Value

macros	Macros to be defined. The format must be as illustrated in the following example. The default is an empty string.

Example

```
amsDirect.vlog macro string "4-gate2-or4-slow8-'16'h03'"
```

Generates an `xmvlog` command similar to the following.

```
ncvlog
    -define gate=or
    -define slow=16'h03
```

markcelldefines

Inserts `‘celldefine` and `‘endcelldefine` compiler directives to tag module instances as cell instances.

Syntax

```
amsDirect.vlog markcelldefines boolean  t | nil
```

Values

`t`	Inserts `‘celldefine` and `‘endcelldefine` compiler directives to tag module instances as cell instances.
`nil`	Does not insert `‘celldefine` and `‘endcelldefine` compiler directives to tag module instances as cell instances. This is the default.

Example

```
amsDirect.vlog  markcelldefines boolean t
```

Generates an `xmvlog` command similar to the following.

```
ncvlog
    -libcell
```

maxErrors

Stops compilation if the number of errors reaches the specified maximum limit.

Syntax

```
amsDirect.vlog maxErrors int maxErrors
```

Value

maxErrors	A positive integer. Halts compilation after this number of errors occur. The default is 50.

Example

```
amsDirect.vlog maxErrors int 50
```

Tells AMS Designer to compile Verilog files with the `-errormax` option. As a result, the generated command might look like this.

```
xmvlog -errormax 50
```

messages

Prints informational messages as the compiler runs.

Syntax

```
amsDirect.vlog messages boolean  t | nil
```

Values

`t`	Prints informational messages as the compiler runs.
`nil`	Does not print informational messages as the compiler runs. This is the default.

Example

```
amsDirect.vlog messages boolean t
```

Tells AMS Designer to compile Verilog files with the `-messages` option. As a result, the generated command might look like this.

```
xmvlog -messages
```

modifyParamScope

Specifies that the AMS netlister treat atPar and dotPar expressions as pPar and iPar expressions, respectively.

Syntax

```
amsDirect modifyParamScope cyclic "no" | "warn" | "yes"
```

Values

`no`	Prints an error message and halts netlisting when the AMS netlister finds atPar or dotPar expressions. This is the default.
`warn`	Generates a warning when the AMS netlister finds an atPar or dotPar expression and treats the atPar or dotPar expression as a pPar or iPar expression, respectively.
`yes`	Treats atPar and dotPar expressions as pPar and iPar expressions, respectively. No warning messages are generated.

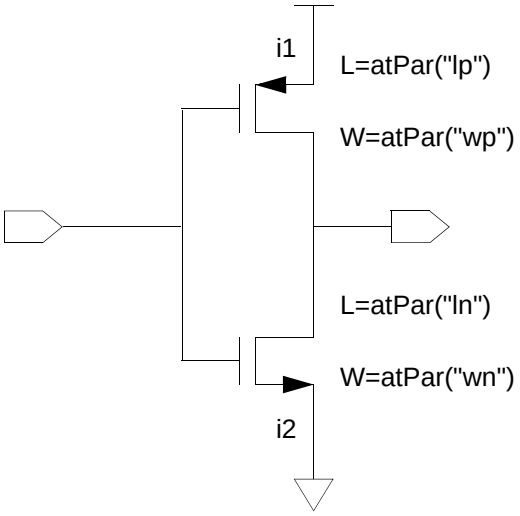
Description

The AMS netlister netlists one cellview at a time; it cannot see hierarchical dependencies defined or resolved outside of the current cellview. In addition, Verilog-AMS requires that passed parameters be resolved through the level of hierarchy immediately preceding the cellview to which the parameter applies. In other words, parameter passing cannot skip levels of the hierarchy. Because atPar and dotPar expressions allow parameters to be resolved in non-contiguous levels of the hierarchy, the AMS netlister does not support these expressions.

If you specify `warn` or `yes`, the AMS netlister treats atPar and dotPar expressions as pPar and iPar expressions, respectively, and generates a netlist. However, to avoid incorrect simulation results, you must ensure that the block instantiating the cell sets the instance parameters appropriately.

Example

Consider the following example of an inverter that employs atPar expressions. Assume that the nmos has defaults of `ln=3u` and `wn=20u` and that the pmos has defaults of `lp=3u` and `wp=40u`.



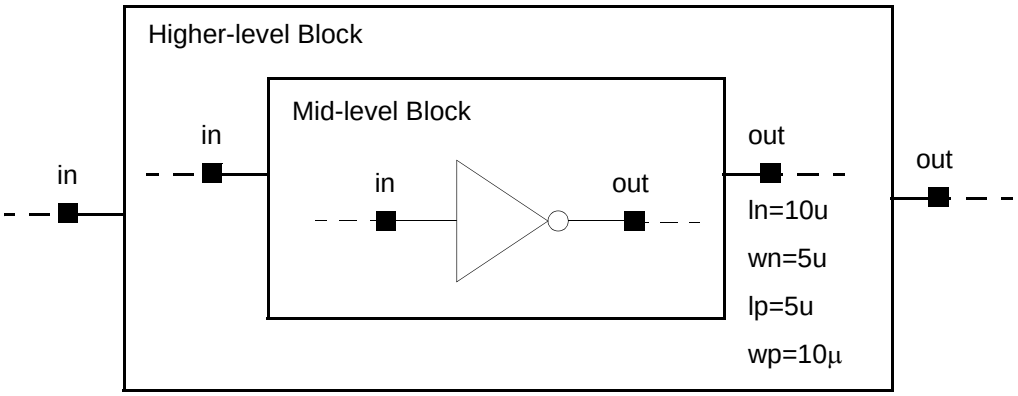
When this inverter is netlisted by the AMS netlister, it has an instance of an nmos and an instance of a pmos, each with parameters to be passed in:

```
pmos #(.W(wp), .L(lp)) i1 ( port_connections );
nmos #(.W(wn), .L(ln)) i2 ( port_connections );
```

The inverter module netlisted by the AMS netlister also has parameter statements for the parameters that are to supply values to the nmos and pmos instances:

```
parameter ln = 3u;
parameter wn = 20u;
parameter lp = 3u;
parameter wp = 40u;
```

Now assume that this inverter is instantiated in a mid-level block, as follows:



The definition of the atPar expression allows the values for the parameters `ln`, `wn`, `lp`, and `wp` to be provided at any level of the hierarchy above the mid-level block. In the preceding diagram, the values set in the higher-level block override the defaults defined in the nmos and pmos, and are used during the simulation:

- `ln` is set to 10µ for the simulation
- `wn` is set to 5µ for the simulation
- `lp` is set to 5µ for the simulation
- `wp` is set to 10µ for the simulation

This behavior is not possible when using Verilog-AMS. Verilog-AMS allows parameters to be passed from one level of hierarchy to the next level below, but the passing must be between contiguous levels. This behavior is identical to what is accomplished by pPar expressions. To be able to generate a netlist for the example, the AMS netlister must treat the atPar expressions as it does pPar expressions, expecting that any overriding of the parameters is done at the level of hierarchy immediately above.

Now assume that the AMS netlister is instructed to treat atPar expressions as it does pPar expressions. In this case, the higher-level block has an instance of the mid-level block, with the parameters set:

```
midlevel #(.ln(10u), .wn(5u), .lp(5u), .wp(10u)) i1 ( port_connections );
```

This instantiation assumes that the mid-level module has parameter declarations for the four parameters being passed in. However, the mid-level block does not reference these parameters at all, so no parameter declarations are printed by the AMS netlister.

The mid-level block has an instance of the inverter, passing no parameters at all:

```
inverter i1 ( port_connections );
```

Thus, when the nmos and pmos parameters are resolved, they are set to the defaults, because no values are passed in to override them:

- `ln` is set to 3µ for the simulation
- `wn` is set to 20µ for the simulation
- `lp` is set to 3µ for the simulation
- `wp` is set to 40µ for the simulation

Notice how these simulation values differ from those listed earlier. This example illustrates how instructing the AMS netlister to treat atPar expressions as pPar expressions might not produce the results you expect. To avoid incorrect results, ensure that parameters are passed in accordance with Verilog-AMS restrictions.

ncelabAccess

Sets the visibility access for all objects in the design.

Syntax

```
amsDirect.prep ncelabAccess cyclic  "Off" | "Read" | "Read/Write" |
    "Connectivity" | "All"
```

Values

`Off`	Equivalent to the option `-access -r-w-c`. This is the default.
`Read`	Appends the option `-access +r-w-c`.
`Read/Write`	Appends the option `-access +r+w-c`.
`Connectivity`	Appends the option `-access +r-w+c`.
`All`	Appends the option `-access +r+w+c`.

Example

```
amsDirect.prep  ncelabAccess    cyclic  "Read/Write"
```

Generates an `xmelab` command that looks like this.

```
xmelab amslib.top:config -access +r+w-c
```

ncelabAfile

Specifies an access file. An access file is a text file that lets you set the visibility access for particular instances or portions of a design.

Syntax

```
amsDirect.prep ncelabAfile string "path_and_file"
```

Value

path_and_file	The default is an empty string.

Example

```
amsDirect.prep ncelabAfile string "/usr1/alpha6/test8/SAR_A2D/afile.acs"
```

Generates an `xmelab` command like the following.

```
xmelab amslib.top:config -afile /usr1/alpha6/test8/SAR_A2D/afile.acs
```

ncelabAnnoSimtime

Enables the use of PLI/VPI routines that modify delays at simulation time.

Syntax

```
amsDirect.prep ncelabAnnoSimtime boolean  t | nil
```

Values

`t`	
`nil`	This is the default.

Description

The PLI/VPI routines that modify routines are `acc_replace_delays`, `acc_append_delays`, and `vpi_put_delays`.

If you do not specify this option at elaboration time, but then run a PLI/VPI routine that tries to modify delays at simulation time, AMS Designer issues a message and does not modify delays.

This option disables optimizations in the simulator that take delays into account and has some performance impact. Use this option only if you intend to modify delays at simulation time.

Using this option sets the default access to simulation objects to read/write when the design is elaborated. Do not use this option if you want to run in regression mode.

Example

```
amsDirect.prep ncelabAnnoSimtime boolean t
```

Tells the AMS netlister to prepare to simulate with routines that modify delays at simulation time. As a result, the generated `xmelab` command looks like the following.

```
xmelab amslib.top:config -anno_simtime
```

ncelabArguments

Specifies additional arguments to be passed to the `xmelab` elaborator.

Syntax

```
amsDirect.prep ncelabArguments string "arguments"
```

Value

arguments	One or more arguments to be passed to the `xmelab` elaborator. The default is an empty string.

Description

If *arguments* is an empty string, the `xmelab` command includes just the arguments listed on the *Elaborator* pane of the AMS Options window. You can use the `ncelabArguments` variable with a non-empty string to pass additional arguments to the elaborator.

Example

```
amsDirect.prep ncelabArguments string "-libverbose"
```

Adds the `-libverbose` argument to the other arguments normally used on the `xmelab` command.

ncelabCoverage

Enables code coverage instrumentation for the digital part of the design.

Syntax

```
amsDirect.prep ncelabCoverage boolean  t | nil
```

Values

`t`	Enables code coverage instrumentation.
`nil`	Turns off code coverage instrumentation. This is the default.

Example

```
amsDirect.prep ncelabCoverage boolean t
```

Generates an `xmelab` command like the following.

```
xmelab amslib.top:config -coverage
```

ncelabDelayMode

Specifies the delay mode to be used for digital Verilog-AMS portions of the hierarchy.

Syntax

```
amsDirect.prep ncelabDelayMode cyclic "None" | "Zero" | "Unit" |
    "Path" | "Distributed"
```

Values

`None`	Delays simulate as specified in the model's source description files. This is the default.
`Zero`	Similar to `Unit` delay mode in that the simulator ignores all module path delay information, timing checks, and structural and continuous assignment delays.
`Unit`	The AMS simulator ignores all module path delay information and timing checks and converts all non-zero structural and continuous assignment delay expressions to a unit delay of one simulation time unit.
`Path`	The AMS simulator derives its timing information from specify blocks. When a module contains a specify block with one or more module path delays, all structural and continuous assignment delays within that module (with the exception of `trireg` charge decay times) are set to zero.
`Distributed`	The AMS simulator ignores all module path delay information and uses all distributed delays and timing checks. Distributed delays are delays on nets, primitives, or continuous assignments–in other words, delays other than those specified in procedural assignments and specify blocks.

Example

```
amsDirect.prep ncelabDelayMode cyclic "Unit"
```

Generates an `xmelab` command like the following.

```
xmelab amslib.top:config -delay_mode Unit
```

When you elaborate with this command, the AMS simulator ignores all module path delay information and timing checks and converts all non-zero structural and continuous assignment delay expressions to a unit delay of one simulation time unit.

ncelabDelayType through ncelabMessages

These `ams.env` `ncelab*` variables correspond to `ncelab` command options as follows:

ams.env Variable	ncelab Command Option
ncelabDelayType	-MAxdelays, -MIndelays, -TYpdelays
ncelabDisableenht	-DISAble_enht
ncelabEpulseFiltering	-EPULSE_ONDetect, -EPULSE_ONEvent
ncelabEpulseNeg	-EPULSE_NEg
ncelabExpand	-EXPand
ncelabExtendtcheckdatalimit	-EXTEND_TCHECK_Data_limit
ncelabExtendtcheckreferencelimit	-EXTEND_TCHECK_Reference_limit
ncelabGenafile	-GENAfile
ncelabIeee1634	-IEEe1364
ncelabInterconnmultisrc	-CAint
ncelabLibverbose	-LIBVerbose
ncelabLoadpli1	-LOADPli1
ncelabLoadvpi	-LOADVpi
ncelabLogFileAction	-LOGfile, -NOLog, -APpend_log
ncelabMaxErrors	-ERrormax
ncelabMessages	-MEssages

For information about `ncelab` command options, see Chapter 8 of *NC-Verilog Simulator Help*.

ncelabMixEsc

Controls whether the `-mixesc` option is passed on the `xmelab` command. The `-mixesc` option is required when elaborating if you instantiate VHDL or VHDL-AMS in a Verilog or Verilog-AMS module and you use escaped entity, port, or generic names within the VHDL or VHDL-AMS descriptions.

Syntax

```
amsDirect.prep ncelabMixEsc boolean t | nil
```

Values

`t`	Places a checkmark next to the *Allow mixed-case, escaped identifiers in VHDL* field, on the *VHDL* pane of the AMS Options window. As a result, the `-mixesc` option is passed on the `xmelab` command.
`nil`	Removes the checkmark, indicating that the `-mixesc` option is not to be passed on the `xmelab` command. This is the default.

Example

```
amsDirect.prep ncelabMixEsc boolean t
```

Places the checkmark next to the *Allow mixed-case, escaped identifiers in VHDL* field. As a result, the elaborator is able to distinguish VHDL entities whose escaped names differ only by the case of letters.

ncelabModelFilePaths

Do not hand-edit this variable.

This variable contains the information that populates the model files table on the Model Library Setup form.

Syntax

```
amsDirect.prep  ncelabModelFilePaths    string  "model_files"
```

Value

model_files	The status, paths, names, and sections of analog model files. The default value is an empty string.

Example

```
amsDirect.prep ncelabModelFilePaths   string "9-isEnabled5-false4-path31-$PROJ3/
SAR_A2D/spectreprim3.scs:9-isEnabled5-false4-path31-$PROJ3/SAR_A2D/
spectreprim2.scs7-section7-typical:9-isEnabled4-true4-path58-/usr1/cds11752/
alpha6/vhdltestdir/SAR_A2D/spectre_prim.scs"
```

ncelabNeverwarn through ncelabVipdelay

These `ams.env` `ncelab*` variables correspond to `ncelab` command options as follows:

ams.env Variable	ncelab Command Option
ncelabNeverwarn	-NEVerwarn
ncelabNoautosdf	-NOAutosdf
ncelabNocopyright	-NOCopyright
ncelabNoipd	-NOIpd
ncelabNonegtchk	-NONEg_tchk
ncelabNonotifier	-NONOtifier
ncelabNosource	-NOSOurce
ncelabNostdout	-NOSTdout
ncelabNoTchkMsg	-NO_TCHK_Msg
ncelabNoTchkXgen	-NO_TCHK_Xgen
ncelabNotimingchecks	-NOTImingchecks
ncelabNovitalaccl	-NOVitalaccl
ncelabNoVpdmsg	-NO_VPD_Msg
ncelabNoVpdXgen	-NO_VPD_Xgen
ncelabNowarn	-NOWarn
ncelabNtcWarn	-NTC_Warn
ncelabOmichecklvl	-OMicheckinglevel
ncelabPathpulse	-PAthpulse
ncelabPlinooptwarn	-PLINOOptwarn
ncelabPlinowarn	-PLINOWarn
ncelabPresrvResFn	-PReserve
ncelabPulseE	-PULSE_E
ncelabPulseIntE	-PULSE_INT_E
ncelabPulseIntR	-PULSE_INT_R

ams.env Variable	ncelab Command Option
ncelabPulseR	-PULSE_R
ncelabRelax	-Relax
ncelabSdfCmdFile	-SDF_Cmd_file
ncelabSdfNocheckCelltype	-SDF_NOCHECK_Celltype
ncelabSdfNoHeader	-NO_Sdfa_header
ncelabSdfNoWarnings	-SDF_NO_Warnings
ncelabSdfprecision	-SDF_Precision
ncelabSdfverbose	-SDF_Verbose
ncelabSdfWorstcaseRounding	-SDF_Worstcase_rounding
ncelabsolverInfo	
ncelabStatus	-STatus
ncelabTopLvlGeneric	-GENEric
ncelabUpdate	-UPDate
ncelabUse5x4vhdl	-USE5X4VHdl
ncelabUseAddArgs	None. Determines whether additional specified arguments are used on the ncelab command.
ncelabUseAfile	-AFile
ncelabUseExtendtcheckdatalimit	-EXTEND_TCHECK_Data_limit
ncelabUseExtendtcheckreferencelimit	-EXTEND_TCHECK_Reference_limit
ncelabUseGenafile	None. Determines whether the option to create the access file is included on the ncelab command.
ncelabUseGeneric	None. Determines whether the option to use the generic value is included on the ncelab command.
ncelabUsePulseE	-PULSE_E
ncelabUsePulseIntE	-PULSE_INT_E
ncelabUsePulseIntR	-PULSE_INT_R

ams.env Variable	ncelab Command Option
ncelabUsePulseR	-PULSE_R
ncelabUseSdfprecision	-SDF_Precision
ncelabV93	-V93
ncelabVipdelay	-VIPDMAx, -VPIDMIn

For information about `ncelab` command options, see Chapter 8 of *NC-Verilog Simulator Help*.

ncsimArguments

Specifies additional arguments to be passed to the `ncsim` simulator.

Syntax

```
amsDirect.prep ncsimArguments string "arguments"
```

Value

arguments	One or more arguments to be passed to the `ncsim` simulator. The default is an empty string.

Description

`ncsim` *configLib.cell*:*view* `-analogcontrol` *fileName* `-amslic`

Illustrates the form of the default command that AMS Designer uses to run the simulator,

If *arguments* is an empty string, the `ncsim` command includes just the arguments listed on the *Simulator* pane of the AMS Option window. You can use the `ncsimArguments` variable with a non-empty string to pass additional arguments to the simulator.

Example

```
amsDirect.prep ncsimArguments string "-status"
```

Adds the `-status` argument to the other arguments normally used on the `ncsim` command.

ncsimEpulseNoMsg through ncsimExtassertmsg

These `ams.env` `ncsim*` variables correspond to `ncsim` command options as follows:

ams.env Variable	ncsim Command Option
`ncsimEpulseNoMsg`	`-EPulse_no_msg`
`ncsimExtassertmsg`	`-EXTassertmsg`

For information about `ncsim` command options, see Chapter 9 of *NC-Verilog Simulator Help*.

ncsimGUI

Controls whether the simulator runs with a graphical user interface (GUI).

Syntax

```
amsDirect.prep ncsimGUI boolean t | nil
```

Values

`t`	Opens the GUI when the simulator runs. This is the default.
`nil`	The GUI does not open. Depending on the value of the `ncsimTcl` variable, either the Tcl interface opens or the simulator runs in batch mode.

Examples

```
amsDirect.prep ncsimGUI boolean t
```

Directs the environment to open the GUI when the simulator runs.

ncsimLoadvpi through ncsimStatus

These `ams.env` `ncsim*` variables correspond to `ncsim` command options as follows:

ams.env Variable	ncsim Command Option
`ncsimLoadvpi`	`-LOADVPi`
`ncsimLogFileAction`	`-LOGfile`, `-NOLOg`, `-APPEND_Log`
`ncsimMaxErrors`	`-ERrormax`
`ncsimMessages`	`-Messages`
`ncsimNeverwarn`	`-NEverwarn`
`ncsimNocifcheck`	`-NOCIfcheck`
`ncsimNosource`	`-NOSOurce`
`ncsimNostdout`	`-NOSTdout`
`ncsimNowarn`	`-NOWarn`
`ncsimOmichecklvl`	`-Omicheckinglevel`
`ncsimPlinooptwarn`	`-PLINOOptwarn`
`ncsimPlinowarn`	`-PLINOWarn`
`ncsimProfile`	`-PROFIle`
`ncsimProfthread`	`-PROFThread`
`ncsimRedmem`	`-REdmem`
`ncsimStatus`	`-STATus`

For information about `ncsim` command options, see Chapter 9 of *NC-Verilog Simulator Help*.

ncsimTcl

Controls whether the simulator opens a Tcl command window. A Tcl command window allows text-based interaction with the simulator.

This variable has an effect only when the `ncsimGUI` variable is set to `nil`.

Syntax

```
amsDirect.prep ncsimTcl boolean t | nil
```

Values

`t`	Opens the Tcl command window when the simulator runs.
`nil`	Runs the simulation in batch mode.This is the default.

Example

```
amsDirect.prep ncsimTcl boolean t
```

If the `ncsimGUI` variable is set to `nil`, this example directs the environment to run the simulation in Tcl mode.

ncsimUnbuffered through ncsimUseAddArgs

These `ams.env` `ncsim*` variables correspond to `ncsim` command options as follows:

ams.env Variable	ncsim Command Option
`ncsimUnbuffered`	`-UNbuffered`
`ncsimUpdate`	`-UPdate`
`ncsimUseAddArgs`	None. Determines whether to use additional specified arguments on the `ncsim` command line.

For information about `ncsim` command options, see Chapter 9 of *NC-Verilog Simulator Help*.

ncvhdlArguments

Specifies arguments, in addition to the standard arguments, to be passed to the `ncvhdl` compiler.

Syntax

```
amsDirect.vhdl ncvhdlArguments string "arguments"
```

Value

arguments	One or more arguments to be passed to the `ncvhdl` compiler.

Description

By default, when the AMS netlister runs the compiler to compile a VHDL module, it uses the command

```
ncvhdl -use5x -work lib
```

where *lib* is the working library.

You can use the `ncvhdlArguments` variable to pass additional arguments to the compiler.

Example

If the working library is `myworklib`, then using the variable

```
amsDirect.vhdl ncvhdlArguments string "-status"
```

runs the `ncvhdl` compiler with the command

```
ncvhdl -use5x -work myworklib -status
```

ncvlogArguments

Specifies additional arguments to be passed to the `ncvlog` compiler.

Syntax

```
amsDirect.vlog ncvlogArguments string "arguments"
```

Value

arguments	One or more arguments to be passed to the `ncvlog` compiler. The default is an empty string.

Description

If *arguments* is an empty string, the `ncvlog` command includes just the arguments listed on the *Compiler* pane of the AMS Options window. You can use the `ncvlogArguments` variable with a non-empty string to pass additional arguments to the compiler.

Example

```
amsDirect.vlog ncvlogArguments string "-status"
```

Adds the `-status` argument to the other arguments normally used on the `ncvlog` command.

ncvlogUseAddArgs

Controls whether the additional compiler arguments specified by the `ncvlogArguments` variable are used on the `ncvlog` command.

Syntax

```
amsDirect.vlog xmvlogUseAddArgs boolean  t | nil
```

Values

`t`	The additional compiler arguments specified by the `ncvlogArguments` variable are used.
`nil`	The additional compiler arguments specified by the `ncvlogArguments` variable are *not* used. This is the default.

netClashFormat

Specifies the format to be used to map the names of nets that collide with names of other netlist constructs.

Syntax

```
amsDirect.vlog netClashFormat string "format"
```

Value

format	All characters, except those listed below, are printed as included in *format*. The following characters have the indicated special meanings. `%b` Original name of the net `%%` Prints the `%` character The default value of *format* is `%b_netclash`, which produces a mapped name like `nname_netclash` for a net originally named `nname`.

If the resulting name is illegal in Verilog-AMS, the name is mapped. If the mapped name clashes with the name of another object, the name undergoes collision mapping.

Example

```
amsDirect.vlog netClashFormat string "%b_nclash"
```

Tells AMS netlister to map clashing net names with a suffixed `_nclash`. For example, you have a net `samp` with a name that clashes with an instance named `samp`. The AMS netlister maps the net to the system-generated name `samp_nclash`.

netlistAfterCdfChange

Controls netlist generation for the cellview when the CDF information for the cell is updated from the CDF editor.

Syntax

```
amsDirect.vlog netlistAfterCdfChange boolean t | nil
```

Values

`t`	Generates netlists for the eligible cellviews of the cell after CDF information is updated (provided that no errors are found while checking CDF data).
`nil`	Does not generate a netlist. This is the default.

Description

```
amsDirect.vlog netlistAfterCdfChange boolean t
```

Tells the AMS netlister to generate a Verilog-AMS netlist for the cell whose CDF is being updated. However, the netlister does not generate a netlist if checking the CDF information reveals any errors.

netlistMode

Controls netlisting.

Syntax

```
amsDirect.prep netlistMode cyclic "none" | "incremental" | "all"
```

Values

`none`	Turns off netlisting.
`incremental`	Netlists cellviews in the hierarchy only if their HDL data is not synchronized with their cellview data. This is the default.
`all`	Netlists all cellviews in the hierarchy, regardless of whether their HDL data is synchronized with their cellview data.

Example

```
amsDirect.prep netlistMode cyclic "all"
```

Tells AMS Designer (working through the AMS netlister) to netlist all the cellviews that can be netlisted.

netlistToRunDir

Controls whether the software writes netlist files to the current run directory or to the master libraries specified by the `cds.lib` file. (Master libraries also include explicit TMP directories and TMP libraries that result from using the `ASSIGN AllLibs` statement in the `cds.lib` file.)

Note: This variable interacts with the `useRunDirNetlistsOnly` variable. Using the default values of the `netlistToRunDir` and `useRunDirNetlistsOnly` variables, the software writes netlists to the master libraries.

Syntax

```
amsDirect netlistToRunDir boolean t | nil
```

Values

`t`	Writes new netlists to the run directory. Use the `useRunDirNetlistsOnly` variable to specify whether you want netlists in master libraries also considered. **Note:** Set `netlistToRunDir` to `t` if you want to match the behavior of the Analog Design Environment (ADE).
`nil`	Writes new netlists to the master libraries that hold the netlisted schematics. This is the default value and the default behavior for AMS Designer.

Controlling Netlisting into Run Directories

Using the default values of the `netlistToRunDir` and `useRunDirNetlistsOnly` variables, the software writes netlists to the master libraries.

ADE has a different default behavior, where netlists are written to run directories. To match this behavior in AMS Designer, set `netlistToRunDir` to `t` (and use the default value for the `useRunDirNetlistsOnly` variable).

Alternatively, you can establish a behavior that facilitates using shared netlists located in master libraries without the need to update the `cds.lib` file with `TMP` assignments for libraries that have out-of-date netlists. In this shared netlist approach, netlists in master libraries can be generated once and then used in multiple configurations and simulations, and

by multiple users.To establish this behavior, set `netlistToRunDir` to `t` and set `useRunDirNetlistsOnly` to `nil`.

There is an additional difference to be aware of when you share netlists between ADE and AMS Designer. In AMS Designer, the default value for the `amsDefinitionViews` variable is `""` but in ADE, the default value is `"symbol schematic extracted"`. (See "amsDefinitionViews" on page 226.) Because the `amsDefinitionViews` setting determines module port ordering and bus details, using these default values means that a netlist created by AMS Designer differs from the netlist created by ADE for the same design.

If you use, for example, the *Tools – AMS – Netlist* command in the CIW to fill a master library with netlists and later, with `useRunDirNetlistsOnly` set to `nil`, you run design preparation, nothing more is netlisted. However, if you use the same master library in ADE, with `useRunDirNetlistsOnly` set to `nil`, and then netlist the design, all of the netlists are regenerated into the run directory. That difference in behavior occurs because the `amsDefinitionViews` variables in the two environment are different by default. The solution, which allows you to take full advantage of netlist sharing, is to set the `amsDefinitionViews` variable in the AMS Designer environment to `"symbol schematic extracted"`.

Example 1

You add the following to your `ams.env` file before running AMS Designer.

```
amsDirect netlistToRunDir boolean t
amsDirect useRunDirNetlistsOnly boolean t
```

The `netlistToRunDir` variable specifies that netlists (and other intermediate files produced by netlisting) are to be written to the run directory.

The `useRunDirNetlistsOnly` variable specifies that the program is to look for netlists only in the run directory. If a required netlist is not found, the netlist is created in the run directory and then used. To determine whether incremental netlisting is necessary, only netlists found in the run directory are considered and netlists that might exist in the master library are ignored.

The default value for `useRunDirNetlistsOnly` is `t`, so you could omit the second statement in the example and the behavior would be the same.

Example 2

You add the following to your `ams.env` file before running AMS Designer.

```
amsDirect netlistToRunDir boolean t
amsDirect useRunDirNetlistsOnly boolean nil
```

The `netlistToRunDir` variable specifies that netlists (and other intermediate files produced by netlisting) are to be written to the run directory.

The `useRunDirNetlistsOnly boolean nil` value specifies that the program is allowed to look for netlists in master libraries.

Example 3

You do not include the `netlistToRunDir` or `useRunDirNetlistsOnly` variables in your `ams.env` file or you set those variables to their default values.

```
amsDirect netlistToRunDir boolean nil
amsDirect useRunDirNetlistsOnly boolean t
```

The `nil` value for the `netlistToRunDir` variable indicates that all netlist information is assumed to be in and is written to the master libraries specified by the `cds.lib` file. You must have write permission to the master libraries (or have corresponding writable TMP directories), or netlisting fails. This behavior is the same behavior AMS Designer had before netlisting to the run directory was supported.

The `useRunDirNetlistsOnly` variable has no effect when `netlistToRunDir` is set to `nil`.

netlistUDFAsMacro

Determines whether user-defined functions (UDFs) are flagged as errors or are converted to macro references. AMS Designer does not provide a graphical interface for setting this variable.

Syntax

```
amsDirect.vlog netlistUDFAsMacro boolean t | nil
```

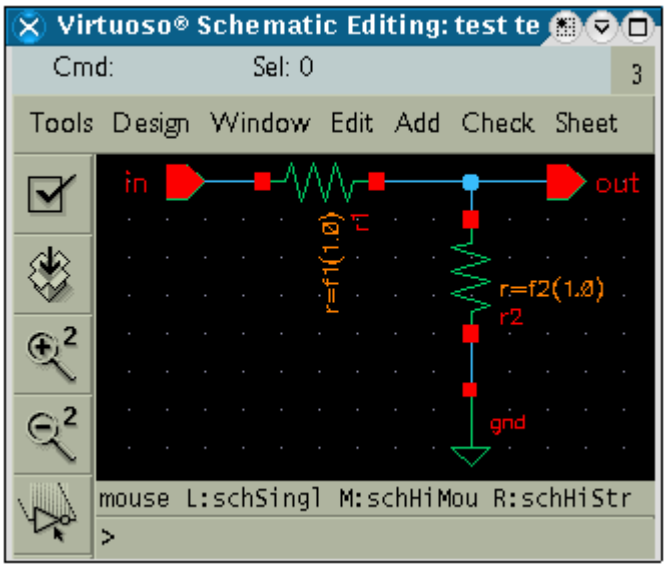
Values

`t`	Specifies that the netlister is to convert UDFs to macro references.
`nil`	Specifies that the netlister is to flag UDFs with errors and not produce a netlist. This is the default.

Description

```
amsDirect.vlog netlistUDFAsMacro boolean t
```

Tells the AMS netlister to convert UDFs to macro references. For example, the following schematic uses UDFs to specify the value of the resistors.



The netlister uses equivalent macro references in the netlist.

```
resistor #(.r(`f1(1.0)))  (* ... *) R1 ( ... );
resistor #(.r(`f2(1.0)))  (* ... *) R1 ( ... );
```

The referenced macros must be defined in an accessible location.

neverwarn

Suppresses all warning messages.

Syntax

```
amsDirect.vlog neverwarn boolean  t | nil
```

Values

`t`	Suppresses all warning messages.
`nil`	Warning messages are displayed. This is the default.

noline

Tells the compiler not to locate the source line of errors, potentially improving performance.

Syntax

```
amsDirect.vlog noline boolean  t | nil
```

Values

`t`	The compiler does *not* locate the source line of errors.
`nil`	The compiler locates the source line of errors. This is the default.

nomempack

Prepares design units for access by the PLI routine `tf_nodeinfo`.

Syntax

```
amsDirect.vlog nomempack boolean  t | nil
```

Values

`t`	Prepares design units for access by the PLI routine `tf_nodeinfo`.
`nil`	Does not prepare design units for access by the PLI routine `tf_nodeinfo`. This is the default.

Example

```
amsDirect.vlog nomempack boolean t
```

Tells AMS Designer to compile Verilog files with the `-nomempack` option. As a result, the generated command might look like this.

```
xmvlog -nomempack
```

nopragmawarn

Suppresses warning messages related to pragmas.

Syntax

```
amsDirect.vlog nopragmawarn boolean  t | nil
```

Values

`t`	Suppresses warning messages related to pragmas.
`nil`	Displays warning messages related to pragmas. This is the default.

nostdout

Suppresses printing of output to the screen but does not change what is written to the log file.

Syntax

```
amsDirect.vlog nostdout boolean  t | nil
```

Values

`t`	Suppresses printing of output to the screen.
`nil`	Prints output to the screen. This is the default.

nowarn

Suppresses warning messages that have specified codes.

Syntax

```
amsDirect.vlog nowarn string "msgcodes"
```

Value

msgcodes	The default is an empty string.

paramDefVals

Specifies a list of Verilog-AMS module parameters and their associated defaults.

Syntax

```
amsDirect.vlog paramDefVals string "{ ( [type:] parameter_name=value
    ) } "
```

Values

type	The type of *parameter_name*: `integer` or `real`.
parameter_name	A Verilog-AMS module parameter.
value	The default associated with *parameter_name*.

Description

The AMS netlister uses this list of parameters when it generates the parameter list for the cellview that is being netlisted and defaults for one or more of those parameters do not appear in the design data. This variable does not affect the generation of the list parameters that are passed into an instantiated cell.

The AMS netlister assumes that all parameter names are in the cellview name space.

The default for `paramDefVals` is an empty string.

Example

```
amsDirect.vlog paramDefVals string "(real:l=1.0)(count=0)(w=1.1)"
```

Specifies defaults for the parameters `l`, `count`, and `w`. The `l` parameter is specified as a real.

paramGlobalDefVal

Specifies a global module parameter default to be used when a CDF value is not available and the AMS netlister cannot find the parameter name in the `paramDefVals` variable.

Syntax

```
amsDirect.vlog paramGlobalDefVal string "value"
```

Value

value	Specifies the global module parameter default to be used. The default is 0.

Description

The AMS netlister uses this global value only when it generates the parameter list for the cellview that is being netlisted and defaults for one or more of those parameters do not appear in the design data. This variable does not affect the generation of the list parameters that are passed into an instantiated cell.

pragma

Parses pragmas contained in HDL source files.

Syntax

```
amsDirect.vlog pragma boolean  t | nil
```

Values

`t`	The compiler parses pragmas contained in HDL source files.
`nil`	The compiler does not parse pragmas contained in HDL source files.

Example

```
amsDirect.vlog pragma boolean t
```

Tells AMS Designer to compile Verilog files with the `-pragma` option. As a result, the generated command might look like this.

```
xmvlog -pragma
```

useRunDirNetlistsOnly

Controls whether the netlister considers netlists in the run directory only or also, if necessary, considers netlists it finds in the master libraries defined by `cds.lib` files. (For the purpose of this discussion, master libraries also include explicit TMP directories and TMP libraries that result from using the `ASSIGN AllLibs` statement in the `cds.lib` file.)

Note: The `useRunDirNetlistsOnly` variable has no effect when the `netlistToRunDir` variable is set to `nil`. Using the default values of the `netlistToRunDir` and `useRunDirNetlistsOnly` variables, the software writes netlists to the master libraries.

Syntax

```
amsDirect useRunDirNetlistsOnly boolean t | nil
```

Values

`t`	Specifies that the netlister is to consider, and, if necessary, update, netlists only from the run directory. Netlists in master libraries are ignored as the netlister determines whether incremental netlisting is needed for any particular object. This is the default.
`nil`	Specifies that the netlister is to first consider, and, if necessary, update, netlists found in the run directory. If a required netlist does not exist in the run directory, the netlister is then to consider netlists from the master libraries specified by the `cds.lib` file. If the necessary netlists are not found in the master libraries or is out of date, the netlister generates corresponding new netlists in the run directory.

Description

As noted, setting this variable to `nil` allows the program to consider netlists located in explicit TMP and master libraries. More specifically, a setting of `nil` means that the program is to

1. Look first for the netlist in the run directory.

   If the netlist is found and is up to date, use that netlist. If the netlist is found, but is not up to date, update it and then use it.

2. If the netlist is not found in the run directory, look in the explicit TMP directory (if one is used).

   If the netlist is found and is up to date, use that netlist. If the netlist is found but is not up to date, create an up-to-date netlist in the run directory and use it.

3. If the netlist is not found in the explicit TMP directory, look in the master library.

   If the netlist exists in the master library and is up to date, use that information. If the netlist is not found or is found but is not up to date, create an up-to-date netlist in the run directory and use it.

Example

See "Example 1" on page 311, "Example 2" on page 311, and "Example 3" on page 312.

processViewNames

Specifies the names of cellviews that are to be netlisted.

Syntax

```
amsDirect.vlog processViewNames string "list_of_view_names"
```

Value

list_of_view_names

A list of view names separated by spaces. Cellviews with these names are netlisted. The default is an empty string.

Description

The following conditions trigger netlisting.

- Changes to cellviews included in *list_of_view_names* while netlisting is enabled.
- Changes to the CDF of cells containing any of the cellviews included in *List_of_view_names* while the `netlistAfterCdfChange` variable is set to `t`.

Using this variable is an alternative to specifying the eligible view types with `amsEligibleViewTypes` and the views to exclude from netlisting with `excludeViewNames`.

Example

```
amsDirect.vlog processViewNames string "sch1 sch[3-4]"
```

prohibitCompile

Controls the automatic compilation of the generated netlist.

Syntax

```
amsDirect.vlog prohibitCompile boolean t | nil
```

Values

`t`	Prohibits the automatic compilation of the generated netlist.
`nil`	Automatically compiles the netlist. This is the default.

Description

By default, the AMS netlister automatically compiles the netlist. If you specify `t`, the netlister does not automatically compile the netlist.

runNcelab

Controls whether the elaborator runs when you click *Run* in the AMS Run Simulation form.

Syntax

```
amsDirect.prep runNcelab boolean t | nil
```

Values

`t`	Runs the elaborator. This is the default.
`nil`	Does not run the elaborator.

Example

```
amsDirect.prep runNcelab boolean nil
```

Tells AMS Designer not to run the elaborator.

runNcsim

Controls whether the simulator runs when you click *Run* in the AMS Run Simulation form.

Syntax

```
amsDirect.prep runNcsim boolean t | nil
```

Values

`t`	Runs the simulator. This is the default.
`nil`	Does not run the simulator.

Example

```
amsDirect.prep runNcsim boolean nil
```

Tells AMS Designer not to run the simulator when you click *Run* in the AMS Run Simulation form.

scaddlglblopts

Specifies options to be appended to the end of the `options` card in the generated simulation control file.

Syntax

```
amsDirect.simcntl scaddlglblopts string "options"
```

Value

options	A list of options separated by spaces.

Description

This variable specifies additional options to be added to the `options` statement in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl scaddlglblopts string "rawfile = \"/hm/kat/amsAnalysis\""
```

In response, the generated simulation control file contains

```
amsOptions options
+ gmin_check = all
+ inventory = detailed
+ rawfile = "/hm/kat/amsAnalysis"
```

scaddltranopts

Specifies additional options to be appended to the end of the `tran` card in the simulation control file.

Syntax

```
amsDirect.simcntl scaddltranopts string "options"
```

Value

options	A list of options separated by spaces.

Description

This variable specifies additional options to be appended to the `tran` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl scaddltranopts string "outputstart=0.0005"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ outputstart=0.0005
```

scale

Specifies the scaling factor for device instances.

Syntax

```
amsDirect.simcntl scale string "factor"
```

Value

factor	The scaling factor for device instances. The default is 1.0.

Description

This variable determines the value assigned to the `options scale` option in a generated simulation control file.

scalem

Specifies the scaling factor for models.

Syntax

```
amsDirect.simcntl scalem string "factor"
```

Value

factor	The scaling factor for models. The default is 1.0.

Description

This variable determines the value assigned to the `options scalem` option in a generated simulation control file.

scannotate

Specifies the degree of annotation for the transient analysis.

Syntax

`amsDirect.simcntl` **scannotate cyclic "sweep"** | **"no"** | **"title"** | **"status"** | **"steps"**

Values

`sweep`

`no`

`title`

`status`

`steps`

scapprox

Specifies that approximate models are to be used. There is a little difference between approximate and exact models.

Syntax

```
amsDirect.simcntl scapprox boolean t | nil
```

Values

`t`	The simulator uses approximate models.
`nil`	The simulator uses exact models. This is the default.

Description

This variable determines the value assigned to the `options approx` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scapprox        boolean t
```

In response, the generated simulation control file contains

```
amsOptions options
+ approx = yes
```

scaudit

Specifies the extent of the information to be returned about the time required by various parts of the simulation.

Syntax

```
amsDirect.simcntl scaudit cyclic "detailed" | "no" | "brief" |
    "full"
```

Values

`detailed`

`no`

`brief`

`full`

Description

This variable determines the value assigned to the `options audit` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scaudit cyclic  "full"
```

In response, the generated simulation control file contains

```
amsOptions options
+ audit = full
```

sccheckstmt

Unsupported by AMS Designer.

Performs a check analysis at any point in a simulation to be sure that the value of component parameters are reasonable. For more information, see “The check Statement” in the “Control Statements” chapter of the Spectre Classic Simulator, Spectre APS, Spectre X, and Spectre XPS User Guide.

sccmin

Specifies the minimum capacitance from each node to ground.

Syntax

```
amsDirect.simcntl sccmin string "capacitance"
```

Value

capacitance	The minimum capacitance from each node to ground.

Description

This variable determines the value assigned to the `tran cmin` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sccmin  string  "0.1"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ cmin = 0.1
```

sccompatible

Specifies a simulator. AMS Designer changes device models to improve consistency with the models in the specified simulator.

Syntax

```
amsDirect.simcntl sccompatible cyclic "spectre" | "spice2" |
    "spice3" | "hspice" | "spiceplus"
```

Values

`spectre`

`spice2`

`spice3`

`hspice`

`spiceplus`

Description

This variable determines the value assigned to the `options compatible` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sccompatible    cyclic  "spiceplus"
```

In response, the generated simulation control file contains

```
amsOptions options
+ compatible = spiceplus
```

scdebug

Prints debugging information.

Syntax

```
amsDirect.simcntl scdebug boolean t | nil
```

Values

`t`	The simulator prints debugging information.
`nil`	The simulator does *not* print debugging information. This is the default.

Description

This variable determines the value assigned to the `options debug` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scdebug boolean t
```

In response, the generated simulation control file contains

```
amsOptions options
+ debug = yes
```

scdiagnose

Prints information that might help diagnose accuracy and convergence problems.

Syntax

```
amsDirect.simcntl scdiagnose boolean t | nil
```

Values

`t`	The simulator prints diagnostic information.
`nil`	The simulator does *not* print diagnostic information. This is the default.

Description

This variable determines the value assigned to the `options diagnose` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scdiagnose      boolean t
```

In response, the generated simulation control file contains

```
amsOptions options
+ diagnose = yes
```

scdigits

Specifies the number of digits used when printing numbers.

Syntax

```
amsDirect.simcntl scdigits int digits
```

Value

digits	The number of digits used when printing numbers

Description

This variable determines the value assigned to the `options digits` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scdigits        int     8
```

In response, the generated simulation control file contains

```
amsOptions options
+ digits = 8
```

scerror

Prints error messages.

Syntax

```
amsDirect.simcntl scerror boolean t | nil
```

Values

`t`	The simulator prints error messages. This is the default.
`nil`	The simulator does *not* print error messages.

Description

This variable determines the value assigned to the `options error` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scerror boolean nil
```

In response, the generated simulation control file contains

```
amsOptions options
+ error = no
```

scerrpreset

Specifies a collection of parameter settings for the analysis. The collection you specify affects simulation speed and accuracy.

Syntax

```
amsDirect.simcntl scerrpreset cyclic "moderate" | "conservative" |
    "liberal"
```

Values

`moderate`	Simulation accuracy approximates a SPICE2 style simulator.
`conservative`	Simulation is the most accurate but also the slowest. This setting is appropriate for sensitive analog circuits.
`liberal`	Simulation is fast but less accurate. This setting is suitable for digital circuits or for analog circuits that have only short time constants.

Description

This variable determines the value assigned to the `tran errpreset` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scerrpreset     cyclic  "conservative"
```

In response, the generated simulation control file includes

```
amsAnalysis tran
+ stop = 0.001
+ errpreset = conservative
```

scfastbreak

Specifies the evaluation method to use for VHDL-AMS `break` statements.

Syntax

```
amsDirect.simcntl scfastbreak boolean t | nil
```

Values

`t`	Requests a method of evaluating VHDL-AMS `break` statements that is often faster than the default method. Under some circumstances, the method chosen by setting `scfastbreak` to `t` does not comply with the VHDL-AMS standard. possible non-compliance with the standard arises when the `break` statement is associated with a discontinuity that causes a zero-delay `Q'ABOVE` event. The `Q'ABOVE` event might be reported with a tiny delay, rather than the expected zero delay. This method might also produce simulation results that differ slightly from the results obtained when the default method is used.
`nil`	Requests the `break` statement evaluation method that complies strictly with the VHDL-AMS standard. This is the default.

Example

```
amsDirect.simcntl scfastbreak boolean t
```

Directs the simulator to use the potentially faster method of evaluating VHDL-AMS `break` statements.

scglobalminr

Specifies the threshold below which resistance inside devices is ignored.

Syntax

```
amsDirect.simcntl scglobalminr string "resistance"
```

Value

resistance	The threshold resistance.

Description

This variable determines the value assigned to the `options minr` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scglobalminr string  "1e-5"
```

In response, the generated simulation control file contains

```
amsOptions options
+ minr = 1e-5
```

scgmin

Specifies the minimum conductance across each nonlinear device.

Syntax

```
amsDirect.simcntl scgmin string "conductance"
```

Value

conductance	The minimum conductance.

Description

This variable determines the value assigned to the `options gmin` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scgmin  string  "1e-11"
```

In response, the generated simulation control file contains

```
amsOptions options
+ gmin = 1e-11
```

scgmincheck

Specifies how the effect of `scgmin` is to be reported.

Syntax

```
amsDirect.simcntl scgmincheck cyclic "max_v_only" | "max_only" |
    "no" | "all"
```

Values

`max_v_only`

`max_only`

`no`

`all`

Description

This variable determines the value assigned to the `options gmin_check` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scgmincheck     cyclic  "max_only"
```

In response, the generated simulation control file contains

```
amsOptions options
+ gmin_check = max_only
```

schomotopy

Specifies the method to use if convergence fails on the initial DC analysis attempt.

Syntax

```
amsDirect.simcntl schomotopy cyclic "all" | "none" | "gmin" |
    "source" |
    "dptran" | "ptran"
```

Values

`all`

`none`

`gmin`

`source`

`dptran`

`ptran`

Description

This variable determines the value assigned to the `options homotopy` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       schomotopy      cyclic  "source"
```

In response, the generated simulation control file contains

```
amsOptions options
+ homotopy = source
```

sciabstol

Specifies the absolute tolerance for differences in the computed values of the currents in the last two iterations of a solution.

Syntax

```
amsDirect.simcntl sciabstol string "tolerance"
```

Value

tolerance	The absolute tolerance for differences in the computed values of the currents.

Description

This variable determines the value assigned to the `options iabstol` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sciabstol       string  "1e-10"
```

In response, the generated simulation control file contains

```
amsOptions options
+ iabstol = 1e-10
```

scic

Controls the interaction of various methods of setting the initial conditions.

Syntax

```
amsDirect.simcntl scic cyclic "all" | "dc" | "node" | "dev"
```

Values

`all`	Uses both `ic` statements and `ic` parameters, and `ic` parameters override `ic` statements.
`dc`	Ignores any initial condition specifiers, and uses the DC solution.
`node`	Uses `ic` statements, and ignores `ic` parameters on capacitors and inductors.
`dev`	Uses `ic` parameters on capacitors and inductors, and ignores `ic` statements.

Description

This variable determines the value assigned to the `tran ic` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scic    cyclic  "node"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ ic = node
```

scicstmt

Specifies initial conditions for nodes and devices in the design.

Syntax

```
amsDirect.simcntl scicstmt string "ic_conditions"
```

Values

ic_conditions	A list of conditions for nodes and devices.

Description

This variable determines the value assigned to the `ic` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl scicstmt string "7=0 out=1 OpAmp1.comp=5 L1:1=1.0u"
```

In response, the generated simulation control file contains

```
ic 7=0 out=1 OpAmp1.comp=5 L1:1=1.0u
```

scignshorts

Tells the simulator to ignore shorted components silently.

Syntax

```
amsDirect.simcntl scignshorts boolean t | nil
```

Values

`t`	The simulator ignores shorted components silently.
`nil`	The simulator reports shorted components. This is the default.

Description

This variable determines the value assigned to the `options ignshorts` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scignshorts     boolean t
```

In response, the generated simulation control file contains

```
amsOptions options
+ ignshorts = yes
```

scinfo

Prints information messages.

Syntax

```
amsDirect.simcntl scinfo boolean t | nil
```

Values

`t`	The simulator prints information messages. This is the default.
`nil`	The simulator does *not* print information messages.

Description

This variable determines the value assigned to the `options info` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scinfo  boolean nil
```

In response, the generated simulation control file contains

```
amsOptions options
+ info = no
```

scinventory

Specifies the extent of the information to be returned about the components used in the simulation.

Syntax

```
amsDirect.simcntl scinventory cyclic "no" | "brief" | "detailed"
```

`no`

`brief`

`detailed`

Description

This variable determines the value assigned to the `options inventory` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scinventory     cyclic  "brief"
```

In response, the generated simulation control file contains

```
amsOptions options
+ inventory = brief
```

sclimit

Specifies the limiting algorithm used to aid DC convergence.

Syntax

```
amsDirect.simcntl sclimit cyclic "dev" | "delta" | "log"
```

Values

`dev`

`delta`

`log`

Description

This variable determines the value assigned to the `options limit` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sclimit cyclic  "delta"
```

In response, the generated simulation control file contains

```
amsOptions options
+ limit = delta
```

sclteratio

Specifies the ratio to use to compute LTE tolerances from Newton tolerance.

Syntax

```
amsDirect.simcntl sclteratio string "ratio"
```

Values

ratio	The ratio to use to compute LTE tolerances from Newton tolerance.

Description

This variable determines the value assigned to the `tran lteratio` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sclteratio      string  "8.0"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ lteratio = 8.0
```

scmacromod

Indicates that the circuit contains macromodels. Sometimes specifying this information improves performance.

Syntax

```
amsDirect.simcntl scmacromod boolean t | nil
```

Values

`t`	Indicates that the circuit contains macromodels.
`nil`	Indicates that the circuit does *not* contain macromodels. This is the default.

Description

This variable determines the value assigned to the `options macromodels` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scmacromod      boolean t
```

In response, the generated simulation control file contains

```
amsOptions options
+ macromodels = yes
```

scmaxiters

Specifies the maximum number of iterations per time step.

Syntax

```
amsDirect.simcntl scmaxiters int maxiters
```

Values

maxiters	The maximum number of iterations per time step.

Description

This variable determines the value assigned to the `tran maxiters` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scmaxiters      int     10
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ maxiters = 10
```

scmaxnotes

Specifies the maximum number of times any particular notice will be issued per analysis.

Syntax

```
amsDirect.simcntl scmaxnotes int maxnotes
```

Values

maxnotes	The maximum number of times any particular notice will be issued per analysis.

Description

This variable determines the value assigned to the `options maxnotes` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scmaxnotes      int     15
```

In response, the generated simulation control file contains

```
amsOptions options
+ maxnotes = 15
```

scmaxnotestologfile

scmaxrsd

Specifies the threshold below which parasitic node reduction occurs.

Syntax

```
amsDirect.simcntl scmaxrsd string "threshold"
```

Values

threshold	The default value is an empty string, `""` equivalent to a value of zero.

Example

```
amsDirect.simcntl scmaxrsd string "1e-8"
```

Tells the AMS simulator to remove parasitic nodes with resistances smaller than `1e-8`. The simulator then uses a linear correction to model the resistance.

scmaxstep

Specifies the maximum time step.

Syntax

```
amsDirect.simcntl scmaxstep string "maxstep"
```

Values

maxstep	The maximum time step.

Description

This variable determines the value assigned to the `tran maxstep` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scmaxstep       string  ".00002"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ maxstep = .00002
```

scmaxwarn

Specifies the maximum number of times any particular warning will be issued per analysis.

Syntax

```
amsDirect.simcntl scmaxwarn int maxwarn
```

Values

maxwarn	The maximum number of times any particular warning will be issued per analysis.

Description

This variable determines the value assigned to the `options maxwarn` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scmaxwarn       int     20
```

In response, the generated simulation control file contains

```
amsOptions options
+ maxwarns = 20
```

scmaxwarntologfile

scmethod

Specifies the integration method to use.

Syntax

```
amsDirect.simcntl scmethod cyclic "traponly" | "gear2" | "euler" |
    "trap" |
    "gear2only" | "trapgear2"
```

Values

`traponly`	Uses almost exclusively the trapezoidal rule method.
`gear2`	Uses the backward-Euler and second-order Gear method.
`euler`	Uses exclusively the backward-Euler method.
`trap`	Uses the backward-Euler and the trapezoidal rule methods.
`gear2only`	Uses almost exclusively Gear's second-order backward-difference method.
`trapgear2`	Allows all three integration methods to be used.

Description

This variable determines the value assigned to the `tran method` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scmethod        cyclic  "traponly"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ method = traponly
```

scmodelevaltype

Specifies whether standard SPICE-like equations or table (accelerated) models are used to evaluate bsim3v3 and bsim4 models.

Syntax

```
amsDirect.simcntl scmodelevaltype cyclic "s" | "a"
```

Values

`s`	Instructs the simulator not to use table models for any instances. This is the default.
`a`	Instructs the simulator to use table (accelerated) models whenever possible. This global option applies to the entire simulated design. You can override this instruction on specific model cards by setting `mos_method = s` as an option on those cards.

Description

This variable determines the value assigned to the `mos_method` option of the `options` statement in a generated analog simulation control file. AMS Designer writes the `mos_method` option to the analog simulation control file only when the `scusemodeleval` variable is set to `t`. For additional information, see “scusemodeleval” on page 404.

Example

You set the variables

```
amsDirect.simcntl       scusemodeleval boolean t
amsDirect.simcntl       scmodelvaltype cyclic  "a"
```

In response, the generated analog simulation control file contains

```
amsOptions options
+ mos_method = a
```

scmosvres

Specifies the voltage increment for the mosfet table model interpolation grid. Smaller values reduce the interpolation error, but might increase memory consumption. A value of 20mV is appropriate for analog circuits that are extremely sensitive to small model parameter variations, and subthreshold and substrate currents.

Syntax

```
amsDirect.simcntl scmosvres string "vresolution"
```

Values

vresolution	The default value is `0.05`.

Example

```
amsDirect.simcntl scmosvres string "0.02"
```

Sets the interpolation grid value to 20mV.

scnarrate

Narrates the simulation.

Syntax

```
amsDirect.simcntl scnarrate boolean t | nil
```

Values

`t`	The simulator narrates the simulation. This is the default.
`nil`	The simulator does *not* narrate the simulation.

Description

This variable determines the value assigned to the `options narrate` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scnarrate       boolean nil
```

In response, the generated simulation control file contains

```
amsOptions options
+ narrate = no
```

scnotation

Specifies the notation to be used when displaying real numbers.

Syntax

```
amsDirect.simcntl scnotation cyclic "eng" | "sci" | "float"
```

Values

`eng`	Uses engineering notation.
`sci`	Uses scientific notation.
`float`	Uses floating point notation.

Description

This variable determines the value assigned to the `options notation` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scnotation      cyclic  "sci"
```

In response, the generated simulation control file contains

```
amsOptions options
+ notation = sci
```

scnote

Prints notice messages.

Syntax

```
amsDirect.simcntl scnote boolean t | nil
```

Values

`t`	The simulator prints notice messages. This is the default.
`nil`	The simulator does *not* print notice messages.

Description

This variable determines the value assigned to the `options note` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scnote  boolean nil
```

In response, the generated simulation control file contains

```
amsOptions options
+ note = no
```

scopptcheck

Specifies that operating point parameters are to be checked against soft limits.

Syntax

```
amsDirect.simcntl scopptcheck boolean t | nil
```

Values

`t`	The operating point parameters are checked against soft limits.
`nil`	The operating point parameters are *not* checked against soft limits.

Description

This variable determines the value assigned to the `options opptcheck` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scopptcheck     boolean nil
```

In response, the generated simulation control file contains

```
amsOptions options
+ opptcheck = no
```

scpivabs

Specifies the absolute pivot threshold.

Syntax

```
amsDirect.simcntl scpivabs string "threshold"
```

Values

threshold	The absolute pivot threshold.

Description

This variable determines the value assigned to the `options pivabs` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scpivabs        string  "0.5"
```

In response, the generated simulation control file contains

```
amsOptions options
+ pivabs = 0.5
```

scpivotdc

Specifies that numeric pivoting be used on every iteration of DC analysis.

Syntax

```
amsDirect.simcntl scpivotdc boolean t | nil
```

Values

`t`	The simulator uses numeric pivoting on every iteration of DC analysis.
`nil`	The simulator does not use numeric pivoting on every iteration of DC analysis. This is the default.

Description

This variable determines the value assigned to the `options pivotdc` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scpivotdc       boolean t
```

In response, the generated simulation control file contains

```
amsOptions options
+ pivotdc = yes
```

scpivrel

Specifies the relative pivot threshold.

Syntax

```
amsDirect.simcntl scpivrel string "threshold"
```

Values

threshold	The relative pivot threshold.

Description

This variable determines the value assigned to the `options pivrel` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scpivrel        string  "1e-7"
```

In response, the generated simulation control file contains

```
amsOptions options
+ pivrel = 1e-7
```

scquantities

Specifies the extent of the information to be returned about quantities.

Syntax

```
amsDirect.simcntl scquantities cyclic "no" | "min" | "full"
```

Values

`no`

`min`

`full`

Description

This variable determines the value assigned to the `options quantities` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scquantities    cyclic  "full"
```

In response, the generated simulation control file contains

```
amsOptions options
+ quantities = full
```

screadic

Specifies a file that contains initial conditions.

Syntax

```
amsDirect.simcntl screadic string "icfile"
```

Values

icfile	The path and name of a file containing initial conditions.

Description

This variable determines the value assigned to the `tran readic` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl screadic string  "/usr1/test6/SAR_A2D/tutorial_run/icfile"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ readic = "/usr1/test6/SAR_A2D/tutorial_run/icfile"
```

screadns

Specifies a file that contains nodesets.

Syntax

```
amsDirect.simcntl screadns string "nsfile"
```

Values

nsfile	The path and name of a file that contains nodesets.

Description

This variable determines the value assigned to the `tran readns` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl screadns string  "/usr1/test6/SAR_A2D/tutorial_run/nsfile"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ readns = "/usr1/test6/SAR_A2D/tutorial_run/nsfile"
```

screlref

Specifies the reference to use for the relative convergence criteria.

Syntax

```
amsDirect.simcntl screlref cyclic "sigglobal" | "allglobal" |
    "pointlocal" | "alllocal"
```

Values

`sigglobal`	Compares relative errors in each of the circuit signals to the maximum for all signals at any previous point in time.
`allglobal`	Same as `sigglobal` except that it also compares the residues (KCL error) for each node to the maximum of that node's past history.
`pointlocal`	Compares the relative errors in quantities at each node to that node alone.
`alllocal`	Compares the relative errors at each node to the largest values found for that node alone for all past time.

Description

This variable determines the value assigned to the `tran relref` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       screlref        cyclic  "allglobal"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ relref = allglobal
```

screltol

Specifies the maximum relative tolerance for values computed in the last two iterations of a solution.

Syntax

```
amsDirect.simcntl screltol string "tolerance"
```

Values

tolerance	The maximum relative tolerance for values computed in the last two iterations of a solution. Default: `1e-3`

Description

This variable determines the value assigned to the `options reltol` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       screltol        string  "0.15"
```

In response, the generated simulation control file contains

```
amsOptions options
+ reltol = 0.15
```

scrforce

Specifies the resistance to be used when forcing nodesets and node-based initial conditions.

Syntax

```
amsDirect.simcntl scrforce string "resistance"
```

Values

resistance	The resistance to be used when forcing nodesets and node-based initial conditions.

Description

This variable determines the value assigned to the `options rforce` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scrforce        string  "1.5"
```

In response, the generated simulation control file contains

```
amsOptions options
+ rforce = 1.5
```

scscale

Specifies the scaling factor for device instances. This variable is obsolete: use the `scale` variable instead.

Syntax

```
amsDirect.simcntl scscale int factor
```

Values

factor	The scaling factor for device instances.

Description

This variable determines the value assigned to the `options scale` option in a generated simulation control file.

scscalem

Specifies the scaling factor for models. This variable is obsolete: use the `scalem` variable instead.

Syntax

```
amsDirect.simcntl scscalem int factor
```

Values

factor	The scaling factor for models.

Description

This variable determines the value assigned to the `options scalem` option in a generated simulation control file.

scscfincfile

Specifies a simulation control file to be included in the simulation control file generated from the options you specify in the GUI.

Syntax

```
amsDirect.simcntl scscfincfile string "sim_con_file"
```

Values

sim_con_file	The path and name of the simulation control file to be included.

Description

AMS Designer

Example

You set the variable

```
amsDirect.simcntl scscfincfile string  "/usr1/test6/SAR_A2D/tutorial_run/fpga.scs"
```

In response, the generated simulation control file contains

```
include "/usr1/test6/SAR_A2D/tutorial_run/fpga.scs"
```

scscftimestamp

A time stamp created by AMS Designer. Do not change this value manually.

Syntax

```
amsDirect.simcntl scscftimestamp string "timestamp"
```

Values

timestamp	A time stamp created by AMS Designer.

Description

AMS Designer uses this variable to track changes made in the simulation control file GUI.

Example

You use the GUI to create a simulation control file. You check the `ams.env` file and find that it contains a timestamp variable similar to the following.

```
amsDirect.simcntl       scscftimestamp  string  "1005580511000"
```

scscfusefileflag

Specifies that AMS simulator is to use an existing simulation control file, rather than a simulation control file created by the GUI.

Syntax

```
amsDirect.simcntl scscfusefileflag boolean t | nil
```

Values

`t`	The AMS simulator uses an existing simulation control file so the GUI for creating a new control file is disabled.
`nil`	The AMS simulator uses a simulation control file created by using the GUI, which is made active.

Description

AMS Designer

Example

You set the variable

```
amsDirect.simcntl       scscfusefileflag        boolean t
```

In response, the GUI for creating a new simulation control file is disabled, and a field is enabled that allows you to specify an existing control file.

scskipcount

Specifies a number of points and directs the simulator to save one point every time it calculates that number of points.

Syntax

```
amsDirect.simcntl scskipcount int skipcount
```

Values

skipcount	The number of points to be calculated for each saved point.

Description

This variable determines the value assigned to the `tran skipcount` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scskipcount     int     18
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ skipcount = 18
```

scskipdc

If `yes`, AMS Designer does not do any DC analysis for transient.

Syntax

```
amsDirect.simcntl scskipdc cyclic "no" | "yes" | "waveless" |
    "rampup" | "autodc"
```

Values

`no`	
`yes`	Skips the DC analysis entirely. The initial solution is the values given in the file you specify by the `screadic` variable, or, if that variable is not given, the values specified on `ic` statements.
`waveless`	
`rampup`	
`autodc`	

Description

This variable determines the value assigned to the `tran skipdc` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scskipdc        cyclic  "waveless"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ skipdc = waveless
```

scskipstart

Specifies a time. The simulator saves all computed data before this time.

Syntax

```
amsDirect.simcntl scskipstart string "time"
```

Values

time	The time before which all computed data is saved.

Description

This variable determines the value assigned to the `tran skipstart` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scskipstart     string  "0.01"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ skipstart = 0.01
```

scskipstop

Syntax

```
amsDirect.simcntl scskipstop string "0.0"
```

Values

The default is 0.0.

scspeed

Specifies the setting for the speed dial on the *Performance* pane of the AMS Options window. The *Speed dial* setting establishes the tradeoff between simulation performance and accuracy by writing the `options speed` parameter to the analog simulation control file. Higher settings result in better performance but with some loss in accuracy.

Syntax

```
amsDirect.simcntl scspeed int 0 | 1 | 2 | 3 | 4 | 5 | 6
```

Values

`0`	The `options speed` parameter is not written to the simulation control file, effectively turning the speed dial off and allowing the underlying settings to take their default values (unless they are individually overridden). This is the default.
`1`	Writes `options speed = 1` to the analog simulation control file.
`2`	Writes `options speed = 2` to the analog simulation control file.
`3`	Writes `options speed = 3` to the analog simulation control file.
`4`	Writes `options speed = 4` to the analog simulation control file.
`5`	Writes `options speed = 5` to the analog simulation control file.
`6`	Writes `options speed = 6` to the analog simulation control file.

The `scspeed` variable sets values for the following fields in the AMS Options window. If you then change the value in one of these fields, the new value overrides the value set by the `scspeed` variable.

Pane	Field	For more information, see
Tran Analysis	*Error preset*	“scerrpreset” on page 344

Pane	Field	For more information, see
Performance	*Node reduction threshold*	“scmaxrsd” on page 362
Convergence/Accuracy	*Reltol*	“screltol” on page 380
Convergence/Accuracy	*Vabstol*	“scvabstol” on page 405
Convergence/Accuracy	*Iabstol*	“sciabstol” on page 350
Tran Convergence/Accuracy	*Lteratio*	“sclteratio” on page 357
Tran Convergence/Accuracy	*Relref*	“screlref” on page 379
Tran Convergence/Accuracy	*Integration method*	“scmethod” on page 366
Tran Convergence/Accuracy	*Maxstep*	“scmaxstep” on page 363

Example

```
amsDirect.simcntl scspeed int 3
```

Causes the environment to write the following statements to the analog simulation control file.

```
amsOptions options
+ speed = 6
```

In addition, the fields affected by the `scspeed` variable change to reflect the corresponding values.

Field	Value
Error preset	`moderate`
Node reduction threshold	`<Value defaulted>`
Reltol	`<Value defaulted>`
Vabstol	`1e-6`
Iabstol	`1e-12`
Lteratio	`<Value defaulted>`
Relref	`<Default value>`
Integration method	`<Default value>`
Maxstep	`<Value defaulted>`

scspscflag

An internal variable used by AMS Designer. Do not change this variable manually.

Syntax

```
amsDirect.simcntl scspscflag boolean t | nil
```

Values

`t`

`nil`

Description

An internal variable used by AMS Designer.

scstats

Prints analysis statistics.

Syntax

```
amsDirect.simcntl scstats boolean t | nil
```

Values

`t`	The simulator prints analysis statistics.
`nil`	The simulator does *not* print analysis statistics. This is the default.

Description

This variable determines the value assigned to the `tran stats` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scstats boolean t
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ stats = yes
```

scstep

Specifies the minimum time step to use.

Syntax

```
amsDirect.simcntl scstep string "minstep"
```

Values

minstep	The minimum time step to use.

Description

This variable determines the value assigned to the `tran step` option in a generated simulation control file. You might need to set this value to maintain the aesthetics of computed waveforms.

Example

You set the variable

```
amsDirect.simcntl       scstep  string  ".00001"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ step = .00001
```

scstop

Specifies the stop time for the analysis.

Syntax

```
amsDirect.simcntl scstop string "stop_time"
```

Values

stop_time	The stop time.

Description

This variable determines the value assigned to the `stop` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scstop  string  "0.003"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
```

scstrobedelay

Specifies an offset time relative to the time specified by `scskipstart`.

Syntax

```
amsDirect.simcntl scstrobedelay string "offset_time"
```

Values

offset_time	The offset time.

Description

This variable determines the value assigned to the `tran strobedelay` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scstrobedelay   string  "0.00001"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ strobedelay = 0.00001
```

scstrobeperiod

Specifies an interval. The simulator calculates and saves a data point in each interval.

Syntax

```
amsDirect.simcntl scstrobeperiod string "interval"
```

Values

interval	The interval defining the strobe period.

Description

This variable determines the value assigned to the `tran strobeperiod` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scstrobeperiod  string  "0.0005"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ strobeperiod = 0.0005
```

sctemp

Specifies the circuit temperature.

Syntax

```
amsDirect.simcntl sctemp string "temperature"
```

Values

temperature	The circuit temperature in degrees Celsius.

Description

This variable determines the value assigned to the `options temp` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sctemp  string  "31.0"
```

In response, the generated simulation control file contains

```
amsOptions options
+ temp = 31.0
```

sctempeffects

Specifies what built-in primitive components are affected by circuit temperature.

Syntax

```
amsDirect.simcntl sctempeffects cyclic "all" | "vt" | "tc"
```

Values

`all`	All built-in temperature models are enabled.
`vt`	Only thermal voltage $V_t = \frac{kT}{q}$ can vary with temperature.
`tc`	In addition to thermal voltage, the component temperature coefficient parameters (parameters that start with `tc`, such as `tc1`, and `tc2`) are active. Use this setting when you want to disable the temperature effects for nonlinear devices.

Description

This variable determines the value assigned to the `options tempeffects` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sctempeffects   cyclic  "vt"
```

In response, the generated simulation control file contains

```
amsOptions options
+ tempeffects = vt
```

sctitle

Specifies a title for the analysis.

Syntax

```
amsDirect.simcntl sctitle string "title"
```

Values

title	The title to be associated with the analysis.

Description

This variable determines the value assigned to the `tran title` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sctitle string  "tran13"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.003
+ title = "tran13"
```

sctnom

Specifies the measurement (nominal) temperature.

Syntax

```
amsDirect.simcntl sctnom string "temperature"
```

Values

temperature	The measurement (nominal) temperature in degrees Celsius.

Description

This variable determines the value assigned to the `tnom` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sctnom  string  "31.0"
```

In response, the generated simulation control file contains

```
amsOptions options
+ tnom = 31.0
** UltraSim option settings **
*ultrasim: .usim_opt tnom = 31.0
```

sctopcheck

Specifies the extent of the error checking applied to the circuit topology.

Syntax

```
amsDirect.simcntl sctopcheck cyclic "full" | "min" | "no"
```

Values

`full`

`min`

`no`

Description

This variable determines the value assigned to the `options topcheck` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       sctopcheck      cyclic  "min"
```

In response, the generated simulation control file contains

```
amsOptions options
+ topcheck = min
```

scusemodeleval

Specifies whether the `mos_method` option is added to the `options` statement in generated analog simulation control files.

Syntax

```
amsDirect.simcntl scusemodeleval boolean t | nil
```

Values

`t`	AMS Designer adds the `mos_method` option to the `options` statement. For guidance on setting the value of the `mos_method` option, see "scmodelevaltype" on page 367.
`nil`	The simulator does *not* add the `mos_method` option to the `options` statement. This is the default.

Description

This variable, in conjunction with the `scmodelevaltype` variable, determines whether standard SPICE-like equations or table (accelerated) models are used to evaluate bsim3v3 and bsim4 models.

Example

You set the variables

```
amsDirect.simcntl       scusemodeleval boolean t
amsDirect.simcntl       scmodelevaltype cyclic "a"
```

In response, the generated simulation control file contains

```
amsOptions options
+ mos_method = a
```

scvabstol

Specifies the absolute tolerance for differences in the computed values of the voltages in the last two iterations of a solution.

Syntax

```
amsDirect.simcntl scvabstol string "tolerance"
```

Values

tolerance	The absolute tolerance for differences in the computed values of the voltages.

Description

This variable determines the value assigned to the `options vabstol` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scvabstol       string  "1e-8"
```

In response, the generated simulation control file contains

```
amsOptions options
+ vabstol = 1e-8
```

scwarn

Prints warning messages.

Syntax

```
amsDirect.simcntl scwarn boolean t | nil
```

Values

`t`	The simulator prints warning messages. This is the default.
`nil`	The simulator does *not* print warning messages.

Description

This variable determines the value assigned to the `options warn` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl       scwarn  boolean nil
```

In response, the generated simulation control file contains

```
amsOptions options
+ warn = no
```

scwrite

Specifies a file to which the simulator writes the initial transient solution.

Syntax

```
amsDirect.simcntl scwrite string "file"
```

Values

file	The path and name of a file to hold the initial transient solution.

Description

This variable determines the value assigned to the `tran write` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl scwrite string "/usr1/tutorial_run/writeinitial"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ write = "/usr1/tutorial_run/writeinitial"
```

scwritefinal

Specifies a file to which the simulator writes the final transient solution.

Syntax

```
amsDirect.simcntl scwritefinal string "file"
```

Values

file	The path and name of a file to hold the final transient solution.

Description

This variable determines the value assigned to the `tran writefinal` option in a generated simulation control file.

Example

You set the variable

```
amsDirect.simcntl scwritefinal string "/usr1/tutorial_run/writefinal"
```

In response, the generated simulation control file contains

```
amsAnalysis tran
+ stop = 0.001
+ writefinal = "/usr1/tutorial_run/writefinal"
```

simcompat

Specifies whether simulation values are to be set for compatibility with Spectre syntax or with HSPICE syntax.

Syntax

```
amsDirect.simcntl simcompat cyclic "spectre" | "hspice"
```

Values

`spectre`	The simulation values are set for compatibility with Spectre syntax. This is the default.
`hspice`	The simulation values are set for compatibility with HSPICE syntax.

Description

This variable determines whether the `-simcompatible_ams hspice` option or the `-simcompatible_ams spectre` option is added to the `xmsim` command. Notice that the default is always `spectre`, regardless of the solver that you are using.

Example

You set the variable

```
amsDirect.simcntl simcompat cyclic "hspice"
```

In response, the generated `xmsim` command includes

```
xmsim -simcompatible_ams hspice
```

simRunDirLoc

Specifies the default directory to contain run directories.

Syntax

```
amsDirect simRunDirLoc string "location"
```

Values

location	The path and name of the default directory to contain run directories. The *location* string can contain shell environment variables. If location contains a relative path, the path is evaluated relative to the directory where the Cadence software (for example, `icms` or `cdsHierEditor`) is started. The default value for *location* is an empty string, which means that the current working directory is the default directory to contain run directories.

Description

The AMS Designer environment allows you to designate one or more run directories. You make those designations relative to the directory specified by the `simRunDirLoc` variable.

Example

```
amsDirect simRunDirLoc string "$PROJECT/$BLOCK"
```

If the `simRunDirLoc` variable is set as shown, and the shell variables `$PROJECT` and `$BLOCK` are set to `/newChip` and `comparator`, respectively, the default directory to contain run directories is set to `/newChip/comparator`.

simVisScriptFile

Specifies a script file to be run at the beginning of simulation.

Syntax

```
amsDirect.prep simVisScriptFile string "script_file"
```

Values

script_file	The script file to be run at the beginning of simulation. If *script_file* uses a relative path, the xmsim program looks for the file relative to the run directory. The default is an empty string.

Example

```
amsDirect.prep simVisScriptFile string "demo.tcl"
```

Tells the AMS simulator to run the `demo.tcl` script before starting the simulation.

status

Syntax

```
amsDirect.vlog status boolean  t | nil
```

Values

`t`	
`nil`	This is the default.

templateFile

Specifies a file whose contents are to be incorporated into the header of newly generated netlists.

Syntax

```
amsDirect.vlog templateFile string "text_file"
```

Values

text_file	Specifies the path and filename of a text file whose contents are to be used in netlist headers. The default is an empty string. The file contents are incorporated into the netlist header only when the `headerText` variable has the value `"file"`. For more information, see headerText on page 260.

Example

Specifying the variable

```
amsDirect.vlog templateFile string "./ASICheader"
```

where the file named `ASICheader` contains the following text

```
// Module produced by
// ASIC Team: Ocelot
// San Jose Development Center
```

inserts lines similar to the following at the top of each newly generated netlist.

```
// Verilog-AMS netlist generated by the AMS netlister, version 4.4.6.100.43.
// Cadence Design Systems, Inc.
// Module produced by
// ASIC Team: Ocelot
// San Jose Development Center
```

templateScript

Specifies a file whose contents are a script. The results produced when the script runs are to be incorporated into the header of newly generated netlists.

Syntax

```
amsDirect.vlog templateScript string "script_file"
```

Values

script_file	Specifies the path and filename of a script file. The results produced when the script runs are to be used in netlist headers. The default is an empty string. The results are incorporated into the netlist header only when the `headerText` variable has the value `"script"`. For more information, see headerText on page 260.

Example

Specifying the variable

```
amsDirect.vlog templateScript string "./CRheader"
```

where the file named `CRheader` contains the following script

```
echo '// Module produced by:'
echo '// ASIC Interactive, Ltd.'
printf '// (c) '
date '+DATE: %m/%d/%y%n'
```

inserts lines similar to the following at the top of each newly generated netlist.

```
// Verilog-AMS netlist generated by the AMS netlister, version 4.4.6.100.43.
// Cadence Design Systems, Inc.
// Module produced by:
// ASIC Interactive, Ltd.
// (c) DATE: 10/10/01
```

timescale

Specifies the default timescale for Verilog (digital) modules. The `timescale` variable has no effect on analog behavior.

Syntax

```
amsDirect.prep timescale string "time_unit/time_precision"
```

Values

time_unit	The units of time to use. The default is `1ns`.
time_precision	The time precision required. The default is `1ns`.

Example

```
amsDirect.prep timescale string "2ns/2ns"
```

Tells the simulator to use `2ns` as the basic unit of time and to calculate time values with a precision of `2ns`.

update

Recompiles the design after design units, source files, or compiler directives are added, or if a design unit is changed in a way that introduces a new cross-file dependency.

Syntax

```
amsDirect.vlog update boolean  t | nil
```

Values

`t`	This is the default.
`nil`	

use5xForVHDL

Controls whether configurations apply to VHDL as well as Verilog-AMS.

Syntax

```
amsDirect.prep use5xForVHDL boolean t | nil
```

Values

`t`	Assumes that configurations apply to VHDL as well as Verilog-AMS. This is the default.
`nil`	Assumes that configurations do not apply to VHDL.

Description

If configurations apply to VHDL, the configurations take precedence over VHDL default binding and other searches. For more information, see the “-USe5x4vhdl Option” section of Chapter 7, in the *Spectre AMS Designer and Xcelium Simulator Mixed-Signal User Guide*.

useDefparam

Controls the netlisting of parameters passed onto instantiated modules.

Syntax

```
amsDirect.vlog useDefparam boolean t | nil
```

Values

`t`	Generates one `defparam` statement for each instance that requires passed parameters.
`nil`	Passes parameters by assigning instance parameter values. This is the default.

Description

The AMS netlister passes parameters by assigning instance parameter values; it passes parameters by name to child instances. Note digital simulators do not support passing parameters via instance parameter value assignments.

Another way to pass parameters is to use a `defparam` statement. If you specify `t`, the AMS netlister uses the `defparam` statement to pass parameters instead. One `defparam` statement is generated for each instantiation that requires passed parameters.

Example

```
amsDirect.vlog useDefparam boolean nil
```

Tells the AMS netlister to pass parameters by assigning instance parameter values. The following netlist results:

```
module mybuf (a, b);
    input a;
    output b;

    myinv #(.setup(10), .hold(5)) i0 (a, net10);
    myinv #(.setup(10), .hold(5)) i1 (net10, b);
endmodule
```

Example

```
amsDirect.vlog useDefparam boolean t
```

Tells the AMS netlister to pass parameters by using `defparam` statements. The following netlist results:

```
module mybuf (a, b);
    input a;
    output b;

    myinv i0(a, net10);
        defparam i0.setup = 10, i0.hold = 5;
    myinv i1(net10, b);
        defparam i1.setup = 10, i1.hold = 5;

endmodule
```

useEffectiveCDF

Controls whether the netlister uses effective or base CDF values.

The `useEffectiveCDF` variable has an effect only when the `netlistToRunDir` and `useRunDirNetlistsOnly` variables are both set to `t`.

Syntax

```
amsDirect useEffectiveCDF boolean t | nil
```

Values

`t`	Specifies that effective CDF values are to be used.
`nil`	Specifies that base CDF values are to be used. This is the default.

Example

```
amsDirect netlistToRunDir boolean t
amsDirect useRunDirNetlistsOnly boolean t
amsDirect useEffectiveCDF boolean t
```

All three necessary variables are set to `t` so the netlister uses the effective CDF values, which are the base CDF values updated with user CDF values.

useNcelabNowarn

Controls whether the list of suppressed warnings on the *Elaborator Messages/Errors* pane of the AMS Options window is used.

Syntax

```
amsDirect.prep useNcelabNowarn boolean t | nil
```

Values

`t`	Places a checkmark next to the *Suppress specific warnings* field, indicating that the listed warnings are to be suppressed. This is the default.
`nil`	Removes the checkmark, indicating that any listed warnings are not to be suppressed.

Example

```
amsDirect.prep useNcelabNowarn boolean nil
```

Removes the checkmark next to the *Suppress specific warnings* field. As a result, the `-nowarn` option of the `xmelab` command is not used and the listed warnings are not suppressed during elaboration.

useNcelabSdfCmdFile

Controls whether an SDF command file specified on the *SDF Annotation* pane of the AMS Options window is used.

Syntax

```
amsDirect.prep useNcelabSdfCmdFile boolean t | nil
```

Values

`t`	Places a checkmark next to the *Use SDF command file* field, indicating that the command file (if one is specified) is to be used during elaboration. This is the default.
`nil`	Removes the checkmark, indicating that any SDF command file that might be specified in the *Use SDF command file* field is not to be used.

Example

```
amsDirect.prep useNcelabSdfCmdFile boolean nil
```

Tells the elaborator to use an SDF command file if one is specified in the *Use SDF command file* field.

useNcsimNowarn

Controls whether the list of suppressed warnings on the *Simulator Messages/Errors* pane of the AMS Options window is used.

Syntax

```
amsDirect.prep useNcsimNowarn boolean t | nil
```

Values

`t`	Places a checkmark next to the *Suppress specific warnings* field, indicating that the listed simulation warnings are to be suppressed. This is the default.
`nil`	Removes the checkmark, indicating that any listed warnings are not to be suppressed.

Example

```
amsDirect.prep useNcsimNowarn boolean nil
```

Removes the checkmark next to the *Suppress specific warnings* field. As a result, the `-nowarn` option of the `xmsim` command is not used and the listed warnings are not suppressed during simulation.

useNowarn

Controls whether the list of suppressed warnings on the *Verilog-AMS Messages/Errors* pane of the AMS Options window is used.

Syntax

```
amsDirect.vlog useNowarn boolean t | nil
amsDirect.vhdl useNowarn boolean t | nil
```

Values

`t`	Places a checkmark next to the *Suppress specific warnings* field, indicating that the listed compilation warnings are to be suppressed. This is the default.
`nil`	Removes the checkmark, indicating that any listed warnings are not to be suppressed.

Example

```
amsDirect.vlog useNowarn boolean nil
```

Removes the checkmark next to the *Suppress specific warnings* field. As a result, the `-nowarn` option of the `xmvlog` command is not used and the listed warnings are not suppressed during compilation.

useProcessViewNamesOnly

Controls how the AMS netlister determines which cellviews to process.

Syntax

```
amsDirect.vlog useProcessViewNamesOnly boolean t | nil
```

Values

`t`	The AMS netlister determines which cellviews to process by consulting the `processViewNames` list.
`nil`	The AMS netlister determines which cellviews to process by consulting, in combination, the `amsEligibleViewTypes` list and the `excludeViewNames` list. This is the default.

Description

This variable determines which of two methods is used to select the views that are processed by the AMS netlister.

Example

Given the following values:

```
amsDirect.vlog amsEligibleViewTypes        string "schematic"
amsDirect.vlog excludeViewNames            string "sch[0-3]"
amsDirect.vlog processViewNames            string "sch1 sch[3-4]"
amsDirect.vlog useProcessViewNamesOnly     boolean nil
```

If the AMS netlister runs, for example, in response to a CDF save trigger on cell `mycell`, which has the six schematic views `sch0`, `sch1`, `sch2`, `sch3`, `sch4`, and `sch5`, only the `sch4` and `sch5` views are processed.

On the other hand, if `useProcessViewNamesOnly` is set to `t`, only the `sch1`, `sch3`, and `sch4` views are processed.

useScaddlglblopts

Controls whether the list of additional options on the *Analog Solver* pane of the AMS Options window is used.

Syntax

```
amsDirect.simcntl useScaddlglblopts boolean t | nil
```

Values

`t`	Places a checkmark next to the *Additional options* field, indicating that the listed options are to be passed to the analog solver. This is the default.
`nil`	Removes the checkmark, indicating that any listed additional options are to be ignored.

Example

```
amsDirect.simcntl useScaddlglblopts boolean nil
```

Removes the checkmark next to the *Additional options* field. As a result, any options listed in the field are not used by the analog solver.

useScaddltranopts

Controls whether the list of additional options on the *Tran Analysis* pane of the AMS Options window is used.

Syntax

```
amsDirect.simcntl useScaddltranopts boolean t | nil
```

Values

`t`	Places a checkmark next to the *Additional options* field, indicating that the listed options are to be used for transient analysis. This is the default.
`nil`	Removes the checkmark, indicating that any listed additional options are to be ignored.

Example

```
amsDirect.simcntl useScaddltranopts boolean nil
```

Removes the checkmark next to the *Additional options* field. As a result, any options listed in the field are not used during transient analysis.

useScic

Controls whether the list of initial conditions on the *Tran Convergence/Accuracy* pane of the AMS Options window is used.

Syntax

```
amsDirect.simcntl useScic boolean t | nil
```

Values

`t`	Places a checkmark next to the *Set initial conditions* field, indicating that the listed initial conditions are to be used for transient analysis. This is the default.
`nil`	Removes the checkmark, indicating that any listed initial conditions are to be ignored.

Example

```
amsDirect.simcntl useScic boolean nil
```

Removes the checkmark next to the *Set initial conditions* field. As a result, any conditions listed in the field are not used during transient analysis.

useScreadic

Controls whether initial conditions are read from the file specified on the *Tran Convergence/ Accuracy* pane of the AMS Options window.

Syntax

```
amsDirect.simcntl useScreadic boolean t | nil
```

Values

`t`	Places a checkmark next to the *Read IC from file* field, indicating that initial conditions are to be read from the specified file. This is the default.
`nil`	Removes the checkmark, indicating that initial conditions are not to be read from the specified file.

Example

```
amsDirect.simcntl useScreadic boolean nil
```

Removes the checkmark next to the *Read IC from file* field. As a result, initial conditions that might be specified in the file are not used.

useScreadns

Controls whether nodesets are read from the file specified on the *Tran Convergence/ Accuracy* pane of the AMS Options window.

Syntax

```
amsDirect.simcntl useScreadns boolean t | nil
```

Values

`t`	Places a checkmark next to the *Read nodesets from file* field, indicating that nodesets s are to be read from the specified file. This is the default.
`nil`	Removes the checkmark, indicating that nodesets are not to be read from the specified file.

Example

```
amsDirect.simcntl useScreadns boolean nil
```

Removes the checkmark next to the *Read nodesets from file* field. As a result, nodesets that might be specified in the file are not used.

useScscfincfile

Controls whether a simulation control file specified on the *Analog Solver* pane of the AMS Options window is included when the analog simulation control file runs.

Syntax

```
amsDirect.simcntl useScscfincfile boolean t | nil
```

Values

`t`	Places a checkmark next to the *Include simulation control file* field, indicating that the file is to be included when the analog simulation control file runs. This is the default.
`nil`	Removes the checkmark, indicating that the specified simulation control file is not to be included when the analog simulation control file runs.

Example

```
amsDirect.simcntl useScscfincfile boolean nil
```

Removes the checkmark next to the *Include simulation control file* field. As a result, any simulation control file that might be specified in the field is not included when the analog simulation control file runs.

useScwrite

Controls whether the initial solution is written to the file specified on the *Tran Output* pane of the AMS Options window.

Syntax

```
amsDirect.simcntl useScwrite boolean t | nil
```

Values

`t`	Places a checkmark next to the *Write initial solution to file* field, indicating that the initial solution is to be written to the specified file. This is the default.
`nil`	Removes the checkmark, indicating that the initial solution is not to be written to the specified file.

Example

```
amsDirect.simcntl useScwrite boolean nil
```

Removes the checkmark next to the *Write initial solution to file* field. As a result, the initial solution is not written to the file.

useScwritefinal

Controls whether the final solution is written to the file specified on the *Tran Output* pane of the AMS Options window.

Syntax

```
amsDirect.simcntl useScwritefinal boolean t | nil
```

Values

`t`	Places a checkmark next to the *Write final solution to file* field, indicating that the final solution is to be written to the specified file. This is the default.
`nil`	Removes the checkmark, indicating that final solution is not to be written to the specified file.

Example

```
amsDirect.simcntl useScwritefinal boolean nil
```

Removes the checkmark next to the *Write final solution to file* field. As a result, the final solution is not written to the file.

useSimVisScriptFile

Controls whether a Tcl input script specified on the *Simulator* pane of the AMS Options window is used.

Syntax

```
amsDirect.prep useSimVisScriptFile boolean t | nil
```

Values

`t`	Places a checkmark next to the *Tcl input script* field, indicating that the script (if one is specified) is to be used to control the simulator. This is the default.
`nil`	Removes the checkmark, indicating that any script that might be specified in the *Tcl input script* field is not to be used.

Example

```
amsDirect.prep useSimVisScriptFile boolean nil
```

Tells the simulator not to ignore any script that might be specified in the *Tcl input script* field.

usimAbstoli through usimWFTres

These `ams.env usim*` variables correspond to UltraSim circuit simulator command options as follows:

ams.env Variable	UltraSim Option
`usimAbstoli`	`.usim_opt abstoli`
`usimAbstolv`	`.usim_opt abstolv`
`usimAddlOptions`	`.usim_opt` followed by the specified options.
`usimAnalog`	`.usim_opt analog`
`usimCapFile`	`.usim_opt capfile`
`usimCgnd`	`.usim _opt cgnd`
`usimCgndr`	`.usim_opt cgndr`
`usimDCMethod`	`.usim_opt dc`
`usimDcut`	`.usim_opt dcut`
`usimDcutField`	`.usim_opt dcut`
`usimDiodeMethod`	`.usim_opt diode_method`
`usimDpfFile`	`.usim_opt dpf`
`usimDumpStep`	`.usim_opt dump_step`
`usimenableNA`	None. Allows UltraSim Advanced Checks Node Activity Analysis information to be added to the control file.
`usimenablePA`	None. Allows UltraSim Advanced Checks Power Analysis information to be added to the control file.
`usimenableRA`	None. Allows UltraSim Advanced Checks Analysis information to be added to the control file.
`usimenableTA`	None. Allows UltraSim Advanced Checks Timing Analysis information to be added to the control file.
`usimLshort`	`.usim_opt lshort`
`usimLvshort`	`.usim_opt lvshort`
`usimMaxstep`	`.usim_opt maxstep`
`usimMaxstepStart`	`.usim_opt maxstep_start`

ams.env Variable	UltraSim Option
usimMaxstepStop	.usim_opt maxstep_stop
usimMaxstepSubckt	.usim_opt maxstep
usimMosMethod	.usim_opt mos_method
usimNALimit	.usim_nact limit
usimNAOutputSort	.usim_nact max_vo
usimNASortIs	.usim_nact
usimOutputStart	tran outputstart
usimPostl	.usim_opt postl
usimRAAgeDomain	.agemethod
usimRAAgeMethod	.agemethod
usimRAAgeproc	.ageproc
usimRAAgingTime	.age
usimRADeltaD	.deltad
usimRADeltaDToggle	.deltad
usimRAMinAge	.minage
usimRAMode	.hci_only, .nbti_only, .nbtiageproc
usimRANBTIAgeproc	.nbtiageproc
usimRcrfmax	.usim_opt rcr_fmax
usimRshort	.usim_opt Rshort
usimRvshort	.usim_opt Rvshort
usimSimMode	.usim_opt sim_mode
usimSpeed	.usim_opt speed
usimSpefFile	.usim_opt spef
usimSpfFile	.usim_opt spf
usimTol	.usim_opt tol
usimTranAddlOptions	tran followed by the specified options.
usimUseAddlOptions	None. Allows additional UltraSim options to be used.

ams.env Variable	UltraSim Option
usimVcdFile	.vcd
usimVcdInfoFile	.vcd
usimVectorFile	.vec
usimWFAbstoli	.usim_opt wf_abstoli
usimWFAbstolv	.usim_opt wf_abstolv
usimWFFilter	.usim_opt wf_filter
usimWFReltol	.usim_opt wf_reltol
usimWFTres	.usim_opt wf_tres

For information about UltraSim circuit simulator command options, see the *Virtuoso UltraSim Simulator User Guide*.

verboseUpdate

Controls whether the names of already up-to-date modules are included in the log file generated for an update compilation.

Syntax

```
amsDirect.vlog verboseUpdate boolean t | nil
```

Values

`t`	Places a checkmark next to the *Print verbose messages during update* field on the *Verilog-AMS* pane of the AMS Options window. This tells the compiler to print the names of already up-to-date cells in the log, while updating cells. This is the default.
`nil`	Removes the checkmark, indicating that the names of up-to-date cells are not to be printed in the log, while updating cells.

Example

```
amsDirect.vlog verboseUpdate boolean t
```

Example

```
amsDirect.simcntl useScaddltranopts boolean nil
```

Removes the checkmark next to the *Print verbose messages during update* field. As a result, the names of up-to-date cells do not appear in the log.

vlogGroundSigs

Specifies which signals are to be declared as ground.

Syntax

```
amsDirect.prep vlogGroundSigs string "signal_list"
```

Values

signal_list	A list of signals to be declared, by default, as ground. The default is `gnd!`.

Description

AMS Designer uses the value of this variable to determine which wires should be declared as ground.

Example

For example, if the variable is defined like

```
amsDirect.prep vlogGroundSigs string "gnd! gnd2!"
```

then AMS Designer declares any new global signals named `gnd!` and `gnd2!` to be a ground.

vloglinedebug

Enables support for setting line breakpoints and for single-stepping through code.

Syntax

```
amsDirect.vlog vloglinedebug boolean  t | nil
```

Values

`t`	
`nil`	This is the default.

Description

Example

```
amsDirect.vlog vloglinedubug boolean t
```

Tells AMS Designer to compile Verilog files with the `-linedebug` option. As a result, the generated command might look like this.

```
xmvlog -linedebug
```

vlogSupply0Sigs

Specifies which signals are to be declared as supply0 wire types.

Syntax

```
amsDirect.prep vlogSupply0Sigs string "signal_list"
```

Values

signal_list	A list of signals to be declared, by default, as supply0 wires. The default is an empty string.

Description

AMS Designer uses the value of this variable to determine which wires should be declared as supply0 wire types.

Example

For example, if the variable is defined like

```
amsDirect.prep vlogSupply0Sigs string "vss! vss2!"
```

then AMS Designer declares any new global signals named `vss!` and `vss2!` to be a supply0 wire type.

vlogSupply1Sigs

Specifies which signals are to be declared as supply1 wire types.

Syntax

```
amsDirect.prep vlogSupply1Sigs string "signal_list"
```

Values

signal_list	A list of signals to be declared, by default, as supply1 wires. The default is an empty string.

Description

AMS Designer uses the value of this variable to determine which wires should be declared as supply1 wire types.

Example

For example, if the variable is defined like

```
amsDirect.prep vlogSupply1Sigs string "vdd! vdd2!"
```

then AMS Designer declares any new global signal named `vdd!` and `vdd2!` to be a supply1 wire type.

wfDefaultDatabase

Specifies the name of the default database for waveform data produced by the simulator. This name appears in the *Default database name* field, on the *Waveforms* pane of the AMS Options window. It also appears as a database in the AMS Databases window and is used as the default

Syntax

```
amsDirect.prep wfDefaultDatabase string "database"
```

Values

database	The name of the default database for waveform data. The default name is `waves`.

Example

```
amsDirect.prep wfDefaultDatabase string "fast_db"
```

Specifies that the `fast_db` database is to be the default database for waveform data produced by the simulator. This name `fast_db` appears in the AMS Databases window and is the default database for new selections in the AMS Save/Plot window.

wfDefInstCSaveAll

Specifies whether current probes are to be created for all levels of the instances selected from the schematic or the navigator.

Syntax

```
amsDirect.prep wfDefInstCSaveAll boolean t | nil
```

Values

`t`	Specifies that current probes are to be created for all levels of the selected instances. This choice is indicated in the AMS Save/Plot list by the word `all` appearing in the *Depth* column for the selected instances.
`nil`	Specifies that current probes for the selected instances are to be created only for the number of levels specified by the `amsDirect.prep wfDefInstCSaveLvl` variable. This is the default.

Example

```
amsDirect.prep wfDefInstCSaveAll boolean t
```

Specifies that current probes are to be created for all levels of the selected instances.

wfDefInstCSaveLvl

Specifies that current probes for the specified number of levels are to be created for instances selected from the schematic or the navigator.

Syntax

```
amsDirect.prep wfDefInstCSaveLvl int level
```

Values

level	The number of levels of current probes to be created for the selected instances. The default value is 1.

Example

```
amsDirect.prep wfDefInstCSaveLvl int 2
```

Specifies that current probes are to be created for two levels of each of the selected instances. In the AMS Save/Plot list, the *Depth* column for the selected instances contains the value `2`.

wfDefInstSaveCurrents

Controls whether current probes are created for the objects selected from the schematic or the navigator.

Syntax

```
amsDirect.prep wfDefInstSaveCurrents boolean t | nil
```

Values

`t`	Places a checkmark next to the *Currents at terminals or ports* label in the *Waveforms* pane of the AMS Options window, indicating that current probes are to be created for the objects selected from the schematic or the navigator.
`nil`	Removes the checkmark next to the *Currents at terminals or ports* label in the *Waveforms* pane of the AMS Options window, indicating that current probes are not to be created. This is the default.

Example

```
amsDirect.prep wfDefInstSaveCurrents boolean t
```

Places a checkmark next to the *Currents at terminals or port* labels, indicating that current probes are to be created for objects selected from the schematic or the navigator.

wfDefInstSaveVoltages

Controls whether voltage probes are created for instances selected from the schematic or the navigator.

Syntax

```
amsDirect.prep wfDefInstSaveVoltages boolean t | nil
```

Values

`t`	Places a checkmark next to the *Voltages/Signals* label in the *Waveforms* pane of the AMS Options window, indicating that voltage probes are to be created for instances selected from the schematic or navigator. This is the default.
`nil`	Removes the checkmark next to the *Voltages/Signals* field in the *Waveforms* pane of the AMS Options window, indicating that voltages are not to be created for instances selected from the schematic or navigator.

Example

```
amsDirect.prep wfDefInstSaveVoltages boolean nil
```

Removes the checkmark next to the *Voltages/Signals* label so voltage probes are not created for objects selected from the schematic or navigator.

wfDefInstVSaveAll

Specifies whether voltage probes are to be created for all levels of the instances selected from the schematic or the navigator.

Syntax

```
amsDirect.prep wfDefInstVSaveAll t | nil
```

Values

`t`	Specifies that voltage probes are to be created for all levels of the selected instances. This choice is indicated in the AMS Save/Plot list by the word `all` appearing in the *Depth* column for the selected instances
`nil`	Specifies that voltage probes for the selected instances are to be created only for the number of levels specified by the `amsDirect.prep wfDefInstVSaveLvl` variable. This is the default.

Example

```
amsDirect.prep wfDefInstVSaveAll nil
```

Specifies that voltage probes are to be created for all levels of the selected instances.

wfDefInstVSaveLvl

Specifies that voltage probes for the specified number of levels are to be created for instances selected from the schematic or the navigator.

Syntax

```
amsDirect.prep wfDefInstVSaveLvl int level
```

Values

level	The number of levels of voltage probes to be created for the selected instances. The default value is 1.

Example

```
amsDirect.prep wfDefInstVSaveLvl int 2
```

Specifies that voltage probes are to be created for two levels of each of the selected instances. In the AMS Save/Plot list, the *Depth* column for the selected instances contains the value `2`.

wfDefInstVSaveObjects

Specifies the objects for which voltages are to be saved when instances are selected from the schematic or navigator.

Syntax

```
amsDirect.prep wfDefInstVSaveObjects cyclic "Input_ports" |
    "Output_ports" | "All_ports" | "All_data"
```

Values

`Input_ports`	Indicates that only the input ports of the selected instances are to be probed for voltages.
`Output_ports`	Indicates that only the output ports of the selected instances are to be probed for voltages.
`All_ports`	Indicates that both the input and the output ports of the selected instances are to be probed for voltages.
`All_data`	Indicates that all ports and internal signals of the selected instances are to be probed for voltages. This is the default.

Example

```
amsDirect.prep wfDefInstVSaveObjects cyclic "Output_ports"
```

Specifies that only output ports are to be probed for voltages when instances are selected. Consequently, the row that appears in the AMS Save/Plot window when an instance is selected contains the *Output Ports* icon in the *Object* column. The Tcl `probe` command created from the row includes the `-outputs` option.

wfFilter

Controls whether domain filters are applied when probe data are saved.

Filtering by domain is not allowed when you probe currents. Filtering by domain has no effect on probes of nets, signals, or terminals.

Syntax

```
amsDirect.prep wfFilter boolean t | nil
```

Values

`t`	Specifies that the domain filters specified by the `wfFilterSpec` variable are to be applied.
`nil`	Specifies that the domain filters specified by the `wfFilterSpec` variable are not to be applied. This is the default.

Example

```
amsDirect.prep wfFilter boolean t
```

Specifies that the domain filters specified by the `wfFilterSpec` variable are to be applied when probe data are saved. In the *Waveforms* pane of the AMS Options window, the *Filtered by domain* field is check marked.

wfFilterSpec

Specifies the domains for which data are to be saved while probing.

Syntax

```
amsDirect.prep wfFilterSpec cyclic "none" | "digital" | "analog"
```

Values

`none`	Specifies that no filtering is to occur, which means that probe results are saved from both the digital and analog domains. This is the default.
`digital`	Saves probe results from only the digital domain.
`analog`	Saves probe results from only the analog domain.

Example

```
amsDirect.prep wfFilterSpec cyclic "digital"
```

Specifies that only probe results from the digital domain are to be saved. In the Waveforms pane of the AMS Options window, the *Digital* field is selected.