
Virtuoso ADE SKILL Reference

Product Version IC23.1
November 2023

© 2023 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

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

Analog Design Environment XL contains technology licensed from, and copyrighted by: Apache Software Foundation, 1901 Munsey Drive Forest Hill, MD 21050, USA © 2000-2007, Apache Software Foundation.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

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

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1

Introduction to Maestro Functions 45

Creation of SKILL Scripts for Maestro Cellviews 45

Executing Scripts for ADE Explorer and ADE Assembler 45

2

Maestro Cellview Functions 47

Functions to Migrate ADE L or ADE XL States to Maestro Cellviews 48

maeConvertAndCombineMultiADELToAssembler 49

maeMigrateADELStateToMaestro 51

maeMigrateADEXLToMaestro 53

Functions to Create, View, Edit, and Save Setups in maestro Cellviews 55

maeAddOutput 58

maeApplyCornerEditingOptions 60

maeCreateTest 62

maeCloseSession 64

maeConvertViewForReferencedAndLocalRunPlanCorners 67

maeConvertViewForSeparateHistoryManagement 69

maeCloseCornersSetupForm 71

maeDeleteCorner 72

maeDeleteMultiWildcardOutputs 73

maeDeleteOutput 76

maeDeleteExplorerHistory 77

maeDeleteParameter 78

maeDeleteTags 79

maeDeleteTest 80

maeDeleteVar 82

maeEnableCorner 83

maeEnableGlobalVar 85

maeEnableParameter 87

maeEnableTests 89

Virtuoso ADE SKILL Reference

<u>maeExportCornersToCSV</u>	91
<u>maeExportCornersToSDB</u>	92
<u>maeExportGlobalVariablesToCSV</u>	93
<u>maeExportRunPreview</u>	94
<u>maeExportSetupForExplorer</u>	95
<u>maeGetAllExplorerHistoryNames</u>	96
<u>maeGetAnalysis</u>	97
<u>maeGetCornerEditingOptions</u>	99
<u>maeGetCurrentRunMode</u>	101
<u>maeGetEnabledAnalysis</u>	103
<u>maeGetEnvOption</u>	104
<u>maeGetExplorerTestName</u>	107
<u>maeGetSetup</u>	109
<u>maeGetSimOption</u>	111
<u>maeGetSessions</u>	113
<u>maeGetSetupLibrary</u>	114
<u>maeGetTags</u>	115
<u>maeGetTestEnvVar</u>	117
<u>maeGetTestSession</u>	119
<u>maeGetVar</u>	121
<u>maeImportOutputsFromFileInDocuments</u>	124
<u>maeImportSetupForExplorer</u>	126
<u>maelsPrerunScriptEnabled</u>	127
<u>maelsSetupModified</u>	128
<u>maelsSingleTest</u>	129
<u>maelsValidMaestroSession</u>	130
<u>maeLoadCorners</u>	131
<u>maeMakeEditable</u>	133
<u>maeMakeReadOnly</u>	134
<u>maeOpenSetup</u>	135
<u>maeLoadSetupState</u>	137
<u>maeLoadStateForTest</u>	140
<u>maeResetCornerEditingOptions</u>	144
<u>maeRegTimer</u>	145
<u>maeRunDspfChecker</u>	146
<u>maeSaveSetup</u>	147

Virtuoso ADE SKILL Reference

<u>maeSaveSetupStateForOptimization</u>	149
<u>maeSetCornerEditingOptions</u>	151
<u>maeSetStartingPointForOptimization</u>	153
<u>maeSetTags</u>	155
<u>maeStmConsolidateStimuli</u>	157
<u>maeStmGenerateWaveforms</u>	158
<u>maeSaveSetupState</u>	160
<u>maeSetAnalysis</u>	163
<u>maeSetCorner</u>	165
<u>maeSetCurrentRunMode</u>	167
<u>maeSetDesign</u>	168
<u>maeSetDesignForTest</u>	169
<u>maeSetEnableTestVar</u>	171
<u>maeSetEnvOption</u>	173
<u>maeSetHistoryLock</u>	175
<u>maeSetParameter</u>	176
<u>maeSetSpec</u>	178
<u>maeSetSetup</u>	180
<u>maeSetSetupLibrary</u>	182
<u>maeSetSimOption</u>	184
<u>maeSetTestEnvVar</u>	186
<u>maeSetVar</u>	188
<u>maeUpdateImplicitSignals</u>	190
<u>Example script for Viewing and Editing Setup Details</u>	191
<u>Functions to Work with Simulation Results</u>	192
<u>getSimRunInfo</u>	193
<u>maeCloseResults</u>	195
<u>maeDeleteSimulationData</u>	196
<u>maeDisplayLogDiagnostics</u>	198
<u>maeExportOutputView</u>	199
<u>maeForceRdbStatus</u>	202
<u>maeGetNBestDesignPoints</u>	204
<u>maeGetOutputValue</u>	205
<u>maeGetParamConditions</u>	207
<u>maeGetParameter</u>	209
<u>maeGetResultOutputs</u>	211

Virtuoso ADE SKILL Reference

<u>maeGetResultTests</u>	212
<u>maeGetResultsViewSelectedCellsDetails</u>	213
<u>maeGetSpecStatus</u>	215
<u>maeGetTestOutputs</u>	216
<u>maeGetOverallSpecStatus</u>	218
<u>maeGetOverallYield</u>	221
<u>maeImportHistory</u>	222
<u>maeOpenResults</u>	224
<u>maeReadResDB</u>	226
<u>maeRestoreHistory</u>	232
<u>maeRunLogDiagnostics</u>	233
<u>maeWriteDatasheet</u>	236
<u>Example script for Viewing Simulation Results</u>	239
<u>Functions for Checks and Asserts</u>	240
<u>maeCloseViolationDb</u>	241
<u>maeOpenViolationDb</u>	242
<u>maeRefreshViolationDisplay</u>	244
<u>maeWaiveViolation</u>	245
<u>maeUnWaiveViolation</u>	248
<u>Functions to Manage Run Plans</u>	250
<u>maeAddPostRunScript</u>	251
<u>maeAddPreRunScript</u>	252
<u>maeCopySimulationOptions</u>	253
<u>maeCreateNewRun</u>	255
<u>maeCreateRunCondition</u>	257
<u>maeCreateRunCopy</u>	259
<u>maeCopyTestToRun</u>	261
<u>maeDeleteRun</u>	263
<u>maeDeleteRunCondition</u>	265
<u>maeEditSimulationOptions</u>	266
<u>maeEnableRun</u>	267
<u>maeGetEnabledRuns</u>	269
<u>maeGetNumberOfExecutedRuns</u>	270
<u>maeGetHistoryLockFlag</u>	271
<u>maeGetRunPlan</u>	273
<u>maeGetCurrentRunPlanName</u>	275

Virtuoso ADE SKILL Reference

<u>maeGetHistoryNameForCurrentRunInRunPlan</u>	276
<u>maelsFirstRunInRunPlan</u>	277
<u>maelsFinalRunCompleted</u>	278
<u>Functions that Provide Support for Run Plan</u>	279
<u>Functions to Work with Plotting Templates</u>	280
<u>maeGetAllPlottingTemplates</u>	281
<u>maePlotWithPlottingTemplate</u>	282
<u>maeSaveImagesUsingPlottingTemplate</u>	284
<u>Functions for Sensitivity Analysis Setup</u>	286
<u>maeSensDeleteModel</u>	287
<u>maeSensDeleteModelGroup</u>	288
<u>maeSensDeleteParameter</u>	289
<u>maeSensDeleteVar</u>	291
<u>maeSensEnableDesignVariation</u>	292
<u>maeSensEnableStatVariation</u>	293
<u>maeSensGetModel</u>	294
<u>maeSensGetModelGroup</u>	295
<u>maeSensGetModels</u>	297
<u>maeSensGetParameter</u>	299
<u>maeSensGetParameters</u>	301
<u>maeSensGetVar</u>	302
<u>maeSensGetVars</u>	303
<u>maeSensSetMethod</u>	304
<u>maeSensSetModel</u>	305
<u>maeSensSetModelGroup</u>	307
<u>maeSensSetParameter</u>	308
<u>maeSensSetVar</u>	310
<u>Functions for XML File Management</u>	312
<u>sevOpenXmlFile</u>	313
<u>sevConvertStateFormat</u>	314
<u>sevCloseXmlFile</u>	317
<u>sevWriteTable</u>	318
<u>sevReadTable</u>	320
<u>sevWriteValue</u>	321
<u>sevReadValue</u>	323
<u>Functions to Work with the Locally Scoped Models and Options (MTS Options)</u>	324

Virtuoso ADE SKILL Reference

<u>maeGetMTSMode</u>	325
<u>maeSetMTSMode</u>	326
<u>maeGetMTSBlock</u>	328
<u>maeSetMTSBlock</u>	330

3

Fault Simulation Functions 333

<u>getFaultTranData</u>	335
<u>maeAddFaultRule</u>	337
<u>maeAddFaultsToFaultGroup</u>	341
<u>maeAddUndetectedFaultsFromHistoryToGroup</u>	343
<u>maeChangeTypeFromExprToFaultExpr</u>	345
<u>maeClearExistingFaultsForRevalidation</u>	346
<u>maeCopyFaultGroups</u>	347
<u>maeCreateOrRenameFaultGroup</u>	348
<u>maeDeleteFaultGroup</u>	350
<u>maeDeleteFaultRule</u>	351
<u>maeEditFaultRule</u>	352
<u>maeEnableFaults</u>	355
<u>maeGetDUTForFaults</u>	357
<u>maeGetFaultGroups</u>	358
<u>maeGetFaultGroupToRun</u>	359
<u>maeGetFaultRule</u>	360
<u>maeGetFaultRules</u>	362
<u>maeGetFaultRunModeOptions</u>	363
<u>maeGetFaults</u>	365
<u>maeGetFaultSamplingOptions</u>	367
<u>maeGetGlobalFaultOptions</u>	368
<u>maeRunFaultSimulationWithFaultDroppingForActiveTests</u>	369
<u>maeSetDUTForFaults</u>	370
<u>maeSetFaultAnalysisType</u>	371
<u>maeSetFaultDFARunModeOptions</u>	373
<u>maeSetFaultSamplingOptions</u>	375
<u>maeSetFaultTFARunModeOptions</u>	378
<u>maeSetGlobalFaultOptions</u>	381

Virtuoso ADE SKILL Reference

<u>maeGenerateFuSaReport</u>	384
<u>maeGetNumberOfUndetectedFaultsFromHistory</u>	387
<u>maeMergeFaultHistories</u>	388
<u>maePrintFaultDroppingStatistics</u>	390
<u>maeSaveFaultsRunCount</u>	392
<u>maeSwitchActiveFaultGroupForCurrentRun</u>	393
<u>maeSetFaultGroupToRun</u>	394

4

Session Functions 397

<u>axlCloseSession</u>	399
<u>axlCloseSessionInWindow</u>	400
<u>axlCreateSession</u>	402
<u>axlGetMainSetupDB</u>	404
<u>axlGetSessionCellName</u>	405
<u>axlGetSessionLibName</u>	406
<u>axlGetSessionViewName</u>	407
<u>axlGetSessionWindowNumber</u>	408
<u>axlGetToolSession</u>	409
<u>axlGetWindowSession</u>	411
<u>axlGetCurrentResultSimulationHost</u>	412
<u>axlIsSessionReadOnly</u>	413
<u>axlIsValidAXLSession</u>	414
<u>axlMainAppSaveSetup</u>	415
<u>axlNoSession</u>	416
<u>axlRemoveSetupState</u>	417
<u>axlSaveSetupState</u>	418
<u>axlSessionConnect</u>	421
<u>axlSessionDisconnect</u>	423
<u>axlSessionRegisterCreationCallback</u>	425
<u>axlSessionSignalList</u>	427
<u>axlSessionSignalSignature</u>	428
<u>axlSetMainSetupDB</u>	429
<u>axlSetMainSetupDBLCV</u>	430
<u>axlSetupStates</u>	432

<u>axlSuppressPersistedQuestionDialog</u>	433
<u>axlShowPersistedQuestionDialog</u>	434
<u>axlToolSetOpPointInfo</u>	435
<u>Working with Signals or Triggers</u>	436

5

Setup Database Functions 451

<u>axlCloseSetupDB</u>	454
<u>axlCommitSetupDB</u>	455
<u>axlCommitSetupDBAndHistoryAs</u>	456
<u>axlCommitSetupDBas</u>	457
<u>axlDiffSetup</u>	458
<u>axlDeleteNote</u>	460
<u>axlGetCopyRefResultsOption</u>	462
<u>axlGetElementParent</u>	463
<u>axlGetEnabled</u>	465
<u>axlGetLocalResultsDir</u>	466
<u>axlIsLocalResultsDir</u>	467
<u>axlExportSetup</u>	468
<u>axlGetHistoryGroupChildren</u>	470
<u>axlGetActiveSetup</u>	471
<u>axlGetHistoryGroupChildrenEntry</u>	472
<u>axlGetNote</u>	473
<u>axlGetPointNetlistDir</u>	474
<u>axlGetPointPsfDir</u>	476
<u>axlGetPointRunDir</u>	478
<u>axlGetPointTroubleshootDir</u>	480
<u>axlGetResultsLocation</u>	482
<u>axlGetReuseNetlistOption</u>	484
<u>axlGetScript</u>	485
<u>axlGetScriptPath</u>	486
<u>axlGetScripts</u>	487
<u>axlGetSessionFromSetupDB</u>	488
<u>axlGetSetupDBDir</u>	489
<u>axlGetSetupInfo</u>	490

Virtuoso ADE SKILL Reference

<u>axlGetTopLevel</u>	492
<u>axlGetUseIncremental</u>	493
<u>axlImportSetup</u>	494
<u>axlLoadSetupState</u>	495
<u>axlNewSetupDB</u>	498
<u>axlNewSetupDBLCV</u>	499
<u>axlPutNote</u>	500
<u>axlPutScript</u>	502
<u>axlPutTest</u>	503
<u>axlRemoveElement</u>	504
<u>axlResetActive</u>	505
<u>axlSaveSetup</u>	506
<u>axlSaveSetupToLib</u>	507
<u>axlSDBDebugPrint</u>	508
<u>axlSDBGetChild</u>	509
<u>axlSDBGetChildren</u>	510
<u>axlSDBGetChildVal</u>	512
<u>axlSDBGetExtension</u>	513
<u>axlSDBGetName</u>	514
<u>axlSDBGetValue</u>	515
<u>axlSDBHp</u>	516
<u>axlSDBPutExtension</u>	517
<u>axlSDBSetChild</u>	518
<u>axlSDBSetMultipleEntry</u>	521
<u>axlSDBSetValue</u>	523
<u>axlSetAllSweepsEnabled</u>	524
<u>axlSetCopyRefResultsOption</u>	525
<u>axlSetEnabled</u>	526
<u>axlSetReferenceHistoryItemName</u>	528
<u>axlSetReuseNetlistOption</u>	530
<u>axlSetUseIncremental</u>	532
<u>axlSetScriptPath</u>	534
<u>axlWriteDatasheet</u>	535
<u>axlWriteDatasheetForm</u>	538
<u>maeGetReferenceHistories</u>	539

6

Variable Functions	541
axlGetVar	542
axlGetVars	544
axlGetVarValue	546
axlPutVar	548
axlGetAllVarsDisabled	550
axlSetAllVarsDisabled	551
axlSetDefaultVariables	553
axlSetDesignVariablePerTest	555
maeExportDesignVariables	557
maeImportDesignVariables	559

7

Parameter Functions	561
axlGetParameters	562
axlGetParameter	563
axlGetParameterValue	565
axlGetAllParametersDisabled	566
axlRegisterCustomDeviceFilter	567
axlSetParameter	569
axlSetAllParametersDisabled	571

8

Model Functions	573
axlAddModelPermissibleSectionLists	575
axlGetModel	577
axlGetModelBlock	578
axlGetModelFile	580
axlGetModelGroup	581
axlGetModelGroupName	582
axlGetModelGroups	584
axlGetModelPermissibleSectionLists	585
axlGetModelSection	587

Virtuoso ADE SKILL Reference

<u>axlGetModelSections</u>	588
<u>axlGetModelTest</u>	589
<u>axlGetModels</u>	591
<u>axlPutModel</u>	592
<u>axlPutModelGroup</u>	595
<u>axlSetModelBlock</u>	596
<u>axlSetModelFile</u>	598
<u>axlSetModelGroupName</u>	600
<u>axlSetModelPermissibleSectionLists</u>	601
<u>axlSetModelSection</u>	603
<u>axlSetModelTest</u>	604
<u>Functions to Work with Model Files</u>	606

9

<u>Output Functions</u>	609
<u>ALIAS</u>	611
<u>axlAddOutputExpr</u>	613
<u>axlAddOutputs</u>	616
<u>axlAddOutputsColumn</u>	617
<u>axlAddOutputSignal</u>	618
<u>axlDeleteOutput</u>	620
<u>axlDeleteOutputsColumn</u>	622
<u>axlGetCustomOutputResultMenu</u>	623
<u>axlGetOutputNotation</u>	625
<u>axlGetOutputSignificantDigits</u>	626
<u>axlGetOutputSuffix</u>	627
<u>axlGetOutputUnits</u>	628
<u>axlGetOutputUserDefinedData</u>	629
<u>axlGetTemperatureForCurrentPointInRun</u>	630
<u>axlGetUserDefinedOutputsColumns</u>	631
<u>axlOutputResult</u>	632
<u>axlOutputsExportToFile</u>	634
<u>axlOutputsImportFromFile</u>	635
<u>axlPutOutputNotation</u>	637
<u>axlPutOutputSignificantDigits</u>	639

Virtuoso ADE SKILL Reference

<u>axlPutOutputSuffix</u>	640
<u>axlPutOutputUnits</u>	641
<u>axlRenameOutputsColumn</u>	642
<u>axlSetOutputUserDefinedData</u>	643
<u>calcVal</u>	645
<u>Troubleshooting calcVal</u>	666
<u>measureAcross</u>	670

10

Test Functions 673

<u>axlGetCornersForATest</u>	674
<u>axlGetEnabledGlobalVarPerTest</u>	676
<u>axlGetEnabledTests</u>	678
<u>axlGetOrigTestToolArgs</u>	679
<u>axlGetTest</u>	680
<u>axlGetTests</u>	682
<u>axlGetTestToolArgs</u>	683
<u>axlSaveResValue</u>	685
<u>axlSetTestToolArgs</u>	686
<u>axlToolSetOriginalSetupOptions</u>	688
<u>axlToolSetSetupOptions</u>	690
<u>axlCustomADETestName</u>	694
<u>axlWriteOceanScriptLCV</u>	696

11

Specification Functions 699

<u>axlAddSpecToOutput</u>	700
<u>axlGetSpecs</u>	703
<u>axlGetSpec</u>	704
<u>axlGetSpecData</u>	706
<u>axlGetSpecWeight</u>	708

12

Corners Functions	709
<u>axlGetAllCornersEnabled</u>	711
<u>axlCorners</u>	712
<u>axlGetCorner</u>	714
<u>axlGetCorners</u>	716
<u>axlGetCornerDisabledTests</u>	718
<u>axlGetCornerCountForName</u>	719
<u>axlGetCornerNameForCurrentPointInRun</u>	720
<u>axlGetNominalCornerEnabled</u>	721
<u>axlGetNominalCornerTestEnabled</u>	722
<u>axlGetStatVars</u>	724
<u>axlLoadCorners</u>	726
<u>axlLoadCornersFromPcfToSetupDB</u>	728
<u>axlPlotAcrossDesignPoints</u>	730
<u>axlPutCorner</u>	731
<u>axlPutDisabledCorner</u>	732
<u>axlSetDefaultCornerEnabled</u>	734
<u>axlSetAllCornersEnabled</u>	736
<u>axlSetCornerName</u>	737
<u>axlSetCornerTestEnabled</u>	738
<u>axlSetNominalCornerEnabled</u>	740
<u>axlSetNominalCornerTestEnabled</u>	741
<u>axlSetWCCTime</u>	743
<u>axlGetWCCCorner</u>	744
<u>axlGetWCCHistory</u>	745
<u>axlGetWCCRResult</u>	746
<u>axlGetWCCSpec</u>	747
<u>axlGetWCCSpecs</u>	748
<u>axlGetWCCTest</u>	749
<u>axlGetWCCTime</u>	751
<u>axlGetWCCRRangeBound</u>	753
<u>axlGetWCCVar</u>	754
<u>axlGetWCCVarMonotonicity</u>	755
<u>axlGetWCCVars</u>	756

13

Optimization Functions	759
axlGetWYCSigmaTargetLimit	760
axlSetWYCSigmaTargetLimit	761

14

Run-Related Functions	762
maeGetSimulationMessages	765
maeGetMappingForJobAndPoint	767
maeOpenLogViewer	768
maeResumeSimulation	769
maeRunSimulation	770
maeStopSimulation	775
maeSuspendSimulation	776
maeSetPreRunScript	777
maeSetRunOption	779
maeWaitUntilDone	782
maeWriteScript	784
axlExportOutputView	786
axlGetAllSweepsEnabled	788
axlGetCurrentRunMode	789
axlGetParasiticRunMode	791
axlGetParasiticParaLCV	792
axlGetParasiticSchLCV	793
axlGetPreRunScript	794
axlGetRunDistributeOptions	795
axlGetRunData	797
axlGetRunMode	799
axlGetRunModes	800
axlGetRunOption	801
axlGetRunOptionName	804
axlGetRunOptions	805
axlGetRunOptionValue	806
axlGetRunStatus	807

Virtuoso ADE SKILL Reference

<u>axlIsSimUsingStatParams</u>	809
<u>axlPutRunOption</u>	810
<u>axlRunAllTests</u>	813
<u>axlRunAllTestsWithCallback</u>	815
<u>axlRunSimulation</u>	817
<u>axlSetCurrentRunMode</u>	818
<u>axlImportPreRunScript</u>	821
<u>axlSetParasiticRunMode</u>	823
<u>axlSetPreRunScript</u>	825
<u>axlSetPreRunScriptEnabled</u>	826
<u>axlSetRunDistributeOptions</u>	828
<u>axlSetRunOptionName</u>	830
<u>axlStop</u>	831
<u>axlStopAll</u>	832
<u>axlReadHistoryResDB</u>	833
<u>axlReadResDB</u>	834
<u>axlSetRunOptionValue</u>	839

15

<u>History Functions</u>	842
<u>maeGetHistoryCloudUUID</u>	844
<u>axlGetCurrentHistory</u>	845
<u>axlGetDataViewHistoryUserMenu</u>	846
<u>axlGetHistory</u>	848
<u>axlGetHistoryCheckpoint</u>	849
<u>axlGetHistoryEntry</u>	851
<u>axlGetHistoryGroup</u>	852
<u>axlGetHistoryLock</u>	853
<u>axlGetHistoryName</u>	855
<u>axlGetHistoryOutputs</u>	856
<u>axlGetHistoryPrefix</u>	857
<u>axlGetHistoryResults</u>	858
<u>axlGetOverwriteHistory</u>	859
<u>axlGetOverwriteHistoryName</u>	860
<u>axlLoadHistory</u>	861

Virtuoso ADE SKILL Reference

<u>axlSetHistoryLock</u>	862
<u>axlSetHistoryName</u>	864
<u>axlSetHistoryPrefixInPreRunTrigger</u>	865
<u>axlSetOverwriteHistory</u>	866
<u>axlSetOverwriteHistoryName</u>	867
<u>axlOpenResDB</u>	868
<u>axlPutHistoryEntry</u>	869
<u>axlReEvaluateHistory</u>	870
<u>axlRemoveSimulationResults</u>	872
<u>axlRestoreHistory</u>	874
<u>axlViewHistoryResults</u>	875
<u>axlWriteMonteCarloResultsCSV</u>	876

16

Job Policy Functions 878

<u>axlAddJobPolicy</u>	881
<u>axlAttachJobPolicy</u>	884
<u>axlDeleteJobPolicy</u>	886
<u>axlDetachJobPolicy</u>	887
<u>axlJobIntfcDebugPrintf</u>	888
<u>axlJobIntfcDebugToFile</u>	889
<u>axlJobIntfcDebugp</u>	890
<u>axlJobIntfcExitMethod</u>	891
<u>axlJobIntfcHealthMethod</u>	892
<u>axlJobIntfcSetDebug</u>	894
<u>axlJobIntfcStartJobArrayMethod</u>	895
<u>axlJobIntfcStartMethod</u>	897
<u>axlJPGUICustDiffer</u>	898
<u>axlJPGUICustHiddenSections</u>	899
<u>axlJPGUICustHIFields</u>	900
<u>axlJPGUICustOffset</u>	901
<u>axlJPGUICustReadFromForm</u>	902
<u>axlJPGUICustSelected</u>	903
<u>axlRegisterJobIntfc</u>	904
<u>axlJPGUICustWriteToForm</u>	906

Virtuoso ADE SKILL Reference

<u>axlRegisteredJobIntfcNames</u>	908
<u>axlRegisterJPGUICust</u>	909
<u>axlGetAttachedJobPolicy</u>	910
<u>axlGetJobPolicy</u>	912
<u>axlGetJobPolicyTypes</u>	913
<u>axlIsCRPPProcess</u>	914
<u>axlSaveJobPolicy</u>	915
<u>axlSetJobPolicyProperty</u>	916
<u>axlStopAllJobs</u>	918
<u>axlStopJob</u>	920
<u>maeClearAllTestJobPolicies</u>	921
<u>maeClearTestJobPolicy</u>	922
<u>maeCreateNetlistForCorner</u>	923
<u>maeGetAllJobPolicies</u>	924
<u>maeGetJobControlMode</u>	925
<u>maeGetJobPolicy</u>	926
<u>maeGetJobPolicyByName</u>	929
<u>maeHasTestJobPolicy</u>	930
<u>maelsEvaluatorProcess</u>	931
<u>maelsNetlistProcess</u>	932
<u>maeSetJobControlMode</u>	933
<u>maeSetJobPolicy</u>	934
<u>maeStopAllJobs</u>	936
<u>maeStopJob</u>	938

17

<u>Core Netlister Functions</u>	940
<u>Initialization Functions</u>	941
<u>Customizing an Inherited Method for Your Simulator</u>	948
<u>Workflow for Adding Features to Simulators</u>	949
<u>Changing Banner Menus</u>	952
<u>Changing Banner Menus for a Particular Simulator</u>	953
<u>asiInit<yourSimulator></u>	954
<u>asiRegisterTool</u>	955
<u>asiInitDataAccessFunction</u>	957

Virtuoso ADE SKILL Reference

<u>asiInitEnvOption</u>	958
<u>asiInitAnalysis</u>	959
<u>asiInitAdvAnalysis</u>	960
<u>asiInitSimOption</u>	961
<u>asiGetPageCallBack</u>	962
<u>asiSetPageCallBack</u>	963
<u>Netlisting Invocation Functions for Direct Integration</u>	964
<u>asiGetNetlistFormatterClass</u>	965
<u>asiSetNetlistFormatterClass</u>	966
<u>asiCreateFormatter</u>	967
<u>asiCreateCdsenvFile</u>	968
<u>asiGetFormatter</u>	969
<u>asiGetSimInputFileName</u>	970
<u>asiGetSimInputFileSuffix</u>	971
<u>Netlist Functions</u>	972
<u>The nlAnalogFormatter Class</u>	973
<u>nlGetNetlister</u>	976
<u>nlGetPCellParamSource</u>	977
<u>netlistDir</u>	978
<u>nlGetScratchInstance</u>	979
<u>nlGetScratchInstPcellParamValue</u>	980
<u>nlGetSwitchMaster</u>	981
<u>nlGetToolName</u>	982
<u>nlInitialize</u>	983
<u>nlPrintHeader</u>	986
<u>nlIncludePspiceFile</u>	987
<u>nlIncludeVerilogaFile</u>	988
<u>nlIncludeVerilogFile</u>	989
<u>nlIncludeDbDSPFTextFile</u>	990
<u>nlIncludeDbSPICEMODELTextFile</u>	991
<u>nlIsPcellInstance</u>	992
<u>nlIsPcellParam</u>	994
<u>nlIsSmartExtractedView</u>	996
<u>nlPrintFooter</u>	997
<u>nlPrintSubcktHeaderComments</u>	998
<u>nlPrintTopCellHeaderComments</u>	999

Virtuoso ADE SKILL Reference

<u>nlPrintTopCellFooterComments</u>	1000
<u>nlPrintTopCellHeader</u>	1001
<u>nlPrintTopCellFooter</u>	1002
<u>nlPrintSubcktHeader</u>	1003
<u>nlPrintSubcktFooter</u>	1004
<u>nlPrintSubcktFooterComments</u>	1005
<u>nlPrintInstComments</u>	1006
<u>nlPrintInst</u>	1007
<u>nlPrintInstEnd</u>	1009
<u>nlPrintSubcktBegin</u>	1010
<u>nlPrintSubcktName</u>	1011
<u>nlPrintSubcktEnd</u>	1012
<u>nlPrintHeaderComments</u>	1013
<u>nlPrintSubcktParameters</u>	1014
<u>nlPrintSubcktTerminalList</u>	1015
<u>nlPrintInstName</u>	1016
<u>nlPrintInstSignals</u>	1017
<u>nlPrintModelName</u>	1018
<u>nlPrintInstParameters</u>	1019
<u>The Netlist Object</u>	1020
<u>nlError</u>	1028
<u>nlObjError</u>	1029
<u>nlGetDesign</u>	1030
<u>nlGetGlobalNets</u>	1031
<u>nlGetNetlistDir</u>	1032
<u>nlDisplayOption</u>	1033
<u>nlGetCurrentSwitchMaster</u>	1034
<u>nlGetNetlistedStopCellViewList</u>	1035
<u>nlGetOption</u>	1036
<u>nlGetOptionNameList</u>	1037
<u>nlMapGlobalNet</u>	1038
<u>nlInfo</u>	1039
<u>nlSetOption</u>	1040
<u>nlWarning</u>	1041
<u>nlPrintComment</u>	1042
<u>nlPrintIndentString</u>	1043

Virtuoso ADE SKILL Reference

<u>nlPrintString</u>	1044
<u>nlPrintStringNoFold</u>	1045
<u>Methods for Instances</u>	1046
<u>nlIsModelNameInherited</u>	1047
<u>nlGetFormatter</u>	1048
<u>nlGetSimName</u>	1049
<u>nlGetSignalList</u>	1051
<u>nlGetTerminalList</u>	1052
<u>nlGetTerminalSignalName</u>	1053
<u>nlGetNumberOfBits</u>	1054
<u>nlGetModelName</u>	1055
<u>nlGetParamList</u>	1057
<u>nlGetParamStringValue</u>	1058
<u>nlGetId</u>	1061
<u>nlIncludeSrcFile</u>	1062
<u>nlPrintComments</u>	1063
<u>hnlGetInstanceCount</u>	1064
<u>Cellviews</u>	1065
<u>nlGetCellName</u>	1066
<u>nlGetLibName</u>	1067
<u>nlGetSimTerminalNets</u>	1068
<u>nlGetTerminalNets</u>	1069
<u>nlGetSwitchViewList</u>	1070
<u>nlGetViewName</u>	1071
<u>Designs</u>	1072
<u>nlGetTopLibName</u>	1073
<u>nlGetTopCellName</u>	1074
<u>nlGetTopViewName</u>	1075
<u>nlTranslateFlatIEPathName</u>	1076
<u>Other Customization Procedures</u>	1077
<u>nlSetPcellName</u>	1078
<u>ansCdICompPrim</u>	1079
<u>ansCdIHnlPrintInst</u>	1080
<u>ansCdIPrintString</u>	1081
<u>ansCdIPrintInheritedParams</u>	1082
<u>ansCdIPrintInstParams</u>	1083

Virtuoso ADE SKILL Reference

<u>ansCdIPrintInstProps</u>	1084
<u>ansCdIPrintInstName</u>	1085
<u>ansCdIPrintModelName</u>	1086
<u>ansCdIPrintModuleName</u>	1088
<u>ansCdIPrintConnections</u>	1089
<u>ansCdIGetSegmentConnections</u>	1090
<u>ansCdIPrintSwitchPCellInst</u>	1093
<u>ansCdIPrintSwitchPCellInstParam</u>	1094
<u>ansCdIPrintSwitchPCellSubcktConn</u>	1095
<u>ansCdIPrintSwitchPCellSubCircuit</u>	1097
<u>ansCdIGetSegmentInfo</u>	1098
<u>ansCdIGetSegmentInstParams</u>	1100
<u>ansCdIGetSimPropValue</u>	1102
<u>ansCdIGetMultiplicity</u>	1103
<u>auCdI</u>	1105
<u>Other Backend Netlister Functions</u>	1106
<u>acdIArtPrintIncludedNetlist</u>	1107
<u>acdICurrentInstName</u>	1109
<u>auLvs</u>	1110
<u>auProbeAddDevsForNet</u>	1111
<u>LVS</u>	1112
<u>HSPICE Functions</u>	1113
<u>hnlHspicePrintInstPropVal</u>	1114
<u>hnlHspiceInstPropVal</u>	1115
<u>hnlHspicePrintInstPropEqVal</u>	1116
<u>hnlHspicePrintMOSfetModel</u>	1117
<u>hnlHspicePrintNMOSfetElement</u>	1118
<u>Name Mapping Variables</u>	1119
<u>Netlisting Option Functions for Socket Interfaces</u>	1122
<u>asiDisplayNetlistOption</u>	1123
<u>asiGetNetlistOption</u>	1124
<u>asiInit<your Simulator>NetlistOption</u>	1125
<u>asiSetNetlistOption</u>	1126
<u>Mixed-Signal Simulation Functions</u>	1127
<u>asiVerilogNetlistMoreCB</u>	1128
<u>asiGetDigitalNetlistFileName</u>	1129

Virtuoso ADE SKILL Reference

<u>asiConstructDigitalNetlist</u>	1130
<u>asiInitializeNetlisterMixed</u>	1131
<u>asiNetlistMixed</u>	1132
<u>asiGetVerilogCommandLineOption</u>	1133
<u>asiGetDigitalCommandLineOption</u>	1134
<u>asiPrepareDigitalSimulation</u>	1135
<u>asiCheckDigitalSimulationSuccess</u>	1136
<u>Simulation Functions for Direct and Socket Interfaces</u>	1137
<u>asiGetNetworkId</u>	1139
<u>asiGetDigitalStimulusFileName</u>	1140
<u>asiEditDigitalStimulus</u>	1141
<u>asiPartitionDesign</u>	1142
<u>asiGetDigitalSimulatorLogFileName</u>	1143
<u>asiGetDigitalSimExecName</u>	1144
<u>asiSetVerilogHost</u>	1145
<u>asiSetVerilogHostMode</u>	1146
<u>asiGetVerilogHost</u>	1147
<u>asiGetVerilogHostMode</u>	1148
<u>asiGetAnalogRunDir</u>	1149
<u>asiGetDigitalRunDir</u>	1150
<u>asiGetAnalogKeepList</u>	1151
<u>asiGetDigitalKeepList</u>	1152
<u>asiInitMixedKeepOption</u>	1153
<u>asiInitVerilog</u>	1154
<u>asiInitVerilogEnvOption</u>	1155
<u>asiInitVerilogFNLEnvOption</u>	1156
<u>asiInitVerilogHNLEnvOption</u>	1157
<u>asiInitVerilogSimOption</u>	1158
<u>asiSetAnalogSimulator</u>	1159
<u>asiSetDigitalSimulator</u>	1160
<u>mvpDisplaySetPartSetupForm</u>	1161
<u>mvpEditIPEProps</u>	1162
<u>Functions for Formatting Hierarchical Interface Elements</u>	1163
<u>hnlVerilogPrintNmosPmos</u>	1164
<u>hnlVerilogPrintCmos</u>	1165
<u>nlGetCdf</u>	1166

18

Functions to Customize Simulator Integration 1168

Environment Variable Functions 1185

<u>asiAddEnvOption</u>	1186
<u>asiChangeEnvOption</u>	1193
<u>asiChangeEnvOptionFormProperties</u>	1199
<u>asiDeleteEnvOption</u>	1202
<u>asiDisplayEnvOption</u>	1203
<u>asiDisplayEnvOptionFormProperties</u>	1204
<u>asiGetEnvOptionChoices</u>	1205
<u>asiGetEnvOptionVal</u>	1206
<u>asiInit<yourSimulator>EnvOption</u>	1207
<u>asiSetEnvOptionChoices</u>	1208
<u>asiSetEnvOptionVal</u>	1209

OASIS Functions 1211

<u>asiExtractQRCTechPathFromDSPF</u>	1213
<u>asiExtractQRCTechPathFromSession</u>	1214
<u>asiGetAnalogSimulator</u>	1215
<u>asiGetAdvAnalysis</u>	1216
<u>asiGetDesignVarListWithOriginalExpressions</u>	1217
<u>asiGetEMIROptionVal</u>	1218
<u>asiGetNetlistFileListToSymLink</u>	1219
<u>asiGetDigitalSimulator</u>	1220
<u>asiAnalogAutoloadProc</u>	1221
<u>ansAnalogRegCDFsimInfo</u>	1222
<u>asiCheckAcEnabledWhenNoiseEnabled</u>	1223
<u>asiCheckAnalysis</u>	1224
<u>asiCheckBlank</u>	1225
<u>asiCreateIncludeStatementFile</u>	1226
<u>asiGetAnalysisField</u>	1227
<u>asiGetHighPerformanceOptionVal</u>	1228
<u>asiSetHighPerformanceOptionVal</u>	1229
<u>asiDisplayHighPerformanceOption</u>	1230
<u>asiGetDesignCellName</u>	1232
<u>asiGetDesignLibName</u>	1233

Virtuoso ADE SKILL Reference

<u>asiGetDesignViewName</u>	1234
<u>asiGetDrIData</u>	1235
<u>asiGetId</u>	1238
<u>asiGetIterationUpdateFile</u>	1239
<u>asiGetResultsPsfDir</u>	1241
<u>asiGetResultsNetlistDir</u>	1242
<u>asiGetSimulatorList</u>	1243
<u>asiGetSimCommandLineOrder</u>	1244
<u>asiGetStimulusGlobals</u>	1245
<u>asiGetStimulusInputs</u>	1246
<u>asilsConfigDesign</u>	1247
<u>asiSetValid</u>	1248
<u>asiCheckBlankNumericLeq</u>	1249
<u>asiCheckBlankNumericGeq</u>	1250
<u>asiFormatGraphicalStimuli</u>	1251
<u>asiFormatGraphicalStimulusFileList</u>	1252
<u>asiAddOceanAlias</u>	1253
<u>asiGetAvailableMCOptions</u>	1254
<u>asiGetSupportedMCOptions</u>	1256
<u>asiSetEMIROptionVal</u>	1257
<u>OASIS Print Functions</u>	1260
<u>artOutfile</u>	1261
<u>artFprintf</u>	1263
<u>artClose</u>	1264
<u>artCloseAllFiles</u>	1265
<u>artFlush</u>	1266
<u>artListOpenFiles</u>	1267
<u>Simulator Option Functions</u>	1268
<u>asiAddSimOption</u>	1269
<u>asiChangeSimOption</u>	1277
<u>asiChangeSimOptionFormProperties</u>	1283
<u>asiDeleteSimOption</u>	1285
<u>asiDisplaySimOption</u>	1286
<u>asiDisplaySimOptionFormProperties</u>	1287
<u>asiGetReservedWordList</u>	1288
<u>asilsCaseSensitive</u>	1289

Virtuoso ADE SKILL Reference

<u>asiGetSimOptionChoices</u>	1290
<u>asiGetSimOptionNameList</u>	1291
<u>asiGetSimOptionSendMethod</u>	1292
<u>asiGetSimOptionVal</u>	1293
<u>asiGetSimulationRunCommand</u>	1294
<u>asiInit<yourSimulator>SimOption</u>	1295
<u>asiIsCLPSupported</u>	1296
<u>asiSetHostOptions</u>	1297
<u>asiSetSimOptionChoices</u>	1299
<u>asiSetSimOptionVal</u>	1300
<u>asiGetSimulatorSrcList</u>	1301
<u>Analysis Function</u>	1302
<u>asiAddAnalysis</u>	1305
<u>asiAddAnalysisField</u>	1308
<u>asiAddAnalysisOption</u>	1313
<u>asiChangeAnalysis</u>	1319
<u>asiChangeAnalysisField</u>	1322
<u>asiChangeAnalysisOption</u>	1327
<u>asiChangeAnalysisOptionFormProperties</u>	1333
<u>asiCreateAnalysisField</u>	1335
<u>asiCreateAnalysisOption</u>	1341
<u>asiDeleteAnalysis</u>	1347
<u>asiDeleteAnalysisField</u>	1348
<u>asiDeleteAnalysisOption</u>	1349
<u>asiDisableAnalysis</u>	1350
<u>asiDisplayAnalysis</u>	1351
<u>asiDisplayAnalysisField</u>	1352
<u>asiDisplayAnalysisOption</u>	1353
<u>asiDisplayAnalysisOptionFormProperties</u>	1354
<u>asiEnableAnalysis</u>	1355
<u>asiFormatAnalysis</u>	1356
<u>asiGetAnalysis</u>	1358
<u>asiGetAnalysisFieldChoices</u>	1359
<u>asiGetAnalysisFieldList</u>	1360
<u>asiGetAnalysisFieldVal</u>	1361
<u>asiGetAnalysisFormFieldChoices</u>	1362

Virtuoso ADE SKILL Reference

<u>asiGetAnalysisFormObj</u>	1363
<u>asiGetAnalysisFormFieldVal</u>	1364
<u>asiGetAnalysisName</u>	1366
<u>asiGetAnalysisNameList</u>	1367
<u>asiGetAnalysisOptionChoices</u>	1368
<u>asiGetAnalysisOptionList</u>	1369
<u>asiGetAnalysisOptionSendMethod</u>	1371
<u>asiGetAnalysisOptionVal</u>	1372
<u>asiGetAnalysisParamNameList</u>	1373
<u>asiGetEnabledAnalysisList</u>	1374
<u>asiInit<yourSimulator>Analysis</u>	1375
<u>asIsAnalysisEnabled</u>	1376
<u>asiSetAnalysisFieldChoices</u>	1377
<u>asiSetAnalysisFieldVal</u>	1378
<u>asiSetAnalysisFormFieldChoices</u>	1379
<u>asiSetAnalysisFormFieldVal</u>	1380
<u>asiSetAnalysisFormWidth</u>	1381
<u>asiSetAnalysisOptionFormProperties</u>	1382
<u>asiSetAnalysisOptionChoices</u>	1384
<u>asiSetAnalysisOptionVal</u>	1385
<u>Simulation Control Functions for Direct Interfaces</u>	1386
<u>asiInitialize</u>	1387
<u>asiNetlist</u>	1389
<u>asiInterruptSim</u>	1390
<u>asiSetProjectDirChangeSetup</u>	1391
<u>asiQuitSimulator</u>	1392
<u>asiRunSimulation</u>	1394
<u>asiGetPredefinedCommandLineOption</u>	1395
<u>asiGetCommandFooter</u>	1396
<u>Integrator Overloadable Methods for Formatting Control Statements</u>	1397
<u>asiFormatControlStmts</u>	1398
<u>asiFormatDesignVarList</u>	1399
<u>asiFormatInitCond</u>	1401
<u>asiFormatNodeSet</u>	1403
<u>asiFormatKeepList</u>	1405
<u>asiFormatSimulatorOptions</u>	1407

Virtuoso ADE SKILL Reference

<u>asiFormatAnalysisList</u>	1409
<u>atAnalysis</u>	1410
<u>asiFormatModelLibSelectionList</u>	1412
<u>asiFormatDefinitionFileList</u>	1413
<u>asiFormatTextStimulusFileList</u>	1414
<u>asiNeedSuffixEvaluation</u>	1415
<u>asiInvalidateControlStmts</u>	1416
<u>Utility Functions</u>	1417
<u>asiGetSimExecName</u>	1418
<u>asiGetCommandLineOption</u>	1419
<u>asiGetAnalysisSigList</u>	1420
<u>asiGetAnalysisType</u>	1421
<u>asiGetAnalysisSimFieldList</u>	1422
<u>asiGetModelLibSelectionList</u>	1423
<u>asiGetModelLibFile</u>	1424
<u>asiGetModelLibSection</u>	1425
<u>asiGetDefinitionFileList</u>	1426
<u>asiGetTextStimulusFileList</u>	1427
<u>asiGetFormattedVal</u>	1428
<u>asiGetSelObjName</u>	1430
<u>asiGetSelObjType</u>	1431
<u>asiGetSelObjValue</u>	1432
<u>asiMapOutputName</u>	1433
<u>asiGetSimInputFileList</u>	1435
<u>artInvalidateAmap</u>	1436
<u>Flowchart Functions</u>	1437
<u>asiAddFlowchartLink</u>	1438
<u>asiAddFlowchartStep</u>	1439
<u>asiChangeFlowchartStep</u>	1441
<u>asiCreateFlowchart</u>	1443
<u>asiDeleteFlowchartLink</u>	1444
<u>asiDeleteFlowchartStep</u>	1445
<u>asiDisplayFlowchart</u>	1447
<u>asiExecuteFlowchart</u>	1448
<u>asiFinalNetlist</u>	1449
<u>asiGetFlowchart</u>	1450

Virtuoso ADE SKILL Reference

<u>asiInit<yourSimulator>Flowchart</u>	1451
<u>asiInvalidateFlowchartStep</u>	1453
<u>asiRawNetlist</u>	1454
<u>asiSendAnalysis</u>	1455
<u>asiSendControlStmts</u>	1456
<u>asiSendDesignVars</u>	1458
<u>asiSendInitCond</u>	1459
<u>asiSendInitFile</u>	1460
<u>asiSendKeepList</u>	1461
<u>asiSendModelPath</u>	1462
<u>asiSendNetlist</u>	1463
<u>asiSendNodeSets</u>	1464
<u>asiSendOptions</u>	1465
<u>asiSendRestore</u>	1467
<u>asiSendUpdateFile</u>	1468
<u>Keep Option Functions</u>	1469
<u>asiAddKeepOption</u>	1470
<u>asiChangeKeepOption</u>	1475
<u>asiChangeKeepOptionFormProperties</u>	1480
<u>asiDeleteKeepOption</u>	1482
<u>asiDisplayKeepOption</u>	1483
<u>asiDisplayKeepOptionFormProperties</u>	1484
<u>asiGetKeepOptionChoices</u>	1485
<u>asiGetKeepOptionVal</u>	1486
<u>asiInit<yourSimulator>KeepOption</u>	1487
<u>asiSetKeepOptionChoices</u>	1488
<u>asiSetKeepOptionVal</u>	1489
<u>Reliability Functions</u>	1490
<u>calcValForRel</u>	1491
<u>maeGetStressFile</u>	1494
<u>maeAddRelxSetup</u>	1498
<u>maeCopyRelxSetup</u>	1502
<u>maeDeleteRelxSetup</u>	1504
<u>maeCreateNetlistForRel</u>	1506
<u>maeExportScenarioSetup</u>	1508
<u>maeGetReliabilityAnalysisEnabled</u>	1511

Virtuoso ADE SKILL Reference

<u>maeGetReliabilityModel</u>	1513
<u>maeGetReliabilityOptionVal</u>	1515
<u>maeGetReliabilityVarValue</u>	1517
<u>maeGetRelxSetups</u>	1519
<u>maeGetRelxSetupEnabled</u>	1521
<u>maeGetRelxStressFile</u>	1523
<u>maeGetRelxTest</u>	1524
<u>maeGetRelxTestEnabled</u>	1526
<u>maeImportScenarioSetup</u>	1528
<u>maeRenameRelxSetup</u>	1532
<u>maeSetReliabilityAnalysisEnabled</u>	1534
<u>maeSetReliabilityModel</u>	1536
<u>maeSetReliabilityOptionVal</u>	1538
<u>maeSetReliabilityVarValue</u>	1540
<u>maeSetRelxSetupEnabled</u>	1542
<u>maeSetRelxStressFile</u>	1544
<u>maeSetRelxTest</u>	1546
<u>maeSetRelxTestEnabled</u>	1548
<u>relxEnableFormTab</u>	1550
<u>relxDisplayDiscField</u>	1552
<u>relxEnableDiscField</u>	1555
<u>relxHideAgeCalculationApproachField</u>	1558
<u>relxGetCustomTabName</u>	1559
<u>relxCustomizeDisplayOrEnableStatus</u>	1560
<u>relxCreateCustomizedTab</u>	1562
<u>relxAddSetupRelxOption</u>	1565
<u>asiFormatSpecialParameterForRel</u>	1570
<u>relxInitOptionsInCdsenv</u>	1571
<u>relxAddReliabilityInStateComponent</u>	1572
<u>relxAddReliabilityOption</u>	1573
<u>relxCreateRunObjectFile</u>	1576
<u>relxDisplayReliabilityForm</u>	1577
<u>relxDisplayResult</u>	1578
<u>relxFormatRXControlFile</u>	1580
<u>relxGetModifyNetlistVal</u>	1581
<u>relxGetMosAgingTimeUnitVal</u>	1582

Virtuoso ADE SKILL Reference

<u>relxGetMosAgingTimeVal</u>	1583
<u>relxGetReliabilityOptionChoices</u>	1584
<u>relxGetReliabilityOptionVal</u>	1586
<u>relxGetRelxStage</u>	1587
<u>relxGetStressFileDir</u>	1588
<u>relxGetRXControlFileName</u>	1589
<u>relxGetSimulationRunCommand</u>	1590
<u>relxGetSpecifiedReliabilityStateFileName</u>	1591
<u>relxGetUserCmdLine</u>	1592
<u>relxHighLightDevices</u>	1593
<u>relxInitAdapterReliabilityOption</u>	1595
<u>relxInitReliabilityOption</u>	1596
<u>relxIsAgingOn</u>	1598
<u>relxIsReliabilityEnabled</u>	1599
<u>relxIsStressOn</u>	1600
<u>relxPostSimulation</u>	1601
<u>relxRunSimulation</u>	1602
<u>relxSetAgingVal</u>	1603
<u>relxSetReliabilityOptionFormProperties</u>	1604
<u>relxSetReliabilityOptionVal</u>	1606
<u>relxSetReliabilityVal</u>	1608
<u>Stimulus Generation Functions</u>	1609
<u>asiStmSupportWaveformGeneration</u>	1610
<u>asiStmGenerateNetlist</u>	1611
<u>asiStmRunSimulation</u>	1612
<u>asiStmGenerateWaveform</u>	1613
<u>Miscellaneous Functions</u>	1615
<u>ahdlUpdateViewInfo</u>	1616
<u>amseGeneralSetupForm</u>	1618
<u>amseQuickSetupForm</u>	1619
<u>amsOSSUnlExecutePostNetlistTrigger</u>	1620
<u>amsUpdateTextviews</u>	1621
<u>vmsUpdateCellViews</u>	1623
<u>annRetrieveFromEffectiveCDF</u>	1625
<u>artEnableAnnotationBalloon</u>	1626
<u>artGenerateHierSymbolCDF</u>	1628

Virtuoso ADE SKILL Reference

<u>artGetCdfTargetCV</u>	1629
<u>artGetCellViewDesignVarList</u>	1630
<u>artCurrentInstSimName</u>	1631
<u>artListToWaveform</u>	1632
<u>artBlankString</u>	1633
<u>artMakeString</u>	1634
<u>artMakeStringPrec15</u>	1636
<u>asiAddDesignVarList</u>	1637
<u>asiAddVerilogArgs</u>	1638
<u>asiGetDataReaderVersion</u>	1639
<u>asiLoadState</u>	1640
<u>asiSaveState</u>	1642
<u>asiCheck</u>	1644
<u>asiCheckDesignVariable</u>	1646
<u>asiCheckExpression</u>	1647
<u>asiCheckExpressionGreater</u>	1648
<u>asiCheckBlankNumeric</u>	1649
<u>asiCheckBlankNumericGreater</u>	1650
<u>asiCheckBlankNumericNequal</u>	1651
<u>asiCheckBlankNetExists</u>	1652
<u>asiCheckBlankInstExists</u>	1653
<u>asiCheckMultipleGreater</u>	1654
<u>asiCheckSimulationSuccess</u>	1655
<u>asiCreateLogFileVerilog</u>	1656
<u>asiDcStore</u>	1657
<u>asiGetCurrentSession</u>	1658
<u>asiGetDesignVarList</u>	1659
<u>asiGetFormFieldChoices</u>	1660
<u>asiGetFormFieldVal</u>	1661
<u>asiGetKeepList</u>	1662
<u>asiGetLogFileList</u>	1663
<u>asiGetMarchList</u>	1665
<u>asiGetNetlistDir</u>	1666
<u>asiGetOutputList</u>	1667
<u>asiGetPlotList</u>	1668
<u>asiGetPsfDir</u>	1669

Virtuoso ADE SKILL Reference

<u>asiGetSession</u>	1670
<u>asiGetSimName</u>	1671
<u>asiGetTool</u>	1672
<u>asiGetTopCellView</u>	1673
<u>asiSendSim</u>	1674
<u>asiSetDesignVarList</u>	1676
<u>asiSetFormFieldChoices</u>	1678
<u>asiSetFormFieldVal</u>	1679
<u>asiSetKeepList</u>	1681
<u>asiSetMarchList</u>	1682
<u>asiSetPlotList</u>	1683
<u>asiSetSyncFlag</u>	1684
<u>asiTransientStore</u>	1685
<u>asiMapNetName</u>	1686
<u>asiMapTerminalName</u>	1687
<u>asiMapInstanceName</u>	1688
<u>asiRegCallBackOnSimComp</u>	1689
<u>asiUnRegCallBackOnSimComp</u>	1690
<u>asiRegCallBackOnSimCompForDist</u>	1691
<u>asiUnRegCallBackOnSimCompForDist</u>	1692
<u>almDefineParam_accuracyMode</u>	1693
<u>almDefineParam_additionalParam</u>	1694
<u>almDefineParam_fq</u>	1695
<u>almDefineParam_noiseParaLabel</u>	1696
<u>almDefineParam_nportFileB</u>	1697
<u>almDefineParam_otherParaLabel</u>	1698
<u>almDefineParam_tranAdvanParaLabel</u>	1699
<u>almDefineParam_tranParaLabel</u>	1700
<u>almGetModuleName</u>	1701
<u>almGetNamePrefix</u>	1702
<u>almGetParameterList</u>	1703
<u>almGetTerminalList</u>	1704
<u>almGetTerminalMap</u>	1705
<u>almSetTerminalMap</u>	1706
<u>almGetOpPointParamMap</u>	1707
<u>almSetOpPointParamMap</u>	1708

Virtuoso ADE SKILL Reference

<u>almGetNetlistProcedure</u>	1709
<u>almGetViewInfoNameList</u>	1710
<u>almGetNetlistType</u>	1711
<u>almHasViewInformation</u>	1712
<u>almSetNamePrefix</u>	1713
<u>almSetModuleName</u>	1714
<u>almSetNetlistProcedure</u>	1715
<u>almSetParameterList</u>	1716
<u>almSetTerminalList</u>	1717
<u>almSetPropMappingList</u>	1718
<u>almGetPropMappingList</u>	1719
<u>almSetOtherParameterList</u>	1720
<u>almGetOtherParameterList</u>	1721
<u>almGetStringParameterList</u>	1722
<u>almSetStringParameterList</u>	1723
<u>ancGetSimInstName</u>	1724
<u>ancAdjustNameCase</u>	1725
<u>drbBrowseFormCB</u>	1726
<u>msgHelp</u>	1727
<u>addCheck</u>	1728
<u>deleteChecks</u>	1731
<u>densityEstimateWaveform</u>	1732
<u>disableAllChecks</u>	1733
<u>disableChecks</u>	1734
<u>disableDeviceChecking</u>	1735
<u>displayChecks</u>	1736
<u>enableAllChecks</u>	1737
<u>enableChecks</u>	1738
<u>enableDeviceChecking</u>	1739
<u>setDevCheckOptions</u>	1740
<u>printViolations</u>	1741
<u>captabSummary</u>	1743
<u>evmOFDM</u>	1744
<u>relxOption</u>	1746
<u>asiAddModelLibSelection</u>	1747
<u>asiRemoveAllModelLibSelection</u>	1748

19

Functions to Access Simulation Results 1750

<u>Data Access Functions</u>	1751
<u>asiDefineDataAccessFunction</u>	1753
<u>asiDefineDataMappingFunction</u>	1755
<u>asiGetCalcResultsDir</u>	1757
<u>asiInit<yourSimulator>DataAccessFunction</u>	1758
<u>VAR</u>	1759
<u>DATA</u>	1760
<u>VS</u>	1761
<u>OP</u>	1762
<u>OPT</u>	1763
<u>MP</u>	1764
<u>NG</u>	1765
<u>VN</u>	1766
<u>VN2</u>	1767
<u>VNP</u>	1768
<u>VNPP</u>	1769
<u>VPD</u>	1770
<u>VF</u>	1771
<u>VT</u>	1772
<u>IS</u>	1773
<u>IT</u>	1774
<u>IF</u>	1775
<u>IDC</u>	1776
<u>VDC</u>	1777
<u>SIMULATOR</u>	1778
<u>Selection Functions</u>	1779
<u>asiSelectAnalysisCompParam</u>	1781
<u>asiSelectAnalysisInst</u>	1783
<u>asiSelectAnalysisNet</u>	1784
<u>asiSelectAnalysisSource</u>	1785
<u>asiSelectInst</u>	1787
<u>asiSelectNet</u>	1789
<u>asiSelectSourceInst</u>	1791

Virtuoso ADE SKILL Reference

<u>asiSelectTerm</u>	1794
<u>asiSelectTermNet</u>	1796
<u>Waveform Data Object Functions</u>	1799
<u>drAddElem</u>	1801
<u>drGetElem</u>	1802
<u>drSetElem</u>	1803
<u>drCreateVec</u>	1804
<u>drCreateEmptyWaveform</u>	1805
<u>drCreateWaveform</u>	1806
<u>drGetWaveformXType</u>	1807
<u>drGetWaveformXVec</u>	1808
<u>drGetWaveformYType</u>	1809
<u>drGetWaveformYVec</u>	1810
<u>drPutWaveformXVec</u>	1811
<u>drPutWaveformYVec</u>	1812
<u>drIsDataVector</u>	1813
<u>drIsParamWave</u>	1814
<u>drIsWaveform</u>	1815
<u>drType</u>	1816
<u>drVectorLength</u>	1817
<u>famAddValue</u>	1818
<u>famCreateFamily</u>	1819
<u>famGetSweepName</u>	1820
<u>famGetSweepValues</u>	1821
<u>famIsFamily</u>	1822
<u>famMap</u>	1823
<u>famValue</u>	1825
<u>Direct Plot Functions for RF Analyses</u>	1827
<u>drplMcpValue</u>	1829
<u>drplWrlsAcprValue</u>	1830
<u>drplEvmWrls</u>	1831
<u>drplACPRWithMask</u>	1832
<u>drplEvmBpsk</u>	1833
<u>drplPacVolGnExpDen</u>	1836
<u>drplJitter</u>	1837
<u>drplRFJc</u>	1839

Virtuoso ADE SKILL Reference

<u>drplRFJcc</u>	1841
<u>drplParamSweepRFJc</u>	1843
<u>drplParamSweepRFJcc</u>	1845
<u>drplRFValueAt</u>	1847
<u>ddrplPssHarmThd</u>	1848
<u>drplSwpHp</u>	1850
<u>drplSwpSp</u>	1851
<u>drplSwpYp</u>	1852
<u>drplSwpZm</u>	1853
<u>drplSwpZp</u>	1854
<u>drplSprobeHPM</u>	1855
<u>drplSprobePortSPData</u>	1857
<u>drplSprobePortVSWR</u>	1859
<u>drplSprobeYPM</u>	1861
<u>drplSprobeZM</u>	1863
<u>drplSprobeZPM</u>	1865
<u>drplGetMeas</u>	1867
<u>drplGetSpectrum</u>	1870
<u>OCEAN Script Functions</u>	1873
<u>asiOpenOceanScript</u>	1874
<u>asiWriteOceanScript</u>	1875
<u>asiCloseOceanScript</u>	1877

20

<u>Simulation Environment Functions</u>	1878
<u>sevGetSessionType</u>	1879
<u>sevSetMainWindowPulldownMenus</u>	1880
<u>sevSetMTSMode</u>	1881
<u>sevMTSMode</u>	1882
<u>sevMTSOptions</u>	1883
<u>sevOpenXterm</u>	1884
<u>sevSetSchematicPulldownMenus</u>	1885
<u>sevSetTypeInWindowPulldownMenus</u>	1886
<u>sevSetMenuitemLists</u>	1887
<u>sevAddMenuitemLists</u>	1888

Virtuoso ADE SKILL Reference

<u>sevDirectPlotMenu</u>	1889
<u>sevEnvironment</u>	1890
<u>sevNoEnvironment</u>	1891
<u>sevSaveState</u>	1892
<u>sevLoadState</u>	1893
<u>sevSaveOceanScript</u>	1894
<u>sevEditOptions</u>	1895
<u>sevOpenSchematic</u>	1896
<u>sevMenuItems</u>	1897
<u>sevReset</u>	1898
<u>sevQuit</u>	1899
<u>sevCreateMainWindow</u>	1900
<u>sevChooseSimulator</u>	1901
<u>sevChooseTemperature</u>	1902
<u>sevMpuTool</u>	1903
<u>sevChooseEnvironmentOptions</u>	1904
<u>sevEditStimulus</u>	1905
<u>sevNonMixedSignal</u>	1906
<u>sevEditSimulationFile</u>	1907
<u>sevChooseDesign</u>	1908
<u>sevEditSelectedAnas</u>	1909
<u>sevEditSelectedVars</u>	1910
<u>sevEditSelectedOuts</u>	1911
<u>sevChangeOutsOnSchematic</u>	1912
<u>sevSaveOptions</u>	1913
<u>sevDeleteSelectedAnas</u>	1914
<u>sevNoAnaSelections</u>	1915
<u>sevActivateSelectedAnas</u>	1916
<u>sevDeleteSelectedVars</u>	1917
<u>sevNoVarSelections</u>	1918
<u>sevFindSelectedVars</u>	1919
<u>sevCopyCellViewVariables</u>	1920
<u>sevCopyVariablesToCellView</u>	1921
<u>sevDeleteSelectedOuts</u>	1922
<u>sevExportOutputsToTxt</u>	1923
<u>sevImportOutputsFromTxt</u>	1924

Virtuoso ADE SKILL Reference

<u>sevExportOutputsToCSV</u>	1925
<u>sevExportOutputsToFile</u>	1926
<u>sevImportOutputsFromCSV</u>	1927
<u>sevImportOutputsFromFile</u>	1928
<u>sevNoOutSelections</u>	1929
<u>sevRemovePlotWindow</u>	1930
<u>sevSetPropertyForSelectedOuts</u>	1931
<u>sevSimulator</u>	1932
<u>sevRunEngine</u>	1933
<u>sevStopEngine</u>	1934
<u>sevIsContinuable</u>	1935
<u>sevSetEngineOptions</u>	1936
<u>sevNetlistFile</u>	1937
<u>sevOpenEncap</u>	1938
<u>sevViewSimulatorOutput</u>	1939
<u>sevNoOutputLog</u>	1940
<u>sevConvergence</u>	1941
<u>sevNoResults</u>	1942
<u>sevNoPlottableOutputs</u>	1943
<u>sevCircuitCond</u>	1944
<u>sevNoDesign</u>	1945
<u>sevSetSimDataDir</u>	1946
<u>sevSaveResults</u>	1947
<u>sevSelectResults</u>	1948
<u>sevDeleteResults</u>	1949
<u>sevEditPlottingOptions</u>	1950
<u>sevPlotAllOutputs</u>	1951
<u>sevNoPlottableSignals</u>	1952
<u>sevPlotSignals</u>	1953
<u>sevEvaluateAndPlotExpressions</u>	1954
<u>sevNoPlottableExpressions</u>	1955
<u>sevPrintResults</u>	1956
<u>sevRetrieveFromEffectiveCDF</u>	1957
<u>sevAnnotateResults</u>	1958
<u>sevRegisterPcellsForAnnotation</u>	1959
<u>sevGetRegisteredPcellsForAnnotation</u>	1960

Virtuoso ADE SKILL Reference

<u>sevParametricTool</u>	1961
<u>sevCornersTool</u>	1962
<u>sevMonteCarloTool</u>	1963
<u>sevOptimizationTool</u>	1964
<u>sevOpenCalculator</u>	1965
<u>sevOpenDRLBrowser</u>	1966
<u>sevOpenPlotWindow</u>	1967
<u>sevOpenPrintWindow</u>	1968
<u>sevOpenJobMonitor</u>	1969
<u>sevIcon</u>	1970
<u>sevDeleteSelections</u>	1971
<u>sevWhatsNew</u>	1972
<u>sevStartSession</u>	1973
<u>sevAboutTool</u>	1974
<u>sevEditModels</u>	1975
<u>sevSetupStimuli</u>	1976
<u>sevSetupSimulationFiles</u>	1977
<u>sevNetlistAndRun</u>	1978
<u>sevRun</u>	1979
<u>sevNetlistAndDebug</u>	1980
<u>sevDebug</u>	1981
<u>sevLMGTool</u>	1982
<u>sevPKGTool</u>	1983
<u>sevKmodelTool</u>	1984
<u>sevPCMTTool</u>	1985
<u>sevBPMTTool</u>	1986
<u>sevBALMTool</u>	1987
<u>sevActiveSelectedAna</u>	1988
<u>sevNonActiveSelectedAna</u>	1989
<u>sevSession</u>	1990
<u>sevSetTopSaveDir</u>	1991
<u>sevTopSaveDir</u>	1992
<u>sevDisplayViolations</u>	1993
<u>sevNoViolationsFound</u>	1994
<u>sevParasiticsDisplayed</u>	1995
<u>sevParasiticsNotDisplayed</u>	1996

Virtuoso ADE SKILL Reference

<u>sevDevChecking</u>	1997
<u>sevSetSolver</u>	1998
<u>sevSetConnectModules</u>	1999
<u>sevInvokeNCBrowse</u>	2000
<u>sevInvokeSimvision</u>	2001
<u>sevInvokeSimvisionDebugger</u>	2002
<u>sevNoLog</u>	2003
<u>sevViewNetlisterLog</u>	2004
<u>sevViewCompilerLog</u>	2005
<u>sevViewElabLog</u>	2006
<u>sevViewNcverilogLog</u>	2007
<u>sevViewSimLog</u>	2008
<u>sevReturnVariablesWithEmptyValues</u>	2009
<u>sevAddExpression</u>	2010
<u>sevGetExpressions</u>	2011
<u>sevDeleteSelectedSubckts</u>	2012
<u>sevDeleteSelectedOpPoints</u>	2013

21

CDF Functions 2014

CDF SKILL Function Properties 2016

<u>cdfCreateBaseCellCDF</u>	2028
<u>cdfCreateBaseLibCDF</u>	2030
<u>cdfCreateParam</u>	2032
<u>cdfCreateUserCellCDF</u>	2038
<u>cdfCreateUserLibCDF</u>	2040
<u>cdfGetBaseLibCDF</u>	2042
<u>cdfGetUserLibCDF</u>	2043
<u>cdfGetLibCDF</u>	2044
<u>cdfGetBaseCellCDF</u>	2045
<u>cdfGetUserCellCDF</u>	2046
<u>cdfGetCellCDF</u>	2047
<u>cdfGetInstCDF</u>	2048
<u>cdfSaveCDF</u>	2049
<u>cdfDump</u>	2050

Virtuoso ADE SKILL Reference

<u>cdfDumpAll</u>	2052
<u>cdfDeleteCDF</u>	2054
<u>cdfDeleteParam</u>	2055
<u>cdfCopyCDF</u>	2056
<u>cdfCopyParam</u>	2058
<u>cdfFindParamByName</u>	2059
<u>cdfUpdateInstParam</u>	2060
<u>cdfRefreshCDF</u>	2061
<u>aedCopyCDF</u>	2062
<u>aedDeleteCDF</u>	2063
<u>cdfGetUnitScaleFactor</u>	2064
<u>cdfSetUnitScaleFactor</u>	2065
<u>cdfEditScaleFactors</u>	2066
<u>cdfEnableScaleFactorRetentionForZero</u>	2067
<u>cdfDisableScaleFactorRetentionForZero</u>	2068
<u>cdfParseFloatString</u>	2069
<u>cdfFormatFloatString</u>	2070
<u>cdfSyncInstParamValue</u>	2071
<u>cdfUpdateInstSingleParam</u>	2072
<u>aedEditCDF</u>	2073
<u>cdfGetCustomViaCDF</u>	2074
<u>cdfUpdateCustomViaParam</u>	2075
<u>cdfRegRemasterTrigger</u>	2076
<u>cdfRemasterAction</u>	2078
<u>cdfUnregRemasterTrigger</u>	2082

Virtuoso ADE SKILL Reference

Introduction to Maestro Functions

This chapter provides an introduction to SKILL scripts that you can write for maestro cellviews. In this chapter, you can find information about

- [Creation of SKILL Scripts for Maestro Cellviews](#)
- [Executing Scripts for ADE Explorer and ADE Assembler](#)

Creation of SKILL Scripts for Maestro Cellviews

You can create scripts for maestro cellviews by using the various SKILL functions described in this book.

Important Points to Note

- The SKILL functions that have the prefix `mae`, can be used with the cellviews of type `maestro` that are saved using either ADE Explorer or ADE Assembler. When these functions are executed from the user interface (CIW), they get executed on the complete cellview, irrespective of the application in which a cellview is currently opened. For example, if you open a cellview that contains multiple tests in the ADE Explorer environment and run the `maeGetSetup` function to get the setup details, it returns the details of all the tests defined in the cellview instead of returning the details of the test that is opened in the ADE Explorer window.
- The functions listed in this user guide cannot be used to modify or save changes in ADE XL cellviews.

Executing Scripts for ADE Explorer and ADE Assembler

If the user interface for Virtuoso ADE Explorer or ADE Assembler is already open, you can directly run the SKILL functions for maestro cellviews from the Command Interpreter Window (CIW).

Virtuoso ADE SKILL Reference

Introduction to Maestro Functions

Alternatively, you can write a script using these functions in a `.il` file and execute by using the `virtuoso` executable, as shown in the following example:

Step 1: Write the following script in `test.il` file

```
maeOpenSetup("solutions" "amptest" "maestro")
maeRunSimulation()
maeWaitUntilDone('All)
maeExportOutputView()
```

Step 2: Execute the script

```
virtuoso -replay test.il -log myLog1 -report
```

The `-report` command line argument prints verbose messages on the command line.

If you include any call to the `printf` function in a script, the output of `printf` is written to the main Virtuoso log.

Important

The execution of the script does not break if any command fails to execute successfully. Instead, the next command is run. Therefore, it is a good practice to view the report or run log after the complete script is run.

Cadence SKILL API Finder

Use the Cadence SKILL API Finder tool to display the description and syntax of the SKILL functions. To open the SKILL API Finder from the CIW, select *Tools – SKILL Finder*. The Cadence SKILL API Finder appears.

Important

For information about using the SKILL API Finder, refer to the [Using SKILL API Finder](#) appendix in the *Cadence SKILL IDE User Guide* or click *Help – Finder Help* in the Finder.

Maestro Cellview Functions

This topic lists the SKILL functions that let you work with maestro cellviews in ADE Explorer and ADE Assembler.

These functions are broadly classified in the following categories:

- [Functions to Migrate ADE L or ADE XL States to Maestro Cellviews](#)
- [Functions to Create, View, Edit, and Save Setups in maestro Cellviews](#)
- [Functions to Work with Simulation Results](#)
- [Functions for Checks and Asserts](#)
- [Functions to Manage Run Plans](#)
- [Functions to Work with Plotting Templates](#)
- [Functions for Sensitivity Analysis Setup](#)
- [Functions for XML File Management](#)
- [Functions to Work with the Locally Scoped Models and Options \(MTS Options\)](#)

Functions to Migrate ADE L or ADE XL States to Maestro Cellviews

This topic lists the SKILL functions that let you migrate ADE L or ADE XL states.

[maeConvertAndCombineMultiADELToAssembler](#)

[maeMigrateADELStateToMaestro](#)

[maeMigrateADEXLToMaestro](#)

maeConvertAndCombineMultiADELToAssembler

```
maeConvertAndCombineMultiADELToAssembler(  
    l_stateList  
    t_maestroLib  
    t_maestroCell  
    [ ?maestroView t_maestroView ]  
    [ ?migrateFrom t_migrateFrom ]  
    [ ?rootPath t_rootPath ]  
    [ ?testNameList t_testNameList ]  
    [ ?overwrite g_overwrite ]  
)  
=> l_cellviewDetails / nil
```

Description

Combines two or more ADE L states into one maestro cellview for ADE Assembler. In the resulting maestro cellview, the tests are created in the same order in which the states are specified in the list of states to be migrated.

Arguments

<i>l_stateList</i>	The list of ADE L states to be combined and migrated to a maestro cellview.
<i>t_maestroLib</i>	Name of the library in which the maestro cellview is to be saved.
<i>t_maestroCell</i>	Name of the cell in which the maestro cellview is to be saved.
<i>?maestroView t_maestroView</i>	Name for the maestro cellview to be saved. The default value is "maestro".
<i>?migrateFrom t_migrateFrom</i>	The method to find the given ADE L states. The possible values are 'directory', 'cellview'. The default value is 'cellview'.
<i>?rootPath t_rootPath</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

The path that contains the libraries of ADE L states. This argument is used when the `?migrateFrom` argument is set to `'directory`.

The default value is `nil`.

`?testNameList l_testNameList`

List of names to be assigned to the tests created using the states migrated from ADE L.

The default value is `" "`.

`?overwrite g_overwrite`

Boolean value that specifies if an existing view with the same name is to be overwritten by the new cellview.

The default value is `t`.

Value Returned

`l_cellviewDetails` A list containing the names of library, cell, and view of the maestro cellview created by this function is returned.

`nil` Unable to combine the states.

Examples

Migrates the ADE L states to create a maestro cellview.

```
maeConvertAndCombineMultiADELToAssembler((list (list "Two_Stage_Opamp"
"OpAmp_AC_top" "spectre_AC") (list "Two_Stage_Opamp" "OpAmp_TRAN_top"
"spectre_TRAN")) "Two_Stage_Opamp" "OpAmp" ?maestroView "maestro_new")
```

```
=> ("Two_Stage_Opamp" "OpAmp" "maestro_new")
```

```
maeConvertAndCombineMultiADELToAssembler((list (list "Two_Stage_Opamp"
"OpAmp_AC_top" "spectre_AC" "spectre") (list "Two_Stage_Opamp" "OpAmp_TRAN_top"
"spectre_TRAN" "spectre")) "Two_Stage_Opamp" "OpAmp" ?migrateFrom 'directory
?rootPath "~/artist_states/"))
```

Migrates the ADE L states, `spectre_state2` and `stateA` to a new maestro cellview, `maestro_new6`. The two tests in `maestro_new6` are named as `testa` and `testb`.

```
maeConvertAndCombineMultiADELToAssembler((list (list "Two_Stage_Opamp" "OpAmp"
"spectre_state2") (list "Two_Stage_Opamp" "OpAmp" "stateA") ) "Two_Stage_Opamp"
"OpAmp" ?maestroView "maestro_new6" ?testNameList list("testa" "testb"))
```

maeMigrateADELStateToMaestro

```
maeMigrateADELStateToMaestro(  
    t_stateLib  
    t_stateCell  
    t_stateName  
    [ ?maestroLib t_maestroLib ]  
    [ ?maestroCell t_maestroCell ]  
    [ ?maestroView t_maestroView ]  
    [ ?migrateFrom s_migrateFrom ]  
    [ ?statePath t_statePath ]  
    [ ?simulator t_simulator ]  
    [ ?overwrite g_overwrite ]  
)  
=> l_cellviewDetails / nil
```

Description

Migrates the given ADE L state to a maestro cellview that can be opened in ADE Explorer.

Arguments

<i>t_stateLib</i>	Name of the library for which the ADE L state is available.
<i>t_stateCell</i>	Name of the cell for which the ADE L state is available.
<i>t_stateName</i>	Name of the view or state file for which the ADE L state is available.
<i>?maestroLib t_maestroLib</i>	Name of the library in which the maestro cellview is to be saved. If not specified, the value given for <i>t_stateLib</i> is used. You can specify any other library name. If the given library does not exist, a library is created with that name.
<i>?maestroCell t_maestroCell</i>	Name of the cell in which the maestro cellview is to be saved. If not specified, the value specified for <i>t_stateCell</i> is used.
<i>?maestroView t_maestroView</i>	Name for the maestro cellview to be saved. The default value is "maestro".

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?migrateFrom s_migrateFrom`

Specifies if the state is to be migrate from the cellview or directory.

The possible values are 'cellview or 'directory.

The default value is 'cellview.

`?statePath t_statePath`

Path to the directory if *s_migrateFrom* is set to 'directory.

`?simulator t_simulator`

Specifies the simulator name. This value is used when *s_migrateFrom* is set to 'directory.

The default value is "spectre".

`?overwrite g_overwrite`

Specifies whether to overwrite the cellview, if already exists.

Value Returned

l_cellviewDetails A list containing the names of library, cell and view of the maestro cellview to which the state is successfully migrated.

nil Unable to migrate the given ADE L state to a maestro cellview.

Examples

Migrates the ADE L state *spectre_statel* to a maestro state *maestro1*.

```
maeMigrateADELStateToMaestro("Two_Stage_Opamp" "OpAmp_AC_top" "spectre_statel" )  
=> ("Two_Stage_Opamp" "OpAmp_AC_top" "maestro1")
```

Migrates an ADE L state from a state file saved in a directory.

```
maeMigrateADELStateToMaestro("Two_Stage_Opamp" "OpAmp_AC_top" "AC_active"  
?migrateFrom 'directory ?statePath "./artist_states")
```

```
=> ("Two_Stage_Opamp" "OpAmp_AC_top" "maestro1")  
; the maestro cellview is created from the following state file:  
; ./artist_states/Two_Stage_Opamp/OpAmp_AC_top/spectre/AC_active
```


maeMigrateADEXLToMaestro

```
maeMigrateADEXLToMaestro(  
    t_stateLib  
    t_stateCell  
    t_stateName  
    [ ?maestroLib t_maestroLib ]  
    [ ?maestroCell t_maestroCell ]  
    [ ?maestroView t_maestroView ]  
    [ ?overwrite g_overwrite ]  
    [ ?skipHistory g_skipHistory ]  
)  
=> t / nil
```

Description

Migrates the given ADE XL view to a new maestro cellview that can be opened in ADE Assembler.

Arguments

<i>t_stateLib</i>	Name of the library for the ADE XL view.
<i>t_stateCell</i>	Name of the cell for the ADE XL view.
<i>t_stateName</i>	Name of ADE XL cellview.
<i>?maestroLib t_maestroLib</i>	Name of the library to which the new maestro cellview is to be saved. If not specified, the value given for <i>t_stateLib</i> is used. You can specify any other library name. If the given library does not exist, a library is created with that name.
<i>?maestroCell t_maestroCell</i>	Name of the cell in which the maestro cellview is to be saved. If not specified, the value specified for <i>t_stateCell</i> is used.
<i>?maestroView t_maestroView</i>	Name of the maestro cellview to be created. The default value is "maestro".
<i>?overwrite g_overwrite</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Boolean value that specifies whether or not to overwrite an existing maestro view with the same name as that specified by the `?maestroView` argument.

- `t`: Overwrites an existing maestro view with the same name as that specified by the `?maestroView` argument. This is the default.
- `nil`: The cellview is not migrated if an existing maestro cellview is found with the same name.

`?skipHistory g_skipHistory`

Boolean value that specifies whether or not to skip histories while copying data from the `adexl` cellview into the new maestro cellview.

- `t`: Skips histories while copying data from the `adexl` cellview into the new maestro cellview..
- `nil`: Histories are copied from the `adexl` cellview into the new maestro cellview. This is the default.

Value Returned

<code>t</code>	The ADE XL view is migrated to a new maestro view.
<code>nil</code>	Unable to migrate the given ADE XL view to a new maestro cellview

Examples

Migrates the `adexl` cellview to a new cellview named `maestro`.

```
maeMigrateADEXLToMaestro("solutions" "ampTest" "adexl")  
=> t
```

Functions to Create, View, Edit, and Save Setups in maestro Cellviews

This topic lists the SKILL functions that let you work with the setup in maestro cellviews.

[maeAddOutput](#)

[maeCreateTest](#)

[maeCloseSession](#)

[maeConvertViewForReferencedAndLocalRunPlanCorners](#)

[maeConvertViewForSeparateHistoryManagement](#)

[maeCloseCornersSetupForm](#)

[maeDeleteCorner](#)

[maeDeleteExplorerHistory](#)

[maeDeleteOutput](#)

[maeDeleteParameter](#)

[maeDeleteTags](#)

[maeDeleteVar](#)

[maeExportCornersToCSV](#)

[maeExportCornersToSDB](#)

[maeExportGlobalVariablesToCSV](#)

[maeExportSetupForExplorer](#)

[maeGetAllExplorerHistoryNames](#)

[maeGetAnalysis](#)

[maeGetCurrentRunMode](#)

[maeGetEnabledAnalysis](#)

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

maeGetEnvOption

maeGetExplorerTestName

maeGetHistoryLockFlag

maeGetSessions

maeGetSetup

maeGetSetupLibrary

maeGetTags

maeGetTestEnvVar

maeGetTestSession

maeGetVar

maeImportOutputsFromFileInDocuments

maeImportSetupForExplorer

maelsPrerunScriptEnabled

maelsSetupModified

maelsSingleTest

maelsValidMaestroSession

maeLoadCorners

maeLoadSetupState

maeLoadStateForTest

maeMakeEditable

maeMakeReadonly

maeOpenSetup

maeRegTimer

maeRunDspfChecker

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

maeSaveSetup

maeSaveSetupState

maeSetAnalysis

maeSetCorner

maeSetCurrentRunMode

maeSetDesign

maeSetDesignForTest

maeSetEnableTestVar

maeSetEnvOption

maeSetHistoryLock

maeSetParameter

maeSetSpec

maeSetSetup

maeSetSetupLibrary

maeSetSimOption

maeSetTags

maeSetTestEnvVar

maeSetVar

maeStmGenerateWaveforms

maeAddOutput

```
maeAddOutput (
    t_outputName
    t_testName
    [ ?outputType t_outputType ]
    [ ?signalName t_signalName ]
    [ ?expr t_expr ]
    [ ?plot g_plot ]
    [ ?save g_save ]
    [ ?session t_sessionName ]
)
=> t / nil
```

Description

Adds an output for the given test in the currently active setup.

Arguments

t_outputName Name of the output to be added.

Note: Each output for a test must have a unique name. To avoid adding duplicate output names, you can specify "" and update the output name later in the user interface.

t_testName Name of the test in which output has to be added.

?outputType t_outputType
Type of the output to be added

The possible values are:

- "point" (default)
- "corners"
- "sweeps"
- "all"
- "terminal"
- "terminalV"
- "net"

?signalName t_signalName

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

	Name of the signal to be added. This is relevant only when <code>?outputType</code> is set to "terminal", "terminalV", or "net".
	The default value is " " .
<code>?expr t_expr</code>	Expression to be used to calculate the output. You can create an expression for all evaluation types. For example, "point", "corners", "sweeps", and "all".
<code>?plot g_plot</code>	Specifies if the output is to be plotted.
<code>?save g_save</code>	Specifies if the output is to be saved.
<code>?session t_sessionName</code>	Name of the session. If not specified, output is added in the current session.

Value Returned

<code>t</code>	Adds the specified output for the given test in the currently active setup.
<code>nil</code>	The specified output is not added for the given test in the currently active setup.

Examples

Adds an output of type `expr` to `test1`.

```
maeAddOutput("Output1" "test1" )
```

Adds an output of type `signal` for `test1`.

```
maeAddOutput("Output2" "test1" ?outputType "point" ?expr "Output1*10")
```

Adds an output of type `net` for `test1`.

```
maeAddOutput("Output3" "test1" ?outputType "net" ?signalName "/net2")
```

Adds the terminal voltage for signal `I0/M1/S` to the output list for test `test1`.

```
maeAddOutput("TerminalOutput2" "test1" ?outputType "terminalV" ?signalName  
"/I0/M1/S")  
maeSaveSetup()
```

maeApplyCornerEditingOptions

```
maeApplyCornerEditingOptions(  
    [ ?session g_sessionName ]  
)  
=> t / nil
```

Description

Applies the updates specified through `maeSetCornerEditingOptions` to the Corners Setup form.

Arguments

`?session g_sessionName`

Name of the session.

If not specified, the updates are made to the Corners Setup form for the current session.

Value Returned

`t` The updates specified through `maeSetCornerEditingOptions` are applied.

`nil` No corner editing options are set.

Examples

Applies the updates specified through `maeSetCornerEditingOptions` to the Corners Setup form.

```
maeSetCornerEditingOptions(?cornerNameFilter "SLOW FAST" ?removeVarList  
list("var1") ?removeDuplicates t ?removeUndefined t)  
=> t
```

```
maeApplyCornerEditingOptions()  
=>t
```

```
maeSaveSetup()
```

Related Topics

Importing Corners

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

maeGetCornerEditingOptions

maeResetCornerEditingOptions

maeSetCornerEditingOptions

maeCreateTest

```
maeCreateTest(  
    t_testName  
    [ ?sourceTest t_sourceTest ]  
    [ ?lib t_lib ]  
    [ ?cell t_cell ]  
    [ ?view t_view ]  
    [ ?simulator t_simulator ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Creates a new test and adds it to the given maestro session. The design name for this test is set using the library, cell, and view name. If a source test name is given, it creates a copy of that test.

Arguments

<code>t_testName</code>	Name of the test to be created.
<code>?sourceTest t_sourceTest</code>	Name of the source test to be copied to create the new test.
<code>?lib t_lib</code>	Name of the library for the new test is to be created.
<code>?cell t_cell</code>	Name of the cell for the new test is to be created.
<code>?view t_view</code>	Name of the view for the new test is to be created.
<code>?simulator t_simulator</code>	Name of the simulator to be used for the new test. The default value is "spectre".
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>t</code>	The test is created.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

nil Unable to create the test.

Examples

Creates a copy of the test `solution:ampTest:1` and names it `solutions:ampTest:2`.

```
maeCreateTest("solutions:ampTest:2" ?sourceTest "solution:ampTest:1")
maeSaveSetup()
```

Creates a new test `solutions:ampTest:2`, sets `solutions/ampTest/schematic` as the design and Spectre as simulator.

```
maeCreateTest("solutions:ampTest:2" ?lib "solutions" ?cell "ampTest" ?view
"schematic" ?simulator "spectre")
maeSaveSetup()
```

maeCloseSession

```
maeCloseSession(  
    [ ?session t_sessionName ]  
    [ ?forceClose g_forceClose ]  
)  
=> t / nil
```

Description

Closes the sessions opened using [maeOpenSetup](#) in the SKILL code. This function cannot be used to close the sessions opened from the Virtuoso user interface.

Arguments

`?session t_sessionName`

Name of the session to be closed.

If not specified, the current session is closed.

`?forceClose g_forceClose`

A Boolean value that specifies whether the current session needs to be closed immediately or not if simulations are pending. In both cases, appropriate information messages are added to the log.

- `t`: Immediately stops the simulations that are in progress and closes the session.
- `nil`: Waits for the simulations that are in progress to complete and then closes the session. This is the default.

Value Returned

`t` Closes the sessions opened using `maeOpenSetup` in the SKILL code.

`nil` Could not close the session.

Examples

Closes the session `fnxSession2` that is opened using the function `maeOpenSetup` and edited using the function `maeSetSetup`.

```
; load the setup
maeOpenSetup("solutions" "ampTest" "maestro")
=>"fnxSession2"

; check which tests are there in setup
maeGetSetup()
=> ("solutions:amptest:1" "solutions:amptest:2")

; Enable a test "solution:amptest:1"
maeSetSetup(?tests '("solutions:amptest:1"))

; Set the value of global variable
"CAP" as 5p: maeSetVar("CAP" 5p)

; Disable corners C0 and C1
maeSetSetup(?corners '("C0" "C1") ?enabled nil)

; save changes in the cellview
maeSaveSetup()
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
; Run Simulation
maeRunSimulation()

maeCloseSession()
=> t

; The tool waits for the simulation to complete before closing the session
```

Related Topics

[maeOpenSetup](#)

maeConvertViewForReferencedAndLocalRunPlanCorners

```
maeConvertViewForReferencedAndLocalRunPlanCorners (  
    t_libName  
    t_cellName  
    t_viewName  
)  
=> t / nil
```

Description

Converts all local and referenced corners created in cellviews saved using IC6.1.8 ISR9 or ICADVM18.1 ISR9 to referenced corners supported in earlier versions.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the maestro cellview.

Value Returned

<i>t</i>	Successful conversion of the run plan.
<i>nil</i>	Unsuccessful conversion of the run plan.

Examples

Converts all local and referenced corners created in the `solutions/ampTest/maestro` cellview saved using IC6.1.8 ISR9 or ICADVM18.1 ISR9 to referenced corners supported in earlier versions.

```
maeConvertViewForReferencedAndLocalRunPlanCorners("solutions" "ampTest"  
"maestro")  
=>t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

maeConvertViewForSeparateHistoryManagement

```
maeConvertViewForSeparateHistoryManagement (
    t_libName
    t_cellName
    t_viewName
    [ ?newLib t_newLibName ]
    [ ?newCell t_newCellName ]
    [ ?newView t_newViewName ]
    [ ?overwrite g_overwrite ]
)
=> t / nil
```

Description

Converts the maestro views in which history setup is integrated with the main setup database to an enhanced format in which separate history management is enabled.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the maestro cellview.
<i>?newView t_newLibName</i>	Name of the library in which the converted cellview is to be saved.
<i>?newCell t_newCellName</i>	Name of the cell in which the converted cellview is to be saved.
<i>?newView t_newViewName</i>	Name to be used for the converted maestro cellview.
<i>?overwrite g_overwrite</i>	A Boolean value specifying whether to overwrite <i>t_viewName</i> with the converted cellview.

Value Returned

<i>t</i>	The cellview is converted.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`nil` The cellview is not converted.

Examples

Converts the maestro view `solution/ampTest/maestro` to an enhanced format in which separate history management is enabled.

```
maeConvertViewForSeparateHistoryManagement("solutions" "ampTest" "maestro"  
?newView "maestro_new")  
=> t
```

maeCloseCornersSetupForm

```
maeCloseCornersSetupForm(  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Closes the Corners Setup from.

Arguments

?session t_sessionName
Name of the session.

Value Returned

t	The Corners Setup from is closed.
nil	The given session or form is not found.

Examples

The following function closes the Corners Setup form opened in the given session:

```
maeCloseCornersSetupForm("fnxSession0")  
=> t
```

maeDeleteCorner

```
maeDeleteCorner(  
    t_cornerName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified corner from the setup database.

You cannot delete the nominal corner using this function.

Arguments

<i>t_cornerName</i>	Name of the corner to be deleted from the setup.
<i>?session t_sessionName</i>	Name of the session. If not specified, the corner is deleted from the current session.

Value Returned

<i>t</i>	The specified corner gets deleted from the setup database.
<i>nil</i>	The specified corner is not deleted.

Examples

Deletes the corner C1 from the `maestro` cellview.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
maeDeleteCorner("C1")  
=> t  
maeSaveSetup()
```

maeDeleteMultiWildcardOutputs

```
maeDeleteMultiWildcardOutputs(  
    l_outputNamesWithWildCards  
    [ ?custom_column l_customColumnValue ]  
    [ ?customColumnName t_customColumnName ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the named outputs from the active setup. You can also delete outputs conditionally, based on a specified value in a custom column. You can use wildcard characters in the list of output names.

The specified arguments are case-sensitive.

Arguments

l_outputNamesWithWildCards

The list of outputs to be deleted.

You can use wildcards to delete outputs with matching patterns in the name.

?custom_column l_customColumnValue

The list of values defined in a custom column to be searched for matching values.

?customColumnName t_customColumnName

The name of the custom column to be searched for matching values.

?session t_sessionName

Name of the Virtuoso session.

If not specified, the current session is used.

Value Returned

t

The specified outputs are deleted from the active setup in the specified session.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`nil` No outputs with the matching criteria are found.

Examples

Let us consider the following setup.

Test	Name	Type	Details	MyColumn	EvalType	Plot	Save	Spec
Filter	Filter	Filter	Filter	Filter	Filter			Filter
Two_Stage_Opamp:OpAmp_AC_top:1	V1	signal (I)	/V1/PLUS	Group1	point	☒	☒	
Two_Stage_Opamp:OpAmp_AC_top:1	out	signal	/OUT	Group1	point	☒	☒	
Two_Stage_Opamp:OpAmp_AC_top:1	Current	expr	abs(I(("/V1/PLUS"))	Group1	point	☒	☐	< 1.5m
Two_Stage_Opamp:OpAmp_AC_top:1	UGF	expr	cross(dB20(mag(VF("/OUT")))) 0 1...	Group1	point	☒	☐	> 533M
Two_Stage_Opamp:OpAmp_AC_top:1	inm	signal	/inm	Group1	point	☒	☒	
Two_Stage_Opamp:OpAmp_AC_top:1	inp	signal	/inp	Group1	point	☒	☒	
Two_Stage_Opamp:OpAmp_AC_top:1	cmin	signal	/cmin	Group2	point	☐	☒	
Two_Stage_Opamp:OpAmp_AC_top:1	nomGain	expr	VAR("nomGain")	Group2	point	☒	☐	
Two_Stage_Opamp:OpAmp_AC_top:1	outdB20	expr	dB20(mag(VF("/OUT")))	Group2	point	☒	☐	
Two_Stage_Opamp:OpAmp_AC_top:1	SettlingTime	expr	settlingTime(VT("/OUT") 0 t 2.5e...	Group2	point	☒	☐	< 8n
Two_Stage_Opamp:OpAmp_AC_top:1	RelativeSwingPercent	expr	((Swing / VAR("vdd")) * 100)	Group2	point	☒	☐	> 75
Two_Stage_Opamp:OpAmp_AC_top:1	PhaseMargin	expr	getData("/phaseMargin" ?result ...	Group3	point	☒	☐	> 20
Two_Stage_Opamp:OpAmp_AC_top:1	Voffset	expr	(VDC("/inm") - VDC("/inp"))	Group1	point	☒	☐	range -10m 10m
Two_Stage_Opamp:OpAmp_AC_top:1	Swing	expr	(value(VT("/OUT") 2.5e-08) - valu...	Group2	point	☒	☐	> 0.98

Deletes all outputs with names that start with the letter S:

```
maeDeleteMultiWildCardOutputs(list("S*"))
maeSaveSetup()
```

The outputs `Swing` and `SettingTime` are deleted.

Deletes all outputs with names starting with the string `out` and the value `Group1` in the custom column `MyColumn`:

```
maeDeleteMultiWildCardOutputs(list("out*") ?customColumnValues list("Group1")
?custom_column "MyColumn")
maeSaveSetup()
```

The output `out` is deleted.

Delete all outputs with names containing the string `out` and the values `Group1` and `Group2` in the custom column `MyColumn`:

```
maeDeleteMultiWildCardOutputs(list("*out*") ?customColumnValues list("Group1"
"Group2") ?custom_column "MyColumn")
maeSaveSetup()
```

The outputs `out` and `outdB20` are deleted.

Deletes all outputs with names that start with the strings `Swing`, `off`, or `out`, and the values `Group1` and `Group2` in the custom column `MyColumn`:

```
maeDeleteMultiWildCardOutputs(list("Swing*" "off*" "out*") ?customColumnValues
list("Group2" "Group1") ?custom_column "MyColumn")
maeSaveSetup()
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

The outputs `Swing`, `out`, and `outdB20` are deleted.

maeDeleteOutput

```
maeDeleteOutput(  
    t_outputName  
    t_testName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified output from the setup database.

Arguments

<i>t_outputName</i>	Name of the output to be deleted from the setup.
<i>t_testName</i>	Name of the test for which the output is to be deleted.
<i>?session t_sessionName</i>	Name of the session. If not specified, the output is deleted from the current session.

Value Returned

<i>t</i>	The specified output gets deleted from the setup database.
<i>nil</i>	The specified output is not deleted.

Examples

Deletes an output from the setup database of the specified test.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
maeDeleteOutput("out1" "test1")  
=> t
```

Deletes multiple outputs from the setup database of the specified test.

```
maeDeleteOutput(list("out1" "out2" "out3") "test1")  
=> t
```


maeDeleteExplorerHistory

```
maeDeleteExplorerHistory(  
    t_sessionName  
    t_historyName  
)  
=> t / nil
```

Description

Deletes the specified ADE Explorer history from the given maestro session.

You can delete only one history at a time.

Arguments

<i>t_sessionName</i>	Name of the maestro session for which the history is to be deleted.
<i>t_historyName</i>	Name of the ADE Explorer history to be deleted.

Value Returned

<i>t</i>	The specified history is deleted from the given maestro session.
<i>nil</i>	The specified history or session name is invalid.

Examples

Deletes the history `ExplorerRun.0` from the maestro session `fnxSession0`.

```
maeDeleteExplorerHistory("fnxSession0" "ExplorerRun.0")  
=> t
```

maeDeleteParameter

```
maeDeleteParameter(  
    t_parameterName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified parameter from the setup database.

Arguments

<i>t_parameterName</i>	Name of the corner to be deleted from the setup.
<i>?session t_sessionName</i>	Name of the session. If not specified, the parameter is deleted from the current session.

Value Returned

<i>t</i>	The specified parameter is deleted from the setup database.
<i>nil</i>	The specified parameter is not deleted from the setup database.

Examples

Deletes the parameter `/I0/1` from the setup database.

```
maeDeleteParameter("/I0/1")  
=> t
```

maeDeleteTags

```
maeDeleteTags(  
    [ ?session t_sessionName ]  
    [ ?variables l_variableNames ]  
    [ ?corners l_cornerNames ]  
)  
=> t / nil
```

Description

Deletes the tags from the given list of variables and corners in the given session.

Arguments

?session t_sessionName

Name of the session.

If not specified, the tags are deleted from the current session.

?variables l_variableNames

A list of valid global or test variables from which the tags are to be deleted.

?corners l_cornerNames

A list of corners from which the tags are to be deleted.

Value Returned

t The tags are deleted from the given list of variables and corners.

nil The tags are not deleted.

Examples

Deletes the tags from two variables and two corners in the current session.

```
maeDeleteTags(?variables '("vdd" "vin") ?corners '("C0" "C1"))  
=> t
```

maeDeleteTest

```
maeDeleteTest(  
    t_testName  
    [ ?run t_run ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Deletes the given test from the active setup or the specified run in the run plan.

Arguments

<code>t_testName</code>	Name of the test to be deleted.
<code>?run t_run</code>	Name of the run from which the test is to be deleted. By default, this argument is set to <code>nil</code> . If not specified, the test is deleted from the active setup in the maestro cellview.
<code>?session t_session</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>t</code>	The test is deleted.
<code>nil</code>	The test is not deleted.

Examples

Deletes the test `testA` from the run `Run.0`.

```
maeDeleteTest("testA" ?run "Run.0")
```

Deletes the test `testA` from the active setup.

```
maeDeleteTest("testA")  
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Related Topics

[Modifying a Run in the Run Plan](#)

[maeDeleteTags](#)

[maeDeleteVar](#)

maeDeleteVar

```
maeDeleteVar(  
    t_varName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified variable from the setup database.

Arguments

<i>t_varName</i>	Name of the variable to be deleted from the setup.
<i>?session</i>	Name of the session.
<i>t_sessionName</i>	If not specified, the output is deleted from the current session.

Value Returned

<i>t</i>	The specified variable is deleted from the setup database.
<i>nil</i>	The specified variable is not deleted.

Examples

Deletes the variable VDD from the specified maestro session.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
maeDeleteVar("VDD")  
=> t
```

maeEnableCorner

```
maeEnableCorner(
    [ ?run t_run ]
    [ ?enable g_enable ]
    [ ?session t_session ]
    [ ?varList l_varList ]
    [ ?all g_all ]
)
=> t / nil
```

Description

Enables or disables corners for a run in the run plan or in the active setup of ADE Assembler.

Arguments

<code>?run t_run</code>	Name of the run for which the corners are to be enabled or disabled. By default, this argument is set to " " and the status of corners is changed in the active setup of ADE Assembler.
<code>?enable g_enable</code>	A Boolean value that specifies the status to be set for the specified corners.

`t`

The valid values are `t` and `nil`.

<code>?session t_session</code>	Name of the session. If not specified, the current session is used.

<code>?varList l_varList</code>	A list of the names of corners to be enabled or disabled. This argument selects or deselects the check box corresponding to the corners mentioned in the list.

By default, this argument is set to `nil`. If not specified, the function selects or deselects the *Corners* check box for the run or in the Data View assistant depending on the value of `?run`.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?all g_all`

A Boolean value that specifies whether all the corners in the setup for a run are to be enabled or disabled. This argument works only if the *Corners* check box is already enabled for the run.

- `t`: Enables all the corners in the setup.
- `nil`: Disables all the corners in the setup.

Value Returned

`t`

The status of the corners is set.

`nil`

The status of the corners is not set.

Examples

Selects the *Corners* check box for the run `Run.0`:

```
maeEnableCorner(?run "Run.0")  
=> t
```

Selects all the corners for the run `Run.0` if the *Corners* check box is already selected in the run setup:

```
maeEnableCorner(?run "Run.0" ?all t)  
=> t
```

Enables two corners `C0` and `C1` for `Run.0`:

```
maeEnableCorner(?run "Run.0" ?varList '("C0" "C1") ?enable t)  
=>t
```

Disables two corners `C0` and `C1` for `Run.0`:

```
maeEnableCorner(?run "Run.0" ?varList '("C0" "C1") ?enable nil)  
=>t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeEnableGlobalVar

```
maeEnableGlobalVar(  
    [ ?run t_run ]  
    [ ?enable g_enable ]  
    [ ?session t_session ]  
    [ ?varList l_varList ]  
    [ ?all g_all ]  
)  
=> t / nil
```

Description

Enables or disables the global variables for the specified run in the run plan or in the active setup of ADE Assembler.

Arguments

<code>?run t_run</code>	Name of the run for which the global variables are to be enabled or disabled. By default, this argument is set to " " and the status of global variables is changed in the active setup of ADE Assembler.
<code>?enable g_enable</code>	A Boolean value that specifies the status to be set for the <i>Global Variables</i> check box or the given list of variables. ■ <code>t</code>: Selects the <i>Global Variables</i> check box or the given list of variables. This is the default. ■ <code>nil</code>: Deselects the <i>Global Variables</i> check box or the given list of variables.
<code>?session t_session</code>	Name of the session. If not specified, the current session is used.
<code>?varList l_varList</code>	A list of the names of variables to be enabled or disabled in the <i>Design Variables</i> or <i>Global Variables</i> section. This argument selects or deselects the check box corresponding to the variables mentioned in the list. By default, this argument is set to <code>nil</code>. If not specified, the function selects or deselects the <i>Global Variables</i> check box for the run or in the Data View assistant depending on the value of <code>?run</code>.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?all g_all`

A Boolean value that specifies whether all the variables added to the setup for a run are to be enabled or disabled. This argument works only if the *Global Variables* check box is already enabled for the run.

- `t`: Enables all the variables added to the setup.
- `nil`: Disables all the variables added to the setup.

Value Returned

<code>t</code>	The status of the specified variables is changed.
<code>nil</code>	The status of the variables is not changed.

Examples

Selects the *Global Variables* check box for the run `Run.0`:

```
maeEnableGlobalVar(?run "Run.0")  
=> t
```

Enables the `cload` variable in the setup for the run `Run.0` if the *Global Variables* check box is already selected:

```
maeEnableGlobalVar(?run "Run.0" ?varList '("cload"))  
=> t
```

Enables all the variables in the setup for the run `Run.0` if the *Global Variables* check box is already selected:

```
maeEnableGlobalVar(?run "Run.0" ?all t)  
=> t
```

Disables `cload` in the setup for the run `Run.0`:

```
maeEnableGlobalVar(?run "Run.0" ?varList '("cload") ?enable nil)  
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeEnableParameter

```
maeEnableParameter(  
    [ ?run t_run ]  
    [ ?enable g_enable ]  
    [ ?session t_session ]  
    [ ?varList l_varList ]  
    [ ?all g_all ]  
)  
=> t / nil
```

Description

Enables or disables all of the given parameters for the given run in the run plan or in the active setup of ADE Assembler.

Arguments

<code>?run t_run</code>	Name of the run for which the parameters are to be enabled or disabled. By default, this argument is set to " " and the status of parameters is changed in the active setup of ADE Assembler.
<code>?enable g_enable</code>	A Boolean value that specifies the status to be set for the <i>Parameters</i> check box or the given list of parameters. ■ <code>t</code>: Selects the <i>Parameters</i> check box or the given list of parameters. This is the default. ■ <code>nil</code>: Deselects the <i>Parameters</i> check box or the given list of parameters.
<code>?session t_session</code>	Name of the session. If not specified, the current session is used.
<code>?varList l_varList</code>	A list of the names of parameters to be enabled or disabled. This argument selects or deselects the check box corresponding to the parameters mentioned in the list. By default, this argument is set to <code>nil</code>. If not specified, the function selects or deselects the <i>Parameters</i> check box for the run or in the Data View assistant depending on the value of <code>?run</code>.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?all g_all`

A Boolean value that specifies whether all the parameters added to the setup for a run are to be enabled or disabled. This argument works only if the *Parameters* check box is already enabled for the run.

- `t`: Enables all the parameters added to the setup.
- `nil`: Disables all the parameters added to the setup. This is the default.

Value Returned

`t`

The status of the specified parameters is changed.

`nil`

The status of the variables is not changed.

Examples

Selects the *Parameters* check box for the run `Run.0`:

```
maeEnableParameter(?run "Run.0")  
=> t
```

Enables the `bertlink/osc13/schematic/C2/c` parameter in the setup for the run `Run.0` if the *Parameters* check box is already selected:

```
maeEnableParameter(?run "Run.0" ?varList '("bertlink/osc13/schematic/C2/c"))  
=> t
```

Enables all the parameters in the setup for the run `Run.0` if the *Parameters* check box is already selected:

```
maeEnableParameter(?run "Run.0" ?all t)  
=> t
```

Disables the `bertlink/osc13/schematic/C2/c` parameter in the setup for the run `Run.0`:

```
maeEnableParameter(?run "Run.0" ?varList '("bertlink/osc13/schematic/C2/c")  
?enable nil)  
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeEnableTests

```
maeEnableTests(  
  [ ?run t_run ]  
  [ ?testNames l_testNames ]  
  [ ?enable g_enable ]  
  [ ?session t_session ]  
  [ ?all g_all ]  
)  
=> t / nil
```

Description

Enables or disables the specified tests for the specified run in the run plan or in the active setup of ADE Assembler.

Arguments

<code>?run <i>t_run</i></code>	Name of run for which tests are to be enabled or disabled. By default, this argument is set to " " and the status of tests is changed in the active setup of ADE Assembler.
<code>?testName <i>l_testNames</i></code>	List of the names of tests to be enabled or disabled. By default, this argument is set to <code>nil</code> .
<code>?enable <i>g_enable</i></code>	A Boolean value that specifies the status to be set for the specified tests in the given run. ■ <code>t</code>: Enables the specified tests. This is the default. ■ <code>nil</code>: Disables the specified tests.
<code>?session <i>t_session</i></code>	Name of the session. If not specified, the current session is used.
<code>?all <i>g_all</i></code>	Enables or disables all tests in the given run. ■ <code>t</code>: Enables all tests in the given run. This is the default. ■ <code>nil</code>: Disables all tests in the given run.

Value Returned

<code>t</code>	Enables or disables the specified tests.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`nil` No change is made to the status of the tests.

Examples

Enables tests `testA` and `testB` in the setup of run `Run.0`:

```
maeEnableTests(?run "Run.0" ?testNames '("testA" "testB"))  
=> t
```

Enables all tests in the setup of run `Run.0`:

```
maeEnableTests(?run "Run.0" ?all t)  
=> t
```

Disables tests `testA` and `testB` in the setup of run `Run.0`:

```
;maeEnableTests(?run "Run.0" ?testNames '("testA" "testB") ?enable nil)  
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeExportCornersToCSV

```
maeExportCornersToCSV(  
    t_sessionName  
    t_CSVFilePath  
)  
=> t / nil
```

Description

Exports corners from a maestro cellview to the given file in CSV format.

Arguments

<i>t_sessionName</i>	Name of the session. If not specified, the corners are exported from the current session.
<i>t_CSVFilePath</i>	Path of the CSV file to which the corners are to be exported. If no relative or absolute path is specified, the file is created in the current working directory.

Value Returned

<i>t</i>	The corners are exported to the CSV file.
<i>nil</i>	The corners are not exported.

Examples

Exports the corners from the current maestro session to the file `corners1.csv`:

```
maeGetSessions()  
=> ("fnxSession0")  
maeCornersToCSV("fnxSession0" "corners1.csv")  
=> t
```

maeExportCornersToSDB

```
maeExportCornersToSDB(  
    t_sessionName  
    t_SDBFilePath  
)  
=> t / nil
```

Description

Exports corners from a maestro cellview to the given file in SDB format.

Arguments

<i>t_sessionName</i>	Name of the session. If not specified, the corners are exported from the current session.
<i>t_SDBFilePath</i>	Path of the file to which the corners are to be exported. If no relative or absolute path is specified, the file is created in the current working directory. The <code>.sdb</code> extension is automatically appended to the name of the file.

Value Returned

<i>t</i>	The corners are exported to the SDB file.
<i>nil</i>	The corners are not exported.

Examples

Exports the corners from the current maestro session to the file `corners1.sdb`:

```
maeGetSessions()  
=> ("fnxSession0")  
maeCornersToCSV("fnxSession0" "corners1")  
=> t
```


maeExportGlobalVariablesToCSV

```
maeExportGlobalVariablesToCSV(  
    t_sessionName  
    t_CSVFilePath  
)  
=> t / nil
```

Description

Exports global variables from Virtuoso ADE Assembler to the given CSV file.

Arguments

<i>t_sessionName</i>	Name of the session. If not specified, the variables are exported from the current session.
<i>t_CSVFilePath</i>	Name of the CSV file to which the variables are to be exported.

Value Returned

t	The global variables are exported to the CSV file.
nil	The global variables are not exported.

Examples

Exports the global variables from the current maestro session to the file `globalvar.csv`:

```
maeGetSessions()  
=> ("fnxSession0")  
maeExportGlobalVariablesToCSV("fnxSession0" "globalvar.csv")  
=> t
```

maeExportRunPreview

```
maeExportRunPreview(  
    ?session t_sessionName  
    ?fileName t_fileName  
)  
=> t / nil
```

Description

Exports the Run Preview table to the specified CSV file. The default location of this file is *path-to-cellview/documents*.

Arguments

<i>t_sessionName</i>	The name of the session from which the run preview table is to be exported. If not specified, the current session is used.
<i>t_fileName</i>	Path to the CSV file to which the run preview table is to be exported. By default, the tool saves the table in <code>runPreviewTable.csv</code> in the <code>libName/cellName/viewName/documents</code> directory. If you specify the file name without a path, it is saved in the <code>libName/cellName/viewName/documents</code> directory.

Value Returned

<i>t</i>	The run preview table is exported to the specified CSV file.
<i>nil</i>	The current run mode is not supported for run preview.

Examples

Exports the run preview table to the specified CSV file.

```
maeExportRunPreview(?session maeGetDefaultSession() ?fileName "check.csv")  
=> t
```

maeExportSetupForExplorer

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

Description

Displays the Save A Copy form in ADE Explorer, which you can use to save the current setup. It is a callback function.

Arguments

<i>t_session</i>	The current ADE Explorer session.

Value Returned

<i>t</i>	The Save A Copy form is displayed.
<i>nil</i>	The Save A Copy form is not displayed.

Examples

Displays the Save A Copy form in the currently active ADE Explorer session.

```
maeExportSetupForExplorer(session)  
=> t
```

maeGetAllExplorerHistoryNames

```
maeGetAllExplorerHistoryNames(  
    t_session  
)  
=> l_historyNames / nil
```

Description

Returns a list of names of all the ADE Explorer histories for the specified maestro session.

Arguments

<i>t_session</i>	Name of the maestro session for which the ADE Explorer history names are to be listed.

Value Returned

<i>l_historyNames</i>	A list of names of ADE Explorer histories is displayed for the specified maestro session.
<i>nil</i>	The specified maestro session name is invalid.

Examples

Returns a list of names of all the ADE Explorer histories for the session `fnxSession0`.

```
maeGetAllExplorerHistoryNames("fnxSession0")  
=> list("ExplorerRun.0" "ExplorerRun.1")
```

maeGetAnalysis

```
maeGetAnalysis(  
    t_testName  
    t_analysis  
    [ ?includeEmpty g_includeEmpty ]  
    [ ?option t_option ]  
    [ ?session t_sessionName ]  
)  
=> l_options / nil
```

Description

Returns a list containing name-value pairs of the options set for the given analysis. By default, the function returns only those options for which the value is not empty. If the *g_includeEmpty* argument is set to *t*, it returns all the options. You can also specify the name of a specific option for which you want to know the value.

Arguments

<i>t_testName</i>	Name of the test.
<i>t_analysis</i>	Name of the analysis for which you want to return the settings.
<i>?includeEmpty g_includeEmpty</i>	Includes or excludes the options for which no value is set.
<i>?option t_option</i>	Name of the specific option for which you want to return the value.
<i>?session t_sessionName</i>	Name of the session.

Value Returned

<i>l_options</i>	List of options set for the given analysis.
<i>nil</i>	The list of options is not returned.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Examples

Returns the list containing name-value pairs of the options set for AC analysis for the given maestro session.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeGetAnalysis( "solutions:ampTest:2" "ac" )
=> (("anaName" "ac") ("sweep" "Frequency") ("rangeType" "Start-Stop") ("start" "1")
("stop" "10G" ("incrType" "Logarithmic") ("stepTypeLin" "Step Size")
("stepTypeLog" "Points Per Decade") ("dec" "20") ("outType" "Voltage") ("srcType"
"isource") ("perturbation" "linear") ("special" "None") ("save" "selected")
("oppoint" "no") ("annotate" "no") )
```

maeGetCornerEditingOptions

```
maeGetCornerEditingOptions(  
    [ ?session g_sessionName ]  
)  
=> l_dpl / nil
```

Description

Returns a disembodied property list containing the arguments specified through `maeSetCornerEditingOptions`.

Arguments

`?session g_sessionName`

Name of the session.

If not specified, the list of arguments given through `maeSetCornerEditingOptions` for the current session are displayed.

Value Returned

`l_dpl`

Returns the disembodied property list containing the arguments specified through `maeSetCornerEditingOptions`.

`nil`

The specified session name is invalid.

Examples

Returns the disembodied property list containing the arguments specified through `maeSetCornerEditingOptions`.

```
maeGetCornerEditingOptions()  
=> (nil cornerNameFilter "SLOW FAST" removeVarList ("var1") removeDuplicates t  
removeUndefined t)
```

Related Topics

[maeApplyCornerEditingOptions](#)

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

maeResetCornerEditingOptions

maeSetCornerEditingOptions

maeGetCurrentRunMode

```
maeGetCurrentRunMode(  
    [ ?session t_sessionName ]  
    [ ?run t_runName ]  
    [ ?Abbreviations g_abbreviation ]  
)  
=> t_runModeName / nil
```

Description

Returns the name of the run mode set in the given session or the given run.

Arguments

?session t_sessionName

Name of an ADE Assembler session.

If not specified, the currently active session is considered.

?run t_runName

Name of the run from the run plan for which you need to get the run mode.

If not specified, the run mode of the session is returned.

?Abbreviations g_abbreviation

Specifies whether the abbreviated form of the run mode name must be returned.

For example, for the "Single Run, Sweeps, and Corners" run mode, "SRSC" is returned.

Value Returned

t_runModeName

Name of the run mode set in the given session or the given run.

nil

The name of the run mode is not returned.

Examples

Returns the run mode for the current session.

```
maeGetCurrentRunMode()  
=> "Single Run, Sweeps and Corners"
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Returns the run mode for a specific run in the current session.

```
maeGetCurrentRunMode(?run "Run.1")  
=> "Monte Carlo"
```

maeGetEnabledAnalysis

```
maeGetEnabledAnalysis(  
    t_testName  
    [ ?session t_sessionName ]  
)  
=> l_analysisNames / nil
```

Description

Returns a list containing names of all the analyses enabled for the given test.

Arguments

<i>t_testName</i>	Name of the test.
<i>?session t_sessionName</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>l_analysisNames</i>	List of the names of analyses enabled for the given test.
<i>nil</i>	The list of analyses names is not returned.

Examples

Returns the list of enabled analyses for a test.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
maeGetEnabledAnalysis( "solutions:ampTest:2")  
=> ("ac" "tran")
```

maeGetEnvOption

```
maeGetEnvOption(  
    t_testName  
    [ ?includeEmpty g_includeEmpty ]  
    [ ?option t_option ]  
    [ ?session t_sessionName ]  
)  
=> l_optionList / nil
```

Description

Returns a list of name-value pairs for all the environment option for the given test. By default, it returns all the environment options whose value is not `nil`. If the `includeEmpty` argument is set to `t`, the function returns all the environment options. If a specific option is given, it returns the value of only that option.

Arguments

<code>t_testName</code>	Name of the test.
<code>?includeEmpty l_includeEmpty</code>	Includes or excludes the options for which no value is set.
<code>?option t_option</code>	Name of the option for which the value is to be returned.
<code>?session t_sessionName</code>	Name of the session If not specified, the current session is used.

Value Returned

<code>l_optionList</code>	A list of name-value pairs of all or the specified environment options is returned.
<code>nil</code>	Indicates an error.

Examples

Returns a list of name-value pairs for all the environment options in the test
`solutions:ampTest:2`.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeGetEnvOption("solutions:ampTest:2")
=> (switchViewList ("spectre" "cmos_sch" "cmos.sch" "schematic" "veriloga" "ahdl")
stopViewList ("spectre")
controlMode "interactive"
modelFiles ("/home/user1/models/Models/allModels.scs" "FF"))
....
)
```

Returns a list of name-value pairs for all the environment options in the test
`solutions:ampTest:2` except for the options with no value.

```
maeGetEnvOption("solutions:ampTest:2" ?includeEmpty t)
=> ( switchViewList ("spectre" "cmos_sch" "cmos.sch" "schematic" "veriloga" "ahdl")
stopViewList ("spectre")
controlMode "interactive"
modelFiles ("/home/user1/models/Models/allModels.scs" "FF"))
paramRangeCheckFile ""
analysisOrder nil
stimulusFile nil
.....)
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Returns the name-value pair for the specified environment option `switchViewList`.

```
maeGetEnvOption("solutions:ampTest:2" ?option "switchViewList")
=> ("spectre" "cmos_sch" "cmos.sch" "schematic" "veriloga" "ahdl")
```

Returns the list of DSPF files used for the test `mytestname`.

```
dspf = maeGetEnvOption("mytestname" ?option "dspfFile")
(printf "dspf = %L\n" dspf)
```

Returns the complete list of options including the options that are currently not set for all enabled tests in ADE Assembler.

```
tests = maeGetSetup(?typeName "tests" ?enabled t)

(foreach test tests
  (printf "test = %s\n" test)
  dspf = maeGetEnvOption(test ?option "dspfFile")
  (printf "dspf = %L\n" dspf)
)
```

maeGetExplorerTestName

```
maeGetExplorerTestName(  
    [ ?session t_sessionName ]  
)  
=> t_testName / nil
```

Description

Returns the name of the test opened in ADE Explorer. If you descend into ADE Explorer from ADE Assembler, use this function to get the name of the current test.

Arguments

?session t_sessionName

Name of an ADE Explorer session.

If not specified, the currently active session is used.

Value Returned

t_testName

Name of the currently opened test in ADE Explorer.

nil

When the function is used for ADE Assembler.

Examples

Returns the name of the test opened in ADE Explorer.

```
maeGetExplorerTestName()  
=> "AC"
```

Specifies the session to be used if multiple sessions of ADE Explorer are open.

```
maeGetSessions()  
=> ("fnxSession0" "fnxSession1")  
testName = maeGetExplorerTestName(?session "fnxSession0")  
=> "AC"  
maeGetTestSession(testName ?session "fnxSession0")  
=> stdobj@0x2e48fde8
```

Returns the variable details for a test in ADE Explorer.

```
maeGetSessions()  
=> ("fnxSession0" "fnxSession1")  
testName = maeGetExplorerTestName(?session "fnxSession0")  
=> "AC"
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
testId = axlGetTest(sdb testName)
;; Find the total sweeps in variables
varList = axlGetVars(testId)
foreach(var cadr(varList)
  varId = axlGetVar(testId var)
  when(axlGetEnabled(varId)
    ;; Get the total number of sweep points in the desired variable
    if(rexMatchp(":" axlGetVarValue(varId)) then
      sweepList=mapcar('evalstring parseString(axlGetVarValue(varId) ":")
      varSweepVal= int((car(last(sweepList))-car(sweepList))/cadr(sweepList))
    else
      varSweepVal = length(parseString(axlGetVarValue(varId)))
    )
    sweepVal=sweepVal*varSweepVal
  );when
);foreach varList
```


maeGetSetup

```
maeGetSetup(  
  [ ?typeName t_typeName ]  
  [ ?enabled g_enabled ]  
  [ ?run t_runName ]  
  [ ?session t_sessionName ]  
)  
=> l_setupDetails / nil
```

Description

Returns the required setup details of variables, parameters, corners, and tests from the given session.

Arguments

`?typeName t_typeName`

Type of data for which the details are to be returned.

The possible values are:

- tests (default)
- corners
- variables
- parameters

`?enabled g_enabled`

Status of variables to be returned.

- t: Returns the enabled variables.
- nil: Returns the disabled variables.
- 'all (default): Returns both enabled and disabled variables.

`?run t_runName`

Name of a run in the run plan for which you want to get the setup details.

`?session t_sessionName`

Name of the session.

If not specified, the currently active session is used.

Value Returned

<i>l_setupDetails</i>	List containing the names of variables, parameters, corners, and tests.
<i>nil</i>	The setup details are not returned.

Examples

Returns the names of all the tests in the current session:

```
maeGetSetup()
```

Returns the names of all the enabled tests in the current session.

```
maeGetSetup(?enabled t)
=> ("AC" "TRAN")
```

Returns the names of all the enabled corners in the current session.

```
maeGetSetup(?typeName "corners" ?enabled t)
=> ("Nominal" "c2")
```

Returns the names of all the enabled variables.

```
maeGetSetup(?typeName "variables" ?enabled t)
=> ("gain" "vcm" "vdd")
```

Returns the list of variables from the given run in the run plan.

```
maeGetSetup(?typeName "variables" ?run "Run.0")
=> ("AVDD")
```

maeGetSimOption

```
maeGetSimOption(  
    t_testName  
    [ ?option t_optionName ]  
    [ ?includeEmpty g_includeEmpty ]  
    [ ?session t_sessionName ]  
)  
=> l_options / nil
```

Description

Returns a list containing name-value pairs for the simulator options for the given test. By default, it returns all the simulator options whose value is not `nil`. If the *includeEmpty* argument is set to `t`, it returns all the simulator options. If a specific option is specified, the function returns the value of only that option.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

<code>t_testName</code>	Name of the test.
<code>?option t_optionName</code>	Name of the simulator option for which the value is to be returned.
<code>?includeEmpty g_includeEmpty</code>	Specifies whether the options with no given value are to be returned or not. ■ <code>t</code>: Returns the options with no specified value. ■ <code>nil</code>: Does not return the options with no specified value. This is the default.
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>l_options</code>	List of name-value pairs of simulator options
<code>nil</code>	The simulator options are not returned.

Examples

Returns the values of all the options for test `solutions:ampTest`:

```
maeGetSimOption("solutions:ampTest:2" ?includeEmpty t)
=> (label1: "" reltol: "1e-3" residualtol: "" vabstol: "1e-6" iabstol: "1e-12" )
```

maeGetSessions

```
maeGetSessions()  
=> l_sessionNames / nil
```

Description

Returns an alphanumerically sorted list of ADE Explorer or ADE Assembler sessions that are currently open.

You can use the session names returned by this function in other functions, such as `maeSaveSetup` or `maeGetSetup`.

Arguments

None

Value Returned

<code>l_sessionNames</code>	List of alphanumerically sorted ADE Explorer or ADE Assembler sessions that are currently open.
<code>nil</code>	The list of sessions is not returned.

Examples

Returns the list of valid ADE Explorer and ADE Assembler sessions that are currently active.

```
maeGetSessions()  
=> ("fnxSession0" "fnxSession1" "fnxSession2" "fnxSession3" "fnxSession4"  
"fnxSession5" "fnxSession6" "fnxSession7" "fnxSession8" "fnxSession9"  
"fnxSession10" "fnxSession11")
```

maeGetSetupLibrary

```
maeGetSetupLibrary(  
    [ ?session t_session ]  
)  
=> t_cellviewName / nil
```

Description

Retrieves the setup library view from the current or specified session.

Arguments

`?session t_session` Name of the ADE Assembler session. If you do not specify this argument, the current session is used.
The default is " ".

Value Returned

<code>t_cellviewName</code>	The <i>library/cell/view</i> that contains the setup library cellview that is bound to the specified Assembler session.
<code>nil</code>	The ADE Assembler session does not contain a setup library cellview.

Examples

The following example shows how to retrieve the setup library cellview name from the specified Assembler session.

Retrieves the setup library from the currently open maestro cellview.

```
maeGetSetupLibrary()  
=> "amsPLL/Project45nm/setuplib"
```

Retrieves the setup library from the specified Assembler session.

```
sess = maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro")  
maeGetSetupLibrary(?session sess)  
=> "setupLibrary/projectSetup/setupLib"
```

maeGetTags

```
maeGetTags (
    [ ?session t_sessionName ]
    [ ?elementType l_elementType ]
)
=> l_tagsList / nil
```

Description

Returns a list of tags assigned to the specified corners or variables in the given session.

Arguments

?session t_sessionName

The name of the session in which the tags are to be queried. If not specified, the current session is used.

?elementType l_elementType

The type of element for which the tags are to be queried.

- **GlobalVars:** Lists the tags assigned to global variables.
- **TestVars:** Lists the tags assigned to the design variables.
- **Corners:** Lists the tags assigned to corners.
- **All:** Lists the tags assigned to global variables, design variables, and corners.

Value Returned

l_tagsList

List of tags assigned to the specified variables.

nil

The specified variables do not have any tags assigned to them.

Examples

Returns a list of tags assigned to the global variables.

```
maeGetTags(?elementType '(GlobalVars))
=> (nil GlobalVars (nil supply ("vdd") calcVal ("cct_freq") load ("cload" "rload"
"res1"))
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
)  
)
```

Returns the list of tags assigned to the design variables.

- The tag `supply` is assigned to the design variable `vdd`.
- The tag `load` is assigned to the design variables `cload`, `rload`, and `vdd`.

```
maeGetTags(?elementType '(TestVars))  
=> (nil TestVars (nil supply ("vdd") load ("cload" "rload" "vdd")))
```

Returns the list of tags assigned to corners.

```
maeGetTags(?elementType '(Corners))  
=> (nil Corners (nil projectA ("C0_0" "C0_1")))
```

The tag `projectA` is assigned to the corners `C0_0` and `C0_1`.

Returns the list of tags assigned to global variables, design variables, and, corners.

- The tag `supply` is assigned to the global and design variable `vdd`.
- The tag `load` is assigned to the design variables `cload`, `rload`, and `vdd`.
- The tag `calcVal` is assigned to the global variable `cct_freq`.

```
maeGetTags(?elementType '(All))  
=> (nil Corners (nil) TestVars (nil supply ("vdd") load ("cload" "rload" "vdd"))  
GlobalVars (nil supply ("vdd") calcVal ("cct_freq")  
load ("cload" "rload" "res1")  
)  
)
```

Related Topics

[maeDeleteTags](#)

[maeSetTags](#)

maeGetTestEnvVar

```
maeGetTestEnvVar(  
    t_testName  
    t_varName  
    [ ?session t_sessionName ]  
)  
=> g_value / nil
```

Description

Returns the value of the specified environment variable for the given test. This function can be used to get the value set for a particular test using the [maeSetTestEnvVar](#) function.

Arguments

<code>t_testName</code>	Name of the test.
<code>t_varName</code>	Name of the environment variable for which the value is to be returned.
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>g_value</code>	The value of the environment variable set for the given test is returned.
<code>nil</code>	The value for the set environment variable is not returned.

Examples

Returns the values set for the *PinCheck Term Mismatch Action* and *PinCheck Term Direction Mismatch Action* environment options for the tests in ADE Assembler.

```
;; Loading setup  
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro11" ?mode "a")  
=> "fnxSession0"  
testnames=maeGetSetup()  
> ("AC" "TRAN")  
maeGetTestEnvVar("AC" "termDirectionMismatch" ?session "fnxSession0")  
>"warning"
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeGetTestEnvVar("TRAN" "termMismatch" )  
=>"ignore"
```

Related Topics

[maeSetTestEnvVar](#)

maeGetTestSession

```
maeGetTestSession (
    t_testName
    [ ?session t_sessionName ]
)
=> h_testSession / nil
```

Description

Returns session handle to the given test. You can use this handle to view or modify the test details.

Arguments

<code>t_testName</code>	Name of the test.
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>h_testSession</code>	The session handle to the test. The handle returned by this function can be used as a session argument to SKILL functions prefixed with <code>asi</code> , as shown in the example.
<code>nil</code>	Indicates an error.

Examples

```
;; Loading setup
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestroll" ?mode "a")
=> "session0"
testnames=maeGetSetup()
> ("AC" "TRAN")
>
testHandle=maeGetTestSession(car(testnames))
=>stdobj@0x21989b60
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
asiAddModelLibSelection(testHandle "../models/spectre/gpdk090.scs" "NN")
=> t
>
maeRunSimulation()
=> "Interactive.1"
maeWaitUntilDone('All)
=> t
maeGetSimulationMessages(?msgType "ERROR")

maeExportOutputView()
=> "/servers/scratch02/testcases/adexl_workshop/design/Interactive.1.csv"
exit()
```

maeGetVar

```
maeGetVar(  
    t_varname  
    [ ?typeName t_typeName ]  
    [ ?typeValue t_typeValue ]  
    [ ?session t_sessionName ]  
)  
=> t_value / nil
```

Description

Returns value of the given variable. Use this function to get the global, test-level, or corner value for the given variable.

Arguments

t_varname Name of variable.

?typeName t_typeName

Type of the variable.

- **test**: Specifies that the variable value is to be returned for the given test. This is the default.
- **corner**: Specifies that the variable value is to be returned for the given corner.

?typeValue t_typeValue

Name of the corner or test for which the variable value has to be returned.

- **Global**: Returns the global value of the given variable. This is the default.

If the value for this argument is set to **Global** when the type name is set to **corner**, an error is displayed.

- **TestName**: Returns the variable value for the specified test.
- **CornerName**: Returns the variable value for the specified corner.

?session t_sessionName

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Name of the session.

If not specified, the current session is used.

`?expanded g_expandedValue`

If the given variable is assigned a swept value or a range, expands and returns a list of values.

The default value is `nil`.

`?resolved g_resolvedValue`

If the value of the given variable is an expression, resolves the expression and returns its resulting value.

The default value is `nil`.

Value Returned

`t_value`

Value of the given variable.

`nil`

When the given corner, test, or variable is not found in the setup.

Examples

Returns the value of global variable `vdd` in a cellview:

```
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro" ?mode "a")
=> "session0"
```

```
maeGetVar("vdd")
=> 2.2
```

```
maeGetVar("vdd" ?typeName "test" ?typeValue "Global")
=> "2.2"
```

```
maeGetVar("vdd" ?typeName "test")
=> "2.2"
```

Expands the swept values for a variable and returns a list.

```
Vdd = 1:2:10
```

```
maeGetVar("vdd" ?expanded t)
=> ("1" "3" "5" "7" "9")
```

Returns a list of values extracted from the given range for a variable.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
Vdd = {From/To}Linear:50u:5u:100u{From/To}
maeGetVar("vdd" ?expanded t)
=> ("50u" "55u" "60u" "65u" "70u" "75u" "80u" "85u" "90u" "95u" "100u")
```

The scalars, sweeps, and VAR returns the resolved results. The function recognizes the type of the variables (global, test design variable, local override) and returns the corresponding values.

Returns local overridden value of nn in test AC.

```
nn = VAR("mm")
maeGetVar("nn" ?typeName "test" ?typeValue "AC" ?resolved t)
=> "1"
```

Returns global value of nn in the setup.

```
maeGetVar("nn" ?typeName "test" ?typeValue "Global" ?resolved t)
=> "2"
```

Resolves the value assigned through a variable.

```
nn = VAR("vdd")
vdd = 1.3 1.4 1.5
maeGetVar("nn" ?resolved t) = "1.3 1.4 1.5"
nn = VAR("myExpr")
myExpr = if(VAR("myNum") == 20 then 1 else 0)
maeGetVar("nn" ?resolved t) = "if(VAR(\"myNum\") == 20 then 1 else 0)"
```

Resolves the basic arithmetic operations in variable values if the related expressions do not depend on design points.

```
nn = VAR("mm")+2
mm = 8
maeGetVar("nn" ?resolved t)
=> "10"
```

Returns original expressions if they depend on the design points generated during simulation.

```
mm = VAR("nn")+2
nn = 8 9 10
maeGetVar("mm" ?resolved t)
=> "VAR(\"nn\") + 2"
mm = calcVal(Voffset, \"AC\")
maeGetVar("mm" ?resolved t)
=> "calcVal(Voffset, \"AC\")"
```

maeImportOutputsFromFileInDocuments

```
maeImportOutputsFromFileInDocuments(  
    t_filename  
    [ ?operation t_operation ]  
    [ ?test t_test ]  
    [ ?session t_session ]  
  
)  
=> t / nil
```

Description

Imports outputs from the specified CSV file in the `documents` directory to one or all tests in an already opened maestro cellview. The usage of this function is similar to [axlOutputsImportFromFile](#) that can import outputs from a CSV file from any location.

Arguments

<i>t_filename</i>	The name of the CSV file from which outputs are to be imported.
<i>?operation t_operation</i>	Specifies how to use the imported outputs. Valid values are: ■ overwrite: Overwrites the existing set of outputs that are already defined with the same name as that of an output being imported. This is the default value. ■ retain: Retains the existing set of outputs when the name of an existing output matches with that of an output being imported. ■ merge: Keeps the existing set of outputs and merges them with the outputs imported from the specified CSV file.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?test t_test` Name of the test for which the imported outputs are to be used.

Specify this argument only when the test name is not saved in the CSV file being imported. If the CSV file contains the *Test* column, each output is imported for the specified test.

If you wish to use the imported outputs for all the tests, set the value for this argument to "Import to All Tests".

Default value: " "

`?session t_session` Name of the maestro session.

Value Returned

`t` The outputs are imported from the specified CSV file.

`nil` Indicates an error.

Examples

The following example opens a maestro cellview in edit mode and imports all outputs from the specified CSV file into the `documents` directory.

Opens maestro cellview in edit mode

```
sessionName = maeOpenSetup("Two_Stage_Opamp" "OpAmp" "maestro_nominal_base")
"fnxSession0"
```

Retrieves the files from the `documents` directory

```
getDirFiles(strcat(axlGetSetupDBDir(axlGetMainSetupDB(sessionName))
"/documents"))
(".." "variant6.il" "variant1.il" "." "variant2.il" "maestro_child.csv"
"variant3.il" "variant5.il" "variant1.csv" "maestro_child.il" "variant4.il"
)
```

Imports the outputs from the CSV file in the `documents` directory

```
maeImportOutputsFromFileInDocuments("maestro_child.csv")
=> t
```

maeImportSetupForExplorer

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

Description

Displays the Import Setup form in ADE Explorer, which you can use to import a saved `maestro` cellview in the current setup. This is a callback function.

Arguments

<code>t_session</code>	The current ADE Explorer session.

Value Returned

<code>t</code>	The Import Setup form is displayed.
<code>nil</code>	The Import Setup form is not displayed.

Examples

Displays the Import Setup form for the currently active ADE Explorer session.

```
maeImportSetupForExplorer(session)
```

maelsPrerunScriptEnabled

```
maelsPrerunScriptEnabled(  
    t_sessionName  
    t_testName  
)  
=> t / nil
```

Description

Checks whether pre-run scripts are enabled for a test in the given maestro session.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>t_testName</i>	Name of the test.

Value Returned

<i>t</i>	If pre-run script is enabled for the specified test.
<i>nil</i>	If pre-run script is not enabled in the test, if session is not found, or the specified test is invalid.

Examples

Checks whether pre-run scripts are enabled for the test `test1`.

```
maelsPrerunScriptEnabled("fnxSession0" "test1")  
=> t
```

maelsSetupModified

```
maelsSetupModified(  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

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

Arguments

<i>t_sessionName</i>	Name of the session.
	By default, the function uses the current maestro session.

Value Returned

<i>t</i>	Indicates that the setup of the specified maestro session was modified after the last time it was saved.
<i>nil</i>	Indicates that the setup of the specified maestro session has not been modified after the last time it was saved.

Examples

Checks whether the setup of the specified maestro session has been modified since the value for the variable *Cap* is set to 2p.

```
sess= maeOpenSetup("solution" "ampTest" "maestro")  
maeSetVar("Cap" "2p")  
maelsSetupModified(?session sess)  
=> t
```

maelsSingleTest

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

Description

Returns *t* if the given cellview contains a single test or multiple tests.

Arguments

<i>t_libName</i>	Name of a library.
<i>t_cellName</i>	Name of a cell in the given library.
<i>t_viewName</i>	Name of a maestro cellview.

Value Returned

<i>t</i>	The specified maestro view contains a single test. This implies that it can be opened in ADE Explorer.
<i>nil</i>	The specified maestro view contains multiple tests. This implies that it can be opened in ADE Assembler.

Examples

Specifies that the given maestro view contains a single test and can be opened in ADE Explorer.

```
maeIsSingleTest("solutions" "ampTest" "maestro")  
=> t
```

maelsValidMaestroSession

```
maelsValidMaestroSession(  
    t_sessionName  
)  
=> t / nil
```

Description

Confirms if the given session is valid for ADE Explorer or ADE Assembler.

Arguments

<i>t_sessionName</i>	Session name to be validated.

Value Returned

<i>t</i>	The specified session is a valid ADE Explorer or ADE Assembler session.
<i>nil</i>	The specified session is not valid.

Examples

Specifies that the session `fnxSession0` is valid.

```
maelsValidMaestroSession("fnxSession0")  
=> t
```

maeLoadCorners

```
maeLoadCorners(  
    t_fileName  
    [?operation t_operation]  
    [?session t_session]  
)  
=> t /nil
```

Description

Loads corners from the given file into the corner setup of the current session or the specified session.

Arguments

t_fileName Name of the .sdb or .csv file from which the corners are to be loaded.

?operation t_operation

Operation to be performed while loading the corners.

- "deleteallncreatenew": Deletes all existing corners and loads new corners.
- "overwrite": If the name of the corner being loaded is the same as that of an existing corner, the existing corner is removed and the new one is added. This is the default.
- "rename": If the name of the corner being loaded is the same as that of an existing corner, retains the existing corner and renames the new corner being loaded.
- "cancel": If the name of the corner being loaded is the same as that of an existing corner, cancels the loading process. No corner is loaded.

?session t_session Name of the session.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The corners are successfully loaded from the source file.
<code>nil</code>	The corners are not loaded.

Examples

Loads corners from two files, `corners.sdb` and `ffCorners.csv`. If the corner names in the two files are conflicting, they are renamed.

```
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro11" ?mode "a")
=> "session0"

maeLoadCorners("corners.sdb" ?operation "overwrite")
=> t
maeLoadCorners("ffCorners.csv" ?operation "rename")
=> t
```


maeMakeEditable

```
maeMakeEditable(  
    [?session t_session]  
)  
=> t / nil
```

Description

Changes the mode of an already open cellview from read-only to editable.

Arguments

?session t_session Name of the session.

Value Returned

<i>t</i>	The corners are successfully loaded from the source file.
<i>nil</i>	The corners are not loaded.

Examples

Changes the mode for the session `fnxSession1` from read-only to editable.

```
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro11" ?mode "r")  
=> "fnxSession1"  
  
maeMakeEditable( ?session "fnxSession1")  
=> t
```

maeMakeReadonly

```
maeMakeReadonly(  
    [?session t_session]  
)  
=> t / nil
```

Description

Changes the mode of an already open cellview from editable to read-only.

Arguments

?session t_session Name of the session.

Value Returned

<i>t</i>	The corners are successfully loaded from the source file.
<i>nil</i>	The corners are not loaded.

Examples

Changes the mode of the session `fnxSession1` from editable to read-only.

```
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro11" ?mode "a")  
=> "fnxSession1"  
  
maeMakeReadonly( ?session "fnxSession1")  
=> t
```

maeOpenSetup

```
maeOpenSetup (
    t_libName
    t_cellName
    t_viewName
    [ ?application t_applicationName ]
    [ ?histName t_historyName ]
    [ ?mode t_mode ]
)
=> t_sessionName / nil
```

Description

Loads the given cellview and restores the setup details from the specified history. If no history name is specified, the active setup is loaded. If the specified view is already opened in the current Virtuoso session, it is not opened again. However, if the view is already open in some other Virtuoso session, it is opened in read mode in the current session. If the given cellview does not exist, the function creates a new cellview with the same name.

If you make changes after opening a setup, it is recommended to use the `maeSaveSetup` function to save those changes.

Arguments

`t_libName` Name of the library.

`t_cellName` Name of the cell.

`t_viewName` Name of the view.

`?application t_applicationName`

Specifies whether to open the setup in ADE Assembler.

Set this argument to `Assembler` when you need to open a single test environment in ADE Assembler to use the features specific to this product.

Note: This argument is not required to open a test in ADE Explorer.

`?histName t_historyName`

Name of the history to be restored. If not given, the active setup is loaded.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?mode t_mode` Mode in which the maestro view is to be opened

- `a`: Opens the maestro view in append mode. This is the default.
- `r`: Opens the maestro view in read mode.

Note: You cannot create a new view in read mode. If you are creating a new view, set this argument to "a".

Value Returned

`t_sessionName` The setup is successfully loaded

`nil` The setup is not loaded.

Examples

Loads the active setup of `solutions/amptest/maestro` in append mode.

```
maeOpenSetup("solutions" "amptest" "maestro")
```

Loads the active setup of `solutions/amptest/maestro` in append mode in ADE Assembler.

```
maeOpenSetup("solutions" "amptest" "maestro" ?application "Assembler")
```

Loads the setup of the history `Interactive.5` for `solutions/amptest/maestro` in append mode.

```
maeOpenSetup("solutions" "amptest" "maestro" ?histName "Interactive.5")
```

Loads the setup of `Interactive.5` history for `solutions/amptest/maestro` in read-only mode.

```
maeOpenSetup("solutions" "amptest" "maestro" ?histname "Interactive.5" ?mode "r")
```

Related Topics

[maeSaveSetup](#)

maeLoadSetupState

```
maeLoadSetupState(  
    t_stateName  
    [ ?run t_runNames ]  
    [ ?tags l_tagNames ]  
    [ ?operation s_operationName ]  
    [ ?session t_sessionName ] )  
=> t_sessionName / nil
```

Description

Loads the given setup state into the given session.

Arguments

<i>t_stateName</i>	Name of the state to be loaded.
<i>?run t_runName</i>	Name of the run in the Run Plan to which the saved state is loaded.
<i>?tags l_tagNames</i>	List of tags that defines the components to be copied from the loaded state. ■ <i>tests</i>: Copies the testbench setups. ■ <i>vars</i>: Copies the global variables. ■ <i>parameters</i>: Copies the parameters and their values. ■ <i>currentMode</i>: Copies the run mode. ■ <i>runOptions</i>: Copies the simulation options for different run modes and the run distribute options. ■ <i>specs</i>: Copies the parameter specifications. ■ <i>corners</i>: Copies the corner details. ■ <i>modelGroups</i>: Copies the model groups. ■ <i>extensions</i>: Copies the extensions. ■ <i>relxanalysis</i>: Copies the reliability analysis setup details. ■ <i>All</i>: Copies all the details. This is the default.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?operation s_operationName`

Operation to handle the existing details in the setup.

- `'merge` - If the existing setup details have the same name as that of the details being loaded, they are deleted and overwritten by the setup information copied from the loaded state.
- `'retain` - If an existing setup detail has the same name as that of the detail being loaded, it is retained and not overwritten by the setup information copied from the loaded state.
- `'overwrite` - All existing setup details are deleted and all the imported outputs are used

`?session t_sessionName`

Name of the session to which the state has to be loaded.

If not specified, the current session is used.

Value Returned

<code>t</code>	Successfully loaded the given setup state.
<code>nil</code>	The setup state is not loaded.

Examples

Loads the corners from state `state1` and merges those with the current corner details:

```
maeLoadSetupState("state1" ?tags list("corners") ?operation 'merge)
=> t
```

Loads variables and parameters from the given state and runs a simulation:

```
; state name to load
statestate = "GlobalOpt.2.PointID.590"

; (optional) backup current vars and params to a setup
(maeSaveSetupState state_backup ?tags '("vars" "parameters"))

(maeLoadSetupState state ?operation 'merge)

(maeRunSimulation ?runMode "Single Run, Sweeps and Corners")
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Loads the corners from state `state1` and merges those with the current corner details in run `Run.1`:

```
maeLoadSetupState("state1" ?tags list("corners") ?run "Run.1" ?operation 'merge)  
=> t
```

maeLoadStateForTest

```
maeLoadStateForTest(  
    t_testName  
    t_stateName  
    [ ?session t_session ]  
    [ ?loadFrom s_loadFrom ]  
    [ ?statePath s_statePath ]  
    [ ?libName t_libName ]  
    [ ?cellName t_cellName ]  
    [ ?simulator t_simulatorName ]  
    [ ?component l_componentList ]  
)  
=> t_sessionName / nil
```

Description

Loads the saved ADE state for the given test.

Arguments

<i>t_testName</i>	Name of the test in the given maestro setup for which the state is to be loaded.
<i>t_stateName</i>	Name of the state to be loaded for the given test.

?session t_session

Name of the session to which the ADE state is to be loaded.

If not specified, the current session is used.

?loadFrom s_loadFrom

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Specifies whether the state is to be loaded from cellview or directory.

- `'directory`: Specifies that the state must be loaded from the given directory.
- `'cellview`: Specifies that the state must be loaded from the given cellview.

If no value is specified for this argument, the `saveAsCellview` environment variable is checked. If the variable is set to `t`, state is loaded from the cellview. Otherwise, the state is loaded from the directory.

`?statePath t_statePath`

State path from where state is to be loaded when `?loadFrom` is set to `'directory`.

It specifies the path where `libName/cellName/simulator/stateName` is located. The default path is taken from the `saveDir` environment variable.

`?libName t_libName`

Library name for which state is to be loaded. If not specified, the library name from the test session is taken.

`?cellName t_cellName`

Cell name for which state is to be loaded. If not specified, the given cell name from test session is taken.

`?simulator t_simulatorName`

Simulator name for which state is to be loaded. If not specified, the simulator name from the test session is taken.

`?component l_componentList`

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

List of the components to be loaded from the state. When no value is given, all components are loaded.

Valid Values:

- analyses
- variables
- outputs
- subckts
- opPoints for
- modelSetup
- simulationFiles
- environmentOptions
- simulatorOptions
- convergence
- waveformSetup_ws
- graphicalStimuli
- conditionsSetup
- printSetup (Results Display Setup)
- devCheckingSetup
- relxOptions
- cosimOptions
- turboOptions (Performance/Parasitic Reduction)
- mdlOptions
- dpSetup (Distributed Processing)
- paramSetup
- all (default)

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t_sessionName</code>	The given state is loaded.
<code>nil</code>	The state is not loaded.

Examples

Loads the test `test1` to the ADE state `spectre_state1`.

```
maeLoadStateForTest("test1" "spectre_state1" ?loadFrom 'cellView ?statePath  
"/home/TEST/TRAINING/training/ampTest/adex1/test_states" ?libName "training"  
?cellName "ampTest" ?simulator "spectre" ?component '(environmentOptions  
simulationFiles))  
=> t
```

Related Topics

[saveAsCellview](#)

[saveDir](#)

maeResetCornerEditingOptions

```
maeResetCornerEditingOptions(  
    [ ?session g_sessionName ]  
)  
=> t / nil
```

Description

Resets the Corners Setup form to its state before `maeSetCornerEditingOptions` was applied.

Arguments

`?session g_sessionName`

Name of the session.

If not specified, the Corners Setup form for the current session is reset.

Value Returned

`t` Resets the Corners Setup form to its state before `maeSetCornerEditingOptions` was specified.

`nil` The specified session name is invalid.

Examples

Resets the Corners Setup form to its state before `maeSetCornerEditingOptions` was specified.

```
maeResetCornerEditingOptions()  
=> t
```

Related Topics

[maeApplyCornerEditingOptions](#)

[maeSetCornerEditingOptions](#)

maeRegTimer

```
maeRegTimer(  
    t_callbackString  
    x_tenthsOfseconds  
)  
=> t / nil
```

Description

Registers or sets a timer to initialize the specified SKILL function after the given time has elapsed during simulation. This timer can be used in ADE interface job customization.

Arguments

<i>t_callbackString</i>	Name of the maestro session to be initialized.
<i>x_tenthsOfseconds</i>	Time, in tenths of seconds, after which the given SKILL function is run.

Value Returned

t	The timer is set.
nil	The timer is not set or the function is not found.

Examples

Sets a timer for the maeStopJob function.

```
maeRegTimer("maeStopJob(3)" 150)  
=> t
```

Related Topics

[maeStopJob](#)

maeRunDspfChecker

```
maeRunDspfChecker(  
    [ ?session t_sessionName ]  
)  
=> l_testName
```

Description

Runs the Spf Checker utility for the tests that have DSPF files in the setup.

You can view the results of this utility in the *DSPF* section of the EMIR Analysis Setup form.

Arguments

`?session t_sessionName`

Name of the maestro session where the Spf Checker is to be run.

Value Returned

`l_testName`

List of the name of the tests that are enabled in the maestro cellview.

Note: Even though the function returns the names of all the enabled tests, the Spf Checker utility is run only for those tests that have DSPF files in the setup.

Examples

Runs the spf checker for the maestro session `fnxSession0`.

```
maeRunDspfChecker("fnxSession0")  
=> ("bertlink_osc13_1")
```

maeSaveSetup

```
maeSaveSetup(  
    [ ?lib t_libName ]  
    [ ?cell t_cellName ]  
    [ ?view t_viewName ]  
    [ ?session t_sessionName ] )  
=> t / nil
```

Description

Saves the setup database file and test state files for the current session in the library, cell, view format. The behavior of this function is similar to *File - Save*. It is recommended to run this function after making changes in the setup.

By default, the setup of the current cellview is saved. If the cellview was opened in the read-only mode, a new library, cell, and view value must be provided for which you need to save the setup details.

Note: If a cellview is managed by a Design Management (DM) system, ensure that it is in *CheckedOut* state before using this function.

Arguments

<code>?lib t_libName</code>	Name of the library.
<code>?cell t_cellName</code>	Name of the cell.
<code>?view t_viewName</code>	Name of the view.
<code>?session t_sessionName</code>	Name of the session.
	If not specified, the current session is used.

Value Returned

<code>t</code>	The setup is saved.
<code>nil</code>	The setup is not saved.

Examples

Saves the current setup.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeSaveSetup()  
=> t
```

Saves the current setup as `solutions/amp/test/maestro/maestro.sdb`.

```
maeSaveSetup(?lib "solutions" ?cell "amp/test" ?view "maestro")  
=> t
```


maeSaveSetupStateForOptimization

```
maeSaveSetupStateForOptimization(  
    [ ?session t_sessionName ]  
    [ ?run t_runName ]  
    [ ?setupState t_setupState ]  
    [ ?pointID x_pointID ]  
    [ ?historyName t_historyName ]  
)  
=> t / nil
```

Description

Saves the setup state for optimization from the best design point of a run or the given history. You can use this saved state as the starting point for an optimization run in a run plan.

Arguments

?session t_sessionName

Name of the session.

If not specified, the current session is used.

?run t_runName

Name of run from which the best point is to be saved in the setup state. If you do not specify a run, the function looks for the *?historyName* argument. If both are not specified, the function returns an error.

The default value is " ".

?setupState t_setupState

Name of setup state to be saved. If you do not specify any value, ADE Assembler saves the setup state with the name of the given run, the current history, or the given point ID.

The default value is " ".

?pointID x_pointID

ID of the point from which the setup details are to be saved in the setup state. If you do not specify any point ID, ADE Assembler saves the setup state by using the details from the best point of the given history or the given run.

The default value is `null`.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?historyName t_historyName`

Name of the history from which the setup state for the best point is to be used. If you do not specify any history, the tool uses the details from the current history.

The default value is " ".

Value Returned

<code>t</code>	The setup state is saved.
<code>nil</code>	The setup state is not saved.

Example

Saves a setup state named `newState` for point 7 in history `Plan.0` of the run named `Run_opt1`:

```
maeSaveSetupStateForOptimization(?session maeGetDefaultSession() ?run "Run_opt1"
?setupState "newState" ?pointID 7 ?historyName "Plan.0")
=> t
```

maeSetCornerEditingOptions

```
maeSetCornerEditingOptions(  
    [ ?cornerNameFilter g_FilterName ]  
    [ ?session g_sessionName ]  
    [ ?removeVarList g_VarList ]  
    [ ?removeDuplicates {t|nil}]  
    [ ?removeUndefined {t|nil}]  
  
    )  
=> t / nil
```

Description

Specifies the options for editing the corners setup. These settings are reflected in the Corners Setup form, when the options *Import Corners from CSV File* and *Import Corners from SDB File* are used.

You can also use the function `maeApplyCornerEditingOptions` to reflect the specified changes when the Corners Setup form is opened.

Arguments

`?cornerNameFilter g_FilterName`

The names of the corners to be filtered.

`?session g_sessionName`

Name of the session.

`?removeVarList g_VarList`

The names of the variables to be removed from the Corners Setup form.

`?removeDuplicates g_removeDuplicates`

Removes duplicate corners.

`?removeUndefined g_removeUndefined`

Removes the corners with undefined variables.

Value Returned

`t` Specifies the editing options for the Corners Setup form.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`nil` The specified session name is invalid.

Examples

Setting these functions updates the Corners Setup form as follows:

- The corners `SLOW` and `FAST` are filtered.
- The variable `var1` is removed.
- The duplicate corners are removed.
- All undefined variables in the corners setup are removed.

```
maeSetCornerEditingOptions(?cornerNameFilter "SLOW FAST" ?removeVarList  
list("var1") ?removeDuplicates t ?removeUndefined t )  
=> t  
  
maeApplyCornerEditingOptions()  
=> t
```

maeSetStartingPointForOptimization

```
maeSetStartingPointForOptimization(  
    [ ?session t_sessionName ]  
    [ ?run t_runName ]  
    [ ?setupState t_setupState ]  
    [ ?runMode t_runMode ]  
)  
=> t / nil
```

Description

Sets the given setup state as the starting point for the Local Optimization or Global Optimization run mode in the specified run.

Arguments

`?session t_sessionName`

Name of the session.

If not specified, the current session is used.

`?run t_runName`

Name of run on which starting point is to be set from the given setup state. If you do not specify this argument, the starting point is set in the run options for the active setup.

The default value is " ".

`?setupState t_setupState`

Name of setup state that is set as the starting point in the settings of the optimization run. If you do not specify this argument, ADE Assembler sets the starting point as Automatic.

The default value is " ".

`?runMode t_runMode`

Specifies an optimization run mode to be used in the given run or the active setup when ?run is not specified.

- Local Optimization: Sets the given setup state as the starting point for the Local Optimization run mode.
- Global Optimization: Sets the given setup state as the starting point for the Global Optimization run mode.

The default value is " ".

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The given setup state is set as the starting point for given run or the active setup.
<code>nil</code>	The given setup state is not set as the starting point.

Examples

Sets `newState` as the starting point in the options of the `Local Optimization` run mode for the `Run_opt2` run in the run plan:

```
maeSetStartingPointForOptimization(?session maeGetDefaultSession() ?run
"Run_opt2" ?setupState "newState" ?runMode "Local Optimization")
```

maeSetTags

```
maeSetTags (  
    [ ?session t_sessionName ]  
    [ ?variables l_variables ]  
    [ ?corners l_corners ]  
    [ ?tags l_tags ]  
    [ ?overwrite g_overwrite ]  
)  
=> t / nil
```

Description

Assigns the given tags to the specified corners and variables. You can also append to or overwrite the existing tags.

Arguments

?session t_sessionName

Name of the session in which the variable tags are to be queried. If not specified, the current session is used.

?variables l_variables

List of global or design variables to which you want to assign the specified tags.

?corners l_corners List of corners to which you want to assign the specified tags.

?tags l_tags List of strings containing the tags to be assigned to the specified variables or corners.

?overwrite g_overwrite

Overwrites the existing tags assigned to a variable or corner. If not specified, the list of tags is appended to the existing tags.

Value Returned

t The given tags are assigned to the specified variables or corners.

nil The specified tags do not exist in the given session.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Examples

Assigns the tags `projectA` and `projectB` to the variables `vdd` and `vin`, and the corners `C0` and `C1`, by overwriting the existing ones.

```
maeSetTags(?variables '("vdd" "vin") ?corners '("C0" "C1") ?tags '("projectA"
"projectB") ?overwrite t)
=> t
```


maeStmConsolidateStimuli

```
maeStmConsolidateStimuli(  
    t_libName  
    t_cellName  
    t_viewName  
    ?reportFile t_reportFile  
)  
=> t / nil
```

Description

Removes duplicate stimuli definitions from the given cellview. For this, ADE Explorer or ADE Assembler reviews each test in the cellview and, if required, reassigns pins and globals by removing duplicates.

Note: Open the Stimuli Assignment form from a maestro cellview before using this function.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>?reportFile t_reportFile</i>	Prints a report of changes in the specified report file.

Value Returned

<i>t</i>	Duplicate stimuli definitions are removed from the given cellview.
<i>nil</i>	The duplicate stimuli definitions are not removed.

Examples

Removes duplicate stimuli definitions from the given cellview when specified for consolidated stimuli in the open maestro cellview.

```
maeStmConsolidateStimuli("testLib" "StimuliTest" "maestro" ?reportFile  
"consolidationReport.txt")  
=> t
```

maeStmGenerateWaveforms

```
maeStmGenerateWaveforms(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?test t_testName ]  
    [ ?reportFile t_reportFile ]  
    [ ?force g_force ]  
)  
=> t / nil
```

Description

Generates preview stimuli waveforms for the stimuli added for a maestro cellview that contains stimuli definitions added through the Stimuli Assignment form. These stimuli definitions are saved in the file

lib-name/cell-name/view-name/namedStimuli/stimuli.xml.

The function examines each of the stimuli definitions for preview waveforms. If a stimuli definition has no associated preview waveform, or the waveform data is out of date, it generates a new SRR waveform data and saves it in the directory

project-dir/lib-name/cell-name/view-name/namedStimuli.

Note: Open the Stimuli Assignment form from a maestro cellview before using this function.

The waveform data is considered out-of-date if the time stamp of a stimuli XML definition is newer than its associated preview waveform data.

To view the waveforms generated by this function, open the Stimuli Assignment form, right-click on a stimuli and select the *Plot Transient Waveform*.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>?test t_testName</i>	Name of the test to which the stimuli is associated. If no test name is specified, the tool uses the first test in the cellview.
<i>?reportFile t_reportFile</i>	

Name of the report file in which the status message for each stimuli waveform is printed.

When no file name is provided, the default file path relative to the current working directory is used.

The default value is

```
"/maeStmGenerateWaveforms.lib.cell.view.test.log".
```

`?force g_force`

Forces waveform generation for each stimuli.

By default, the tool generates waveforms only when the stimuli information does not match the current waveforms, which could happen in the following scenarios:

- No waveform exists.
- Values of variables have been changed after waveform generation.
- Value of a stimuli parameter has been changed.

Value Returned

`t`

Preview stimuli waveforms are generated.

`nil`

Preview stimuli waveforms are not generated.

Examples

Generates stimuli waveforms.

```
maeStmGenerateWaveforms("testLib" "StimuliTest" "maestro")  
=> t
```

Generates stimuli waveforms for all stimuli and prints the report in the given file.

```
maeStmGenerateWaveforms("testLib" "StimuliTest" "maestro" ?force t ?reportFile  
"/reports/stimuliReport.log")  
=> t
```

Related Topics

Setting Up Stimuli

maeSaveSetupState

```
maeSaveSetupState(  
    t_stateName  
    [ ?tags l_tagNames ]  
    [ ?inReadOnly s_readOnlyAction ]  
    [ ?session t_sessionName ] )  
=> t / nil
```

Description

Saves a setup state for the given session.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

<code>t_stateName</code>	State in which you need to save the setup details.
<code>?tags l_tagNames</code>	List of tag names that specify the details to be saved. Available tags are: ■ <code>tests</code>: Testbench setups ■ <code>vars</code>: Global variables ■ <code>parameters</code>: Parameters and their values ■ <code>currentMode</code>: Run mode ■ <code>runOptions</code>: Simulation options for different run modes and the run distribute options ■ <code>specs</code>: Parameter specifications ■ <code>corners</code>: Corner details ■ <code>modelGroups</code>: Model groups ■ <code>extensions</code>: Extensions ■ <code>relxanalysis</code>: Reliability analysis setup details ■ <code>All</code>: Details of all the tests, variables, parameters, current run mode, run options, specs, corners, model groups, extensions, and reliability analysis.
<code>?inReadOnly s_readOnlyAction</code>	Specifies the action to be performed in read only mode. Possible values: ■ <code>'error</code>: Displays an error. This is the default value. ■ <code>'useSaveDir</code>: Saves the setup state in the save directory. ■ <code>'useprojectDir</code>: Saves the setup state in the project directory.
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The setup details are saved.
<code>nil</code>	The setup details are not saved.

Examples

Saves the corners and variables from the current session to a state named `state1`.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeSaveSetupState("state1" '("corners" "vars"))
=> t
```

maeSetAnalysis

```
maeSetAnalysis(  
    t_testName  
    t_analysis  
    [ ?enable g_enabled ]  
    [ ?options l_options ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds or removes an analysis for the given test. The function can also be used to set the value of different options for the analysis.

Arguments

<i>t_testName</i>	Name of the test to which you need to add an analysis.
<i>t_analysis</i>	Name of the analysis.
<i>?enable g_enabled</i>	Status to be set. ■ <i>t</i>: Enables the specified analysis for the given test. This is the default. ■ <i>nil</i>: Disables the specified analysis for the given test.
<i>?options l_options</i>	List of analyses options and the values to be set. Note: You can use maeGetAnalysis to get the list of analysis options, as shown below. <pre>maeGetAnalysis(t_testName t_analysisName ?includeEmpty t)</pre>
<i>?session t_sessionName</i>	Name of the session. If not specified, the current session is used.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The specified analysis is added or removed for the given test.
<code>nil</code>	The analyses setup for the given test is not updated.

Examples

Adds a new analysis `tran` and sets its options, stop time 500 and step 10.:

```
maeSetAnalysis( "solutions:ampTest:2" "tran" ?enable t ?options '("stop" "500")
("step" "10"))
=> t
maeSaveSetup()
```

Related Topics

[maeGetAnalysis](#)

maeSetCorner

```
maeSetCorner(  
    t_cornerName  
    [ ?enabled g_enabled ]  
    [ ?enableTests l_enableTests ]  
    [ ?disableTests l_disableTests ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a new corner and enables/disables it for the given test.

Arguments

<i>t_cornerName</i>	Name of the corner to be added or updated.
<i>?enabled g_enabled</i>	Specifies whether the given corner is to be enabled or disabled. This option updates the status of the check box for this corner in the Run Summary assistant.
<i>?enableTests l_enableTests</i>	List of the tests for which the given corner has to be enabled. The default value is 'ALL'.
<i>?disableTests l_disableTests</i>	List of the tests for which the given corner has to be disabled. The default value is 'ALL'.
<i>?session t_sessionName</i>	Name of the session. If not specified, the currently active session is used.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The given corner is added or updated.
<code>nil</code>	No corners are added or updated.

Examples

Adds a corner C0 and enables it for all tests.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeSetCorner("C0")
maeSaveSetup()
```

Adds a corner C1 and enables it for tests AC and TRAN, but disables the corner in the setup. This means that the check box for this corner is deselected in the Run Summary assistant.

```
maeSetCorner("C1" ?enableTests `("AC", "TRAN") ?enabled nil)
maeSaveSetup()
```

Adds a corner C2 and disables it for test1 and test3.

```
maeSetCorner("C2" ?disableTests '("test1" "test3"))
maeSaveSetup()
```

maeSetCurrentRunMode

```
maeSetCurrentRunMode(  
    [ ?session t_sessionName ]  
    ?runMode t_runModeName  
)  
=> t / nil
```

Description

Updates the current run mode in ADE Assembler.

Arguments

?session *t_sessionName*

Name of the session.

If not specified, the currently active session is used.

?runMode *t_runModeName*

Name of the corner to be added or updated.

Value Returned

t The run mode is changed.

nil Indicates an error.

Examples

Sets Fault Simulation as the current run mode.

```
maeSetCurrentRunMode(?runMode "Fault Simulation")  
=> t
```

maeSetDesign

```
maeSetDesign(  
    t_testName  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?session t_sessionName ]  
)  
=> t_testName / nil
```

Description

Sets a design for the given test. Use this function to change the design associated to a test.

Arguments

<i>t_testName</i>	Name of the test for which you need to change the design.
<i>t_libName</i>	Name of the library of the design to be used.
<i>t_cellName</i>	Name of the cell of the design to be used.
<i>t_viewName</i>	Name of the maestro cellview of the design to be used.
<i>?session t_sessionName</i>	Name of the session. If not specified, the currently active session is used.

Value Returned

<i>t_testName</i>	Returns the name of the test for which the design is changed.
<i>nil</i>	Indicates an error.

Examples

Sets the design for test `test1` to the `config` view of the `ampTest` cell in library `solutions`:

```
maeSetDesign("test1" "solutions" "ampTest" config)  
=>"test1"
```

maeSetDesignForTest

```
maeSetDesignForTest(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?test t_test ]  
    [ ?session t_session ]  
  
    )  
=> t / nil
```

Description

Sets a design for all tests or the specified test.

Arguments

<i>t_libName</i>	The name of the library for the specified design.
<i>t_cellName</i>	The name of the cell for the specified design.
<i>t_viewName</i>	The name of the view for the specified design.
<i>?test t_test</i>	The name of a single test or a list of tests. If no test is specified, the design is set to all tests in the session.
<i>?session t_session</i>	The name of the session. If no name is specified, the currently active session is used.

Value Returned

<i>t</i>	The command was successful.
<i>nil</i>	Indicates an error.

Examples

The following example shows how to change the design for a single test, all tests, or a list of specified tests.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Changes the design for all tests

```
maeSetDesignForTest("Two_Stage_Opamp" "OpAmp_TRAN_top" "config")  
=> t
```

Changes the design for a specified test

```
maeSetDesignForTest("Two_Stage_Opamp" "OpAmp_TRAN_top" "config2" ?test "AC")  
=> t
```

Changes the design for the list of tests

```
maeSetDesignForTest("Two_Stage_Opamp" "OpAmp_TRAN_top" "config2" ?test list("AC"  
"TRAN"))  
=> t
```

maeSetEnableTestVar

```
maeSetEnableTestVar(  
    t_testName  
    l_variableNames  
    [ ?enabled g_enabled ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Enables or disables the specified variables or local sweeps for the given test.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

<code>t_testName</code>	Name of the test.
<code>l_variableNames</code>	List of variable names to be enabled or disabled for the given test
<code>?enabled g_enabled</code>	Status to be set for the given variables. ■ <code>t</code>: Enables the specified variables or local sweeps for the given test. This is the default. ■ <code>nil</code>: Disables the specified variables or local sweeps for the given test.
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>t</code>	The specified variables or local sweeps are enabled or disabled for the given test.
<code>nil</code>	The specified variables or local sweeps are not enabled or disabled.

Examples

Enables `var1` and `var3` local variables for test `test1`.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeSetEnableTestVar("test1" '("var1" "var3") ?enabled t)
=> t
```


maeSetEnvOption

```
maeSetEnvOption(  
    t_testName  
    [ ?options l_options ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets values for one or more environment options for the given test.

Arguments

t_testName Name of the test

?options l_options List of environment options and the values to be set for them.
You can use [maeGetEnvOption](#) to get the list of environment options, as shown below.

```
maeGetEnvOption(t_testName ?includeEmpty t)
```

Note: This function does not validate the values specified for the environment options. Therefore, you must ensure that the names of options and their respective values are in correct format.

?session t_sessionName

Name of the session

If not specified, the current session is used.

Value Returned

t The values for the environment options are set successfully.

nil The values for the environment options are not set.

Examples

Sets values for multiple environment options.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeSetEnvOption("AC" ?options '("stopViewList" ("spectre")) ("modelFiles"  
  ("./models.scs" "FF"))))  
;; Sets the stopViewList as ("spectre") and modelFiles as ("./models.scs" "FF")
```

Sets multiple model files by specifying a list of model file paths.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
  
maeSetEnvOption("AC" ?options '("modelFiles" ("./models.scs" "FF")  
  ("./TECH/GPDK045/gpdk045/models/spectre/gpdk045.scs" "mc")  
  ("./DESIGNS/GPDK045/FACNPLL/models/inductor.scs"))))
```

Related Topics

[maeGetEnvOption](#)

maeSetHistoryLock

```
maeSetHistoryLock(  
    t_historyName  
    g_lock  
    [ ?session t_sessionName ] )  
=> t / nil
```

Description

Locks or unlocks the given history for the given ADE Assembler session.

Arguments

t_historyName Name of the history to be locked or unlocked

g_lock Specifies the lock status.

■ *t*: Locks the specified history.

■ *nil*: Unlocks the specified history.

?session t_sessionName

Name of the session.

If not specified, the current session is used.

Value Returned

t The setup is saved successfully.

nil The setup is not saved.

Examples

Locks the history `Interactive.1`.

```
maeSetHistoryLock("Interactive.1" t)  
=> t
```

maeSetParameter

```
maeSetParameter(  
    t_parameterName  
    g_parameterValue  
    [ ?typeName t_typeName ]  
    [ ?typeValue l_typeValue ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a new parameter at the global level or corner level. If the parameter already exists, its value is updated.

Arguments

<i>t_parameterName</i>	Name of the parameter to be added or updated. Parameter names must contain at least five non-empty names each separated by a / (slash), typically representing the library/cell/view/instance/property to be modified. Multiple "instance/" strings can be used to specify hierarchical parameters, such as library/cell/view/instance0/instance1/instance2/property.
<i>g_parameterValue</i>	Value to be set for the parameter.
<i>?typeName t_typeName</i>	Specifies if the parameter is to be added or updated at the global level or only for corners. ■ test: Adds the specified parameter at a global level. This is the default value. ■ corner: Adds the parameter only for the specified corner.
<i>?typeValue l_typeValue</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

List of the corners for which the parameter is to be added.

Note: If `?typeName` is set to `test`, value given for this argument is ignored. The parameter is added at the global level.

The default value is "Global".

`?session t_sessionName`

Name of the session.

If not specified, the current session is used.

Value Returned

`t` The specified parameter is added.

`nil` The specified parameter is not added.

Examples

Adds a parameter `R0/r` at the test level and sets its value as 8k.

```
sess= maeOpenSetup("Two_Stage_Opamp" "OpAmp" "maestro")
=> "session0"
maeSetParameter("Two_Stage_Opamp/OpAmp/schematic/R0/r" "8k")
=> t
```

Sets the value for the parameter `R0/r` to 6k for corners `C0` and `C1`.

```
maeSetParameter("Two_Stage_Opamp/OpAmp/schematic/R0/r" "6k" ?typeName "corner"
?typeValue '("C0" "C1"))
=> t
```

maeSetSpec

```
maeSetSpec (  
    t_outputName  
    t_testName  
    [ ?minimum g_minValue ]  
    [ ?maximum g_maxValue ]  
    [ ?gt g_greaterThanValue ]  
    [ ?lt g_lessThanValue ]  
    [ ?range g_rangeValues ]  
    [ ?tolerance g_toleranceValue ]  
    [ ?info g_info ]  
    [ ?weight g_weightingFactor ]  
    [ ?corner g_cornerName ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a specification to an output defined for a test. You can also use this function to modify an existing specification for an output.

Arguments

<i>t_outputName</i>	Name of the output for which the specification is to be set.
<i>t_testName</i>	Name of the test to which the output is associated.
<i>?minimum g_minValue</i>	Value for the min specification.
<i>?maximum g_maxValue</i>	Value for the max specification.
<i>?gt g_greaterThanValue</i>	Value for the greater than specification.
<i>?lt g_lessThanValue</i>	Value for the less than specification.
<i>?range g_rangeValues</i>	A range of values for the range specification.
<i>?tolerance g_toleranceValue</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value for the tolerance specification.

?info *g_info* An information string for the info specification.

?weight *g_weightingFactor*

A weighting factor for this specification.

?corner *g_cornerName*

Name of the corner for which the spec is to be enabled. This argument helps to override a specification for a particular corner.

By default, a specification defined for a measurement applies to all the corners enabled for the test. To change the specification for a particular corner, specify the name of that corner in this argument. In ADE Assembler, you can view the overridden corner name in the Override Specifications form.

Note: This argument is considered only for ADE Assembler sessions because ADE explorer does not support overridden specifications.

?session *t_sessionName*

Name of the session.

If not specified, the current session is used.

Value Returned

t The given specification is added to the specified output.

nil The given specification is not added.

Examples

Adds the >1 spec to output *out1* in test *TRAN*.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeSetSpec("out1" "TRAN" ?gt "1")
=> t
```

maeSetSetup

```
maeSetSetup(  
    [ ?tests l_testNames ]  
    [ ?variables l_variables ]  
    [ ?parameters l_params ]  
    [ ?corners l_corners ]  
    [ ?enabled g_enabled ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Enables or disables the tests, global variables, parameters, and corners.

Arguments

?tests l_testNames

List of the tests that are to be enabled or disabled.

?variables l_variables

List of the global variables to be enabled or disabled.

?parameters l_params

List of the parameters to be enabled or disabled.

?corners l_corners

List of the corners to be enabled or disabled.

To enable or disable the Nominal corner, add `Nominal` or `nominal` to the list.

Note: If all other corners are disabled, the function keeps the Nominal corner enabled.

?enabled g_enabled

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Specifies whether the given tests, global variables, parameters, or corners are to be enabled or disabled.

- `t`: Enables the given tests, global variables, parameters, or corners. This is the default value.
- `nil`: Disables the given tests, global variables, parameters, or corners.

`?session t_sessionName`

Name of the session.

If not specified, the current session is used.

Value Returned

<code>t</code>	The specified tests, global variables, parameters, or corners are enabled or disabled.
<code>nil</code>	Unsuccessful operation

Examples

Enables the specified tests `test1`, `test2`, and `test3`, corners `nominal` and `C1`, and variables `VDD` and `IREF`.

```
; load the setup
maeOpenSetup("solutions" "ampTest" "maestro") > t
maeSetSetup(?tests '("test1" "test3"))
; enables test1 and test3.
maeSetSetup(?tests '("test2") ?enabled nil)
; Disable test2
```

Disables the corners `nominal` and `C1` and the global variables `VDD` and `IREF`.

```
maeSetSetup(?corners '("nominal" "C1") ?variables '("VDD" "IREF")?enabled nil)
```

maeSetSetupLibrary

```
maeSetSetupLibrary(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Connects the provided setup library view to the maestro view.

Arguments

<code>t_libName</code>	String value specifying the setup library name.
<code>t_cellName</code>	String value specifying the cell name in the setup library.
<code>t_viewName</code>	String value specifying the view name in the setup library.
<code>?session t_sessionName</code>	String value that specifies the Unix session. If this value is not provided, the value of the active session is picked. The default is "".

Value Returned

<code>t</code>	The specified value in the setup database field <i>SetupLibrary</i> is updated.
<code>nil</code>	Indicates an error.

Examples

Connects the specified setup library view `setupLib` to the currently open maestro cellview.

```
maeSetSetupLibrary("Two_Stage_Opamp" "OpAmp" "setupLib")  
=> t
```

Connects the specified setup library view `setupLib` to the maestro cellview which is open in the session `fnxSession2`.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeSetSetupLibrary("Two_Stage_Opamp" "OpAmp" "setupLib" ?session "fnxSession2")  
=> t
```

maeSetSimOption

```
maeSetSimOption(  
    t_testName  
    [ ?options l_options ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets values for the specified simulator options of the given test. Multiple options can be specified in a single command.

Arguments

t_testName Name of the test.

?options l_options

List of name-value pairs of the simulator options to be set

Note: You can use [maeGetSimOption](#) to get the list of simulator options as shown below.

```
maeGetSimOption(t_testName ?includeEmpty t)
```

?session t_sessionName

Name of the session

If not specified, the current session is used.

Value Returned

t The specified values are set for the given simulator options.

nil The simulator option values are not set.

Examples

Sets the simulator option `temp` as 27 and `maxwarns` as 10 for test `solutions:ampTest:2`:

```
maeSetSimOption("solutions:ampTest:2" ?options '("temp" "27") ("maxwarns" "10"))  
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Related Topics

[maeGetSimOption](#)

maeSetTestEnvVar

```
maeSetTestEnvVar(  
    t_testName  
    t_varName  
    g_varValue  
    [ ?session t_sessionName ]  
)  
=> t_value / nil
```

Description

Sets the value for the given environment variable at the test level. The value is used only by the specified test. Other tests in the session use the value set at the global level or specific values set for them, if any.

Arguments

<code>t_testName</code>	Name of the test for which you need to set the value for an environment variable.
<code>t_varName</code>	Name of the environment variable.
<code>g_varValue</code>	Valid value for the environment variable.
<code>?session t_sessionName</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>t_value</code>	The value of the environment variable is set successfully for the given test.
<code>nil</code>	The value of the environment variable is not set for the given test. The function also returns an error if the given value is invalid.

Examples

Sets the values for *PinCheck Term Mismatch Action* and *PinCheck Term Direction Mismatch Action* in the Environment Options form for test AC in ADE Assembler:

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
; load the setup
sess=maeOpenSetup("opamp090" "full_diff_opamp_AC" "maestro11" ?mode "a")
=> "fnxSession0"
testnames=maeGetSetup()
> ("AC" "TRAN")

maeSetTestEnvVar( "AC" "termDirectionMismatch" "ignore" ?session "fnxSession0")
=>"ignore"
```

maeSetVar

```
maeSetVar(  
    t_varname  
    g_valValue  
    [ ?typeName t_typeName ]  
    [ ?typeValue l_typeValue ]  
    [ ?run t_runName ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a variable to the given test or corner in the given ADE Explorer or ADE Assembler session. If the variable already exists, its value is updated.

Arguments

t_varname Name of the variable.

g_valValue Value of the variable.

?typeName t_typeName

Type of the variable.

■ **test**: Adds the specified variable to the given test. This is the default value.

■ **corner**: Adds the specified variable to the given corner.

?typeValue t_typeValue

List of the corners or tests for which variable has to be added.

The default value is Global.

?run t_runName

Name of the run in run plan to which you want to add a variable.

?session t_sessionName

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Name of the ADE Explorer or ADE Assembler session where you want to make these changes.

If not specified, the current session is used.

Value Returned

t	The specified variable is added to the given test or corner.
nil	The specified variable is not added to the given test or corner.

Examples

Sets the value for the global variable, Cap to 5p.

```
maeSetVar("Cap" "5p")
```

Set value for variable Cap to 5p in test AC.

```
maeSetVar("Cap" "5p" ?typeName "test" ?typeValue '("AC"))
```

Set value for variable Cap to 5p in corners C1, C2, and C3:

```
maeSetVar("Cap" "5p" ?typeName "corner" ?typeValue '("C1" "C2" "C3"))
```

Returns the value of a variable from a previous run and update it for the next run.

```
Res=maeOpenResults(?run "vco_char")
kvco_val=maeGetOutputValue("kvo" "vco_char" ?cornerName "nominal")
maeCloseResults(Res)
maeSetVar("KVC0" kvco_val)
;; This example script can be helpful in run plans.
```

Related Topics

[maeCloseResults](#)

[maeGetOutputValue](#)

[maeOpenResults](#)

[maeSetVar](#)

maeUpdateImplicitSignals

```
maeUpdateImplicitSignals(  
    o_session  
    [?test t_name]  
)  
=> t / nil
```

Description

Updates the implicit signals for the tests in the specified maestro session.

Arguments

<i>o_session</i>	Maestro session object.
?test <i>t_name</i>	Name of the test for which the implicit signal outputs are to be updated.

Value Returned

<i>t</i>	The implicit signals are updated for the tests in the specified maestro session.
<i>nil</i>	The implicit signals are not updated.

Examples

Generates implicit signals for the expressions in the maestro setup, if the implicit signals do not already exist. It also clears the implicit signals which may have existed but are no longer used in any existing expression.

```
maeUpdateImplicitSignals( maeSession ?testName test1)  
=> t
```

Example script for Viewing and Editing Setup Details

The following example shows how to view and edit the setup details:

```
; load the setup
maeOpenSetup("solutions" "ampTest" "maestro") > t
; check which tests are there in setup
maeGetSetup()
=> ("solutions"ampTest:1" "solutions:ampTest:2")
; Enable a test "solution:ampTest:1"
maeSetSetup(?tests '("solutions:ampTest:1"))
; Set the value of global variable
"CAP" as 5p: maeSetVar("CAP" 5p)
; Disable corners C0 and C1
maeSetSetup(?corners '("C0" "C1") ?enabled nil)

; save changes in the setup
maeSaveSetup()
; Run Simulation
maeRunSimulation()
```

Functions to Work with Simulation Results

This topic lists the SKILL functions that let you view and save simulation results for maestro cellviews.

[getSimRunInfo](#)

[maeCloseResults](#)

[maeDeleteSimulationData](#)

[maeDisplayLogDiagnostics](#)

[maeExportOutputView](#)

[maeExportOutputView](#)

[maeGetNBestDesignPoints](#)

[maeGetOutputValue](#)

[maeGetParamConditions](#)

[maeGetResultOutputs](#)

[maeGetResultTests](#)

[maeGetResultTests](#)

[maeGetSpecStatus](#)

[maeGetOverallSpecStatus](#)

[maeGetOverallYield](#)

[maeGetOverallYield](#)

[maeOpenResults](#)

[maeReadResDB](#)

[maeRestoreHistory](#)

[maeRunLogDiagnostics](#)

[maeWriteDatasheet](#)

getSimRunInfo

```
getSimRunInfo(  
    t_Type  
)  
=> s_Value/ nil
```

Description

Returns the value of the specified type of simulation information for a given ADE output expression by accessing the related `psf` directory.

Note the following:

- The specified job control mode must be LSCS.
- This function must be set in the *Outputs Setup* tab.
- The return value can be viewed in the *Results* tab.

Argument

<code>t_Type</code>	The type of the simulation information to be retrieved. The possible values for this argument are <code>memory</code> , <code>threads</code> , <code>nodes</code> , <code>host</code> , <code>jobId</code> and <code>startTime</code> .

Value Returned

<code>s_Value</code>	The value of the specified type of simulation information for a given ADE expression.
<code>nil</code>	If the job control mode is set to ICRP or the specified type of simulation information is invalid.

Examples

Returns the value of the specified type of simulation information for a given ADE output expression.

maeCloseResults

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

Description

Closes the results opened by the `maeOpenResults` function.

Arguments

None

Value Returned

<code>t</code>	The results opened by the <code>maeOpenResults</code> function are closed.
<code>nil</code>	The results opened by the <code>maeOpenResults</code> function are not closed.

Examples

Closes the result `Interactive.1` opened by the `maeOpenResults` function.

```
; Load the results for that history  
maeOpenResults(?history "Interactive.1")  
=> t  
; Get the output value  
maeGetOutputValue("outputA" "solutions:amptest:1")  
=> 2.2  
;close the results  
maeCloseResults()  
=> t
```

Related Topics

[maeOpenResults](#)

maeDeleteSimulationData

```
maeDeleteSimulationData(  
    t_historyName  
    ?session t_sessionName  
    ?keepNetlist g_keepNetlist  
    ?keepQuickPlot g_keepQuickPlot  
)  
=> t / nil
```

Description

Deletes the simulation results data for the given history.

Arguments

t_historyName Name of the history for which you need to delete the simulation data.

?session t_sessionName
 Name of the session.

?keepNetlist g_keepNetlist
 Boolean value that specifies whether to retain the netlist even if simulation data is deleted.

?keepQuickPlot g_keepQuickPlot
 Boolean value that specifies whether to retain the quick plot data even if simulation data is deleted.

Value Returned

t The simulation results data is deleted for the given history.

nil The simulation results data is not deleted.

Examples

Deletes simulation data or netlist and quick plot for the history *Interactive.0*.

```
maeDeleteSimulationData("Interactive.0")  
=> t
```


Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Deletes simulation data while keeping the netlist folder and spectre logs for the history Interactive.0.

```
maeDeleteSimulationData("Interactive.0 ?keepNetlist t)  
=> t
```

maeDisplayLogDiagnostics

```
maeDisplayLogDiagnostics(  
    x_ID  
)  
=> t_fileName / nil
```

Description

Displays the ADE diagnostics report for the diagnostics run associated with the specified ID. The diagnostics report is generated in the background without freezing the maestro session.

However, when this function is called, the maestro session freezes until the report is available. This report is then displayed in the web browser specified by the following environment variable.

```
envSetVal("ui" "webBrowser" 'string "browserName")
```

Arguments

<i>x_ID</i>	ID of the diagnostics run for which the ADE diagnostics report is to be generated.

Value Returned

<i>t_fileName</i>	Name of the JSON file where the generated diagnostics report is saved.
<i>nil</i>	Indicates an error.

Examples

Returns the name of the JSON file where the generated diagnostics report is saved.

```
report = maeRunLogDiagnostics(?lib "Two_Stage_Opamp" ?cell "OpAmp" ?view "maestro"  
?historyName "Interactive.1")  
maeDisplayLogDiagnostics(report)  
=> Interactive.1_2023-04-21_13-35-58.json
```

Related Topics

[Debugging with ADE Diagnostics](#)

[maeRunLogDiagnostics](#)

maeExportOutputView

```
maeExportOutputView(  
    [ ?session t_sessionName ]  
    [ ?fileName t_fileName ]  
    [ ?view t_viewType ]  
    [ ?historyName t_historyName ]  
    [ ?testName t_testName ]  
    [ ?filterName t_filterName ]  
    [ ?clearAllFilters g_clearAllFilters ]  
)  
=> t / nil
```

Description

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

Arguments

`?session t_sessionName`

Name of the ADE Assembler or ADE Explorer session.

If not specified, the current session is used.

`?fileName t_fileName`

Path and name of the file to which results are to be exported.

`?view t_viewType`

Name of the output view to be exported.

The valid values are:

- "Detail"
- "Detail - Transpose"
- "Status"
- "Summary"
- "Yield"
- "Checks/Asserts"
- "Fault"
- "Current" (default)

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

?historyName *t_historyName*

Name of the history for which outputs are to be exported.

The default value is " ".

?testName *t_testName*

Name of the test for which the outputs are to be exported. This argument is useful for exporting results from a multi-test cellview.

Note: This argument is supported only for the Checks/Asserts and Fault result views.

?filterName *t_filterName*

Name of a saved filter to be applied before exporting outputs.

Note: This argument is supported only for the Checks/Asserts and Fault result views.

?clearAllFilters *g_clearAllFilters*

Specifies whether to clear all column filters before exporting outputs or not.

- *t*: Clears the column filters before exporting the outputs.

- *nil*: Does not clear the column filters before exporting the outputs.

Note: This argument is supported only for the Checks/Asserts and Fault result views.

Value Returned

t The given output view is successfully exported.

nil The given output view is not exported.

Examples

Exports the yield view of a Monte Carlo run to a file named *abc.csv*.

```
maeOpenSetup("solutions" "amptest" "maestro")  
;; the above function loads the given cellview and returns the session object  
=> "fnxSession0"  
maeSetVar("rload" 10k)  
;; sets value for the rload variable
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
=> t
maeRunSimulation(?runMode "Monte Carlo Sampling")
;; the above function runs the simulation for the given setup and returns the name
of the history
=> "MonteCarlo.2"
maeWaitUntilDone('All)
=> (0)
maeExportOutputView(?session "fnxSession0" ?fileName "./abc.csv" ?view "Yield")
=> "/servers/scratch02/testcases/adexl_workshop/design/abc.csv"
```

Saves the results of the Checks/Asserts result view for the current history of test demo_top:1 to a file named deviceChecks.csv. While doing that, it clears the filter named Device Checks.

```
maeExportOutputView( ?session "fnxSession0" ?fileName "./deviceChecks.csv" ?view
"Checks/Asserts" ?testName "demo_top:1" ?filterName "Device Checks"
?clearAllFilters t)
```

For the Checks/Asserts view, the optional arguments can be used to specify a named filter.

maeForceRdbStatus

```
maeForceRdbStatus(  
    t_sessionName  
    t_historyName  
    ?pointID x_pointID  
    ?designPointID x_designPointID  
    ?cornerName t_cornerName  
    ?toStatus t_toStatus  
)  
=> t / nil
```

Description

Forcefully sets the status of results for a history or a selected point in a history.

Arguments

<i>t_sessionName</i>	Name of the maestro session.
<i>t_historyName</i>	Name of the history.
<i>?pointID x_pointID</i>	ID of the point for which you need to change the status. If not specified, all points are selected.
<i>?designPointID x_designpointID</i>	ID of the design point for which you need to change the status. If not specified, all design points are selected.
<i>?cornerName t_cornerName</i>	Name of the corner. If not specified, all corners are selected.
<i>?toStatus t_toStatus</i>	Status to be forcefully set. The list of possible values that can be specified by this argument can be customized by using the <code>forceRdbStatusTo</code> environment variable. The default value is <code>done</code> .

Value Returned

<i>t</i>	The status is changed.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`nil`

The status is not changed due to an error.

Examples

Sets the result status for corner C1 in design point with ID 1 to done forcefully.

```
when(session = axlGetWindowSession( hiGetCurrentWindow() )
maeForceRdbStatus(session "Interactive.1" ?designPointID 1 ?cornerName "C1")
)
=> t
```

maeGetNBestDesignPoints

```
maeGetNBestDesignPoints(  
    [ ?count n_designPoints ]  
)  
=> l_bestDesignPoints / nil
```

Description

Returns the best design points for the opened results.

Arguments

?count n_designPoints

Number of best design points required.

The default value is 1.

Value Returned

l_bestDesignPoints

The list of best design points is returned.

nil

The list of best design points is not returned.

Examples

Return the three best design points for the results loaded.

```
; Load the result for history Interactive.1  
maeOpenResults(?history "Interactive.1")  
maeGetNBestDesignPoints(?count 3)  
=> (4 2 7)  
>  
maeCloseResults()  
=> t
```


maeGetOutputValue

```
maeGetOutputValue(  
    t_outputName  
    t_testName  
    [ ?cornerName t_cornerName ]  
    [ ?pointId x_pointId ]  
    [ ?evalType t_evalType ]  
)  
=> x_value / nil
```

Description

Returns value of the given output.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

<code>t_outputName</code>	Name of the output for which value is to be returned.
<code>t_testName</code>	Name of the test.
<code>?cornerName t_cornerName</code>	Name of the corner. If not specified, output value is returned for the nominal corner.
<code>?pointId x_pointID</code>	Point ID for which you need to get the output value. If not specified, output value is shown for the point with ID = 1.
<code>?evalType t_evalType</code>	Evaluation type. The valid values are "point", "corners", "sweeps", "all" The default value is "point".

Value Returned

<code>x_value</code>	The value of the given output is returned.
<code>nil</code>	The value for the given output is not returned.

Examples

Returns the value for the output `outputA` in the test `solutions:amptest:1`.

```
; Load the results for that history
maeOpenResults(?history "Interactive.1")
=> t
>
; Get the output value
maeGetOutputValue("outputA" "solutions:amptest:1")
=> 2.2
```

maeGetParamConditions

```
maeGetParamConditions(  
    x_designPointID  
    ?typeName t_typeName  
)  
=> l_designParamConditions / nil
```

Description

Returns the design parameter conditions for the given design point ID.

Arguments

x_designPointID ID of the design point for which the parameter conditions are to be returned.

?typeName x_designPointID
Specifies the type of information to be returned.
Possible values are: "parameters", "variables", "all".
Default value is "all".

Value Returned

l_designParamConditions
The design parameter conditions are returned for the given design point ID.

nil
The design parameter conditions are not returned for the given design point ID.

Examples

Returns the design parameter conditions for the best design point.

```
;create setup for simulation  
  
; Run Simulation  
maeRunSimulation()  
=> "Interactive.2"
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
; open the results
maeOpenResults(?history "LocalOpt.1")
=> t

; Find the best design point
bestPt=maeGetNBestDesignPoints(?count 2)
=> (8 10)
; view the conditions for the first best point

maeGetParamConditions(car(bestPt))
=> (("vdd" "3") ("opamp090/full_diff_opamp_AC/schematic/R2/m" "2")
)
; using the default value of the ?typeName argument, the function returns both
; parameters and variables
; close the results
maeCloseResults()
=> t
```

maeGetParameter

```
maeGetParameter(  
    t_paramName  
    [ ?typeName t_typeName ]  
    [ ?typeValue t_typeValue ]  
    [ ?session t_sessionName ]  
)  
=> t_value / nil
```

Description

Returns value of the given parameter for the given test or corner.

Arguments

t_paramName Name of the parameter.

Parameter names must contain at least five non-empty names each separated by a / (slash), typically representing the library/cell/view/instance/property to be modified. Multiple "instance/" strings can be used to specify hierarchical parameters such as library/cell/view/instance0/instance1/instance2/property.

?typeName t_typeName

Specifies whether the parameter details are to be returned for the given test or corner.

The possible values are "test" or "corner".

The default value is "test".

?typeValue t_typeValue

Name of the corner for which value is to be returned.

Note: This argument is ignored if *?typeName* is set to test.

The default value is "Global".

?session t_sessionName

Name of the session.

If not specified, the current session is used.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<i>t_value</i>	The value of the given parameter is returned for the given test or corner.
<i>nil</i>	The value of the given parameter is not returned for the given test or corner.

Examples

Returns the value of the parameter I1/1 for the corner C0.

```
maeOpenSetup("solutions" "amptest" "maestro")
;; the above function loads the given cellview and returns the session object
=> "session0"
maeGetParameter("solutions/amptest/schematic/I0/1" ?typeName "corner" ?typevalue
"C0")
=> 10n
```

maeGetResultOutputs

```
maeGetResultOutputs(  
    [ ?testName t_testName ]  
)  
=> l_outputNames / nil
```

Description

Returns the list of outputs for the opened result.

Arguments

?testName t_testName

Test for which the outputs are required. If not given, outputs for all the tests are returned.

Value Returned

l_outputNames

A list containing the name of outputs for the opened result is returned.

nil

The output names are not returned.

Examples

Returns the list of outputs for the opened result `Interactive.1`.

```
;Load the result for history Interactive.1  
maeOpenResults(?history "Interactive.1")  
=> t  
maeGetResultOutputs()  
=> ("Current" "DCGain" "/OUTN" "?OUTP")  
>  
maeCloseResults()  
=> t
```

maeGetResultTests

```
maeGetResultTests()  
=> l_testNames / nil
```

Description

Returns a list of tests for the opened result.

Arguments

None

Value Returned

<i>l_testNames</i>	A list containing the name of tests is returned for the opened result.
<i>nil</i>	The list of test names is not returned.

Examples

Returns the list of tests for the opened result `Interactive.1`.

```
;Load the result for history Interactive.1  
maeOpenResults(?history "Interactive.1")  
=> t  
maeGetResultOutputs()  
=> ("Current" "DCGain" "/OUTN" "?OUTP")  
>  
maeGetResultTests()  
>("solutions:ampTest:1" "solutions:ampTest:2")  
>  
maeCloseResults()  
=> t
```


maeGetResultsViewSelectedCellsDetails

```
maeGetResultsViewSelectedCellsDetails(  
    t_sessionName  
)  
=> l_resultData / l_faultResultData / nil
```

Description

Returns a list of data point ID, test name, and output name for each result value selected in the *Detail*, *Detail-Transpose*, or *Fault* results view of the currently visible run history. If the selected cell is in the column that does not show any results value, for example, *Test* or *Output*, no data is returned from that.

For the *Detail-Transpose* View, the function returns data only for the cells selected in the right pane.

Arguments

<code>t_sessionName</code>	Name of the maestro session.

Value Returned

<code>l_resultData</code>	A list containing lists of point, test and output name for each cell selected in the <i>Detail</i> or <i>Detail - Transpose</i> view of the <i>Results</i> tab.
<code>l_faultResultData</code>	A list of string values from each cell selected in the <i>Fault</i> view of the <i>Results</i> tab.
<code>nil</code>	In the following cases: ■ No result value is selected ■ The current results view none of <i>Detail</i> or <i>Detail - Transpose</i> ■ No history is loaded in the <i>Results</i> tab

Examples

The following example returns the values for the cells selected in the Results pane.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeGetResultsViewSelectedCellsDetails("fnxSession0")
=>(
  (3 "Oscillator_analysis" "/out")
  (1 "Oscillator_analysis" "RiseTime")
  (4 "Oscillator_analysis" "FallTime")
  (8 "Oscillator_analysis" "FallTime")
  (11 "Oscillator_analysis" "/out")
  (7 "Oscillator_analysis" "RiseTime")
)
```

The following example uses the value returned by

`maeGetResultsViewSelectedCellsDetails` to access the netlist directory:

```
t_sessionName = axlGetWindowSession( hiGetCurrentWindow() )
resultsView = maeGetResultsViewSelectedCellsDetails(t_sessionName)
resultsViewOneList = car(resultsView)
testName = nth(1 resultsViewOneList)
netlistDir = axlGetPointNetlistDir(historyEntry testName)
```

maeGetSpecStatus

```
maeGetSpecStatus(  
    t_outputName  
    t_testName  
    [ ?pointId n_pointId ]  
)  
=> pass / fail / undefined
```

Description

Returns the specification status for the given output, test and point id.

Arguments

<i>t_outputName</i>	Name of the output.
<i>t_testName</i>	Name of the test.
<i>?pointId n_pointId</i>	Point ID for which you need to get the spec status The default value is 1.

Value Returned

pass	The specification has passed.
fail	The specification has failed.
undefined	Indicates an invalid case.

Examples

Returns the specification status for the output `outputA` in the test `solutions:amptest:1`.

```
; Load the results for that history  
maeOpenResults(?history "Interactive.1")  
=> t  
>  
; Get the output value  
maeGetOutputValue("outputA" "solutions:amptest:1")  
=> 2.2  
; Get the spec status  
maeGetSpecStatus("outputA" "solutions:amptest:1")  
=> "pass"  
; close the results  
maeCloseResults()  
=> t
```

maeGetTestOutputs

```
maeGetTestOutputs(  
    t_testName  
    [ ?session t_session ]  
)  
=> l_outputs / nil
```

Description

Returns the list of outputs for the specified test in the current session or a specific maestro session.

Arguments

<i>t_testName</i>	Name of the test for which the list of outputs is required.
<i>?session t_session</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>l_outputs</i>	A list containing the outputs for the specified test is returned.
<i>nil</i>	The list of outputs is not returned for the specified test.

Examples

Returns the list of outputs for the test TRAN.

```
; Returns the name of the maestro session.  
maestroSession = maeOpenSetup("Two_Stage_Opamp" "OpAmp" "maestro_basic")  
=> "fnxSession8"  
  
; Returns the name of the tests.  
maeGetSetup(?typeName "tests" ?session maestroSession)  
=> ("AC" "TRAN" "AC:1" "AC:2")  
  
; Returns the outputs for the test TRAN.  
outputList = maeGetTestOutputs("TRAN" ?session maestroSession)  
=> (sevOutputStruct@0x220138a8 sevOutputStruct@0x220138c0  
sevOutputStruct@0x220138d8 sevOutputStruct@0x220138f0 sevOutputStruct@0x22013908  
sevOutputStruct@0x22013920 sevOutputStruct@0x22013938 sevOutputStruct@0x22013950  
sevOutputStruct@0x22013968)  
  
; Prints the name, expression or signal name for each output.  
foreach( mapcar output outputList or(output->name output->expression  
output->signal))
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
=> ("/OUT" "Swing" "SettlingTime" "RelativeSwingPercent" "PhaseMargin"  
    "/net06" "/netAVDD" "/IN" "/gnd!"  
)
```

maeGetOverallSpecStatus

```
maeGetOverallSpecStatus(  
    [ ?verbose verbose ]  
)  
=> t / nil
```

Description

Returns the overall specification status for the current history.

Arguments

<code>?verbose <i>verbose</i></code>	Specifies whether to print a detailed report including the pass or fail status for each test name and output name combination. The default value is <code>nil</code> .

Value Returned

<code>t</code>	The spec status for each test-output combination is returned. For a result that passed the specification, the status is shown as <code>PASSED</code> . For a result that either failed or is near the specification, the status is shown as <code>FAILED</code> . The function also returns a list containing the overall status and a sub list of <code>testName:outputName: <PASSED/FAILED></code> . The first value in the <code>OVERALL</code> list gives the overall status.
<code>nil</code>	Indicates that the function is not run successfully. For example, when no results are available.

Examples

Returns the overall specification status. When `?verbose` is set to `t`, an overall pass or fail status is shown for each output in addition to a detailed report for each output.

```
; Load the results for a history  
maeOpenResults(?history "Interactive.1")  
=> t  
>  
maeGetOverallSpecStatus(?verbose t)  
=>
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
Spec Status:
DCGain:lowValue : PASSED
DCGain:highValue : PASSED
DCGain:dcGain : PASSED
DCGain:lowValue2 : PASSED
DCGain:highValue2 : PASSED
DCGain:dcGain2 : PASSED
DCGain:maxCurrent : PASSED
Swing:Swing : PASSED
Swing:SettlingTime : PASSED
Swing:RelativeSwingPercent : PASSED
Swing:lowValue : PASSED
Swing:highValue : PASSED
Swing:Swing2 : PASSED
Swing:lowValue2 : PASSED
Swing:highValue2 : PASSED
Swing:maxCurrent : PASSED
overAll : PASSED
(("overAll" t)
  ("DCGain:lowValue" t)
  ("DCGain:highValue" t)
  ("DCGain:dcGain" t)
  ("DCGain:lowValue2" t)
  ("DCGain:highValue2" t)
  ("DCGain:dcGain2" t)
  ("DCGain:maxCurrent" t)
  ("Swing:Swing" t)
  ("Swing:SettlingTime" t)
  ("Swing:RelativeSwingPercent" t))
```

; When ?verbose is set to nil, only a list containing the status of each output at
; the test level is returned

```
maeGetOverallSpecStatus(?verbose nil)
=> t
```

```
maeGetOverallSpecStatus(?verbose nil)
(("overAll" t)
  ("DCGain:lowValue" t)
  ("DCGain:highValue" t)
  ("DCGain:dcGain" t)
  ("DCGain:lowValue2" t)
  ("DCGain:highValue2" t)
  ("DCGain:dcGain2" t)
  ("DCGain:maxCurrent" t)
  ("Swing:Swing" t)
  ("Swing:SettlingTime" t)
  ("Swing:RelativeSwingPercent" t)
  ("Swing:lowValue" t))
```

```
;close the results
maeCloseResults()
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Shows the overall results when one of the test-output result fails.

```
; Load the results for a history
maeOpenResults(?history "Interactive.2")
=> t

maeGetOverallSpecStatus(?verbose t)

Spec Status:
ac:delay_read1 : PASSED
tran:delay_read1 : FAILED

Overall : FAILED

(("Overall" nil)
 ("ac:delay_read1" t)
 ("tran:delay_read1" nil)
)
```


maeGetOverallYield

```
maeGetOverallYield(  
    t_historyName  
    [ t_sessionName ]  
)  
=> t / nil
```

Description

Returns the overall yield for the given history.

Arguments

<i>t_historyName</i>	Name of the history.
<i>t_sessionName</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>r_yieldInfo</i>	The yield details are displayed for the history results.
<i>nil</i>	The yield details are not displayed.

Examples

Returns the overall yield for the given history.

```
maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
  
maeRunSimulation()  
=> "Run.2"  
  
maeGetOverallYield("Run.2")  
=>(nil Yield 100 PassedPoints 2 ErrorPoints 0 )
```

maeImportHistory

```
maeImportHistory(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?session t_sessionName ]  
    [ ?history t_historyName ]  
    [ ?copyPSF g_copyPSF ]  
    [ ?overwrite g_overwrite ]  
)  
=> t / nil
```

Description

Imports the zip file for the given history from one cellview into the current cellview.

You can import histories only for cellviews that use the separate history management feature.

Arguments

<i>t_libName</i>	Name of a library.
<i>t_cellName</i>	Name of a cell in the given library.
<i>t_viewName</i>	Name of a maestro cellview from which the history is to be imported.
<i>?session t_sessionName</i>	Name of the ADE Assembler session. If not specified, the current session is used.
<i>?history t_historyName</i>	Name of the history to be imported.
<i>?copyPSF g_copyPSF</i>	A Boolean value that specifies whether to copy the <code>psf</code> directory along with the history zip. The possible values are: ■ <code>t</code>: Copies the <code>psf</code> directory along with the history zip. ■ <code>nil</code>: Does not copy the <code>psf</code> directory.
<i>?overwrite g_overwrite</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

A Boolean value that specifies whether to overwrite a history of the same name if any exists in the destination cellview.

The possible values are:

- `t`: Overwrites any history that exists with the same name.
- `nil`: Does not import the specified history if another history already exists with the same name.

Value Returned

<code>t</code>	The specified history is imported.
<code>nil</code>	The specified history is not imported.

Examples

Imports two histories from the `maestro2` cellview to the `maestro` cellview of the `ampTest` cell.

```
maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"

maeImportHistory("solutions" "ampTest" "maestro2" ?history '("Interactive.1"
"Interactive.2"))
=>t
```

maeOpenResults

```
maeOpenResults(  
    [ ?session t_sessionName ]  
    [ ?history t_historyName ]  
    [ ?run t_runName ]  
)  
=> t / nil
```

Description

Opens the result for the given history or run plan, and sets the result pointer to be used by other functions.

Arguments

`?session t_sessionName`

Name of the ADE Assembler or Explorer session.

If not specified, the current session is used.

`?history t_historyName`

Name of the history or child history.

If a run saves child histories, specify the name of the child history to open results.

You can also use this argument to open the results of a history saved for a run in the run plan.

`?run t_runName`

Name of the run plan.

This argument is considered only when the `?historyName` argument is not available. For example, while accessing the results of one run in another run of a run plan. Since the history is not yet created while you are writing the script, you can use the `?run` argument to access the results of the specified run.

Note: Use the `?run` argument only inside the pre-run script of a run to access the results of another run. This argument does not work when the SKILL function is run from CIW.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

t	The result for the given history or run plan is opened, and the result pointer is set to be used by other functions.
nil	The result for the given history or run plan is not opened.

Examples

Opens the result for the history Interactive.1.

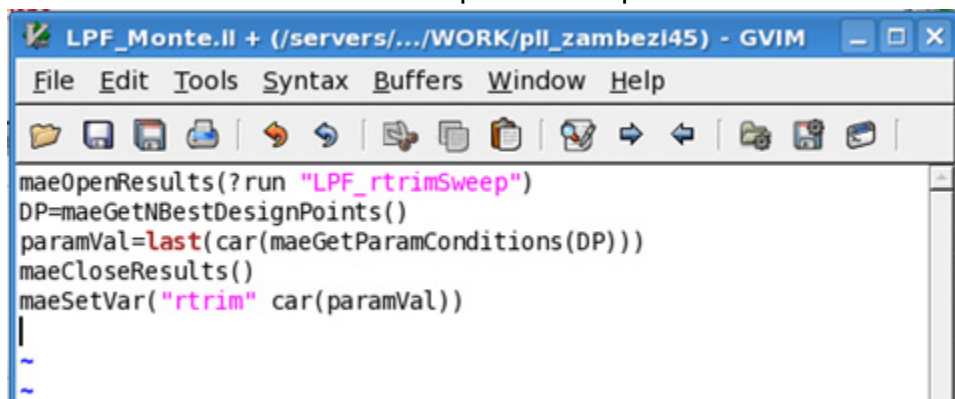
```
; Load the results for a history
maeOpenResults(?history "Interactive.1")
=> t
>
; Get the output value
maeGetOutputValue("outputA" "solutions:amptest:1")
=> 2.2
;close the results
maeCloseResults()
=> t
```

Opens the result for a run plan.

Note: If you are opening the results saved for a run plan, it is important to specify the name of a child history, Plan.0.Run.0, instead of the parent history, Plan.0.

```
maeOpenResults(?history "Plan.0.Run.0")
```

Loads the results of a run in the pre-run script of another run.



maeReadResDB

```
maeReadResDB (
    [ ?historyName t_historyName ]
    [ ?session t_sessionName ]
    [ ?run t_runName ]
)
=> h_resultsDBObj / nil
```

Description

Returns a handle to the results database for the given history.

This handle provides read-only access to the results database that contains objects of the following types:

- point: A design point
- corner: A corner defined for a particular design point
- test: A test defined for a particular corner
- param: A parameter defined for a particular corner
- output: An output defined for a particular test

There is a hierarchical relationship between the instances of these objects. For example, a point is associated with one or more corners. Each corner is associated with one or more tests. Each test is associated with zero or more outputs, and so on. As a result of this relationship, for an object, you can access the properties of the object itself and other objects related to it.

Each object has:

- Three properties, `name`, which returns the name of the object; `value`, which returns its value; and a property that returns ID of the parent object. For example, an object of type corner has a property `pointID` that returns the ID of the parent point object.
- A set of member functions that return the instances of that object type and other related types. For example, using an instance of type output, you can get the value of an output object and its parent test instance. Using the functions given for a test instance, you can get a corner instance. For a corner instance, you can get the parameters which were used to generate the output, as well as the point and its ID.

In addition, the result database provides a function `help()` for each object of the above mentioned types to displays the list and description of functions that can be called using that particular object. For example, function call `help('corner)` displays a list of all the

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

functions that can be called using an object of type corner. `help( 'all')` displays help for all the object types.

Arguments

`?historyName t_historyName`

Name of the history for which results are required. If not specified last active run will be used.

`?session t_sessionName`

Name of the session.

If not specified, the current session is used.

`?run t_runName`

Name of the run plan.

This argument is considered only when the `?historyName` argument is not used.

Value Returned

`h_resultsDBObj`

The handle to the results database is returned.

`nil`

The handle to the results database is not returned.

Examples

Returns a handle to access the results for a history named `Interactive.1`.

```
maeReadResDB( ?historyName "Interactive.1" )  
=> t
```

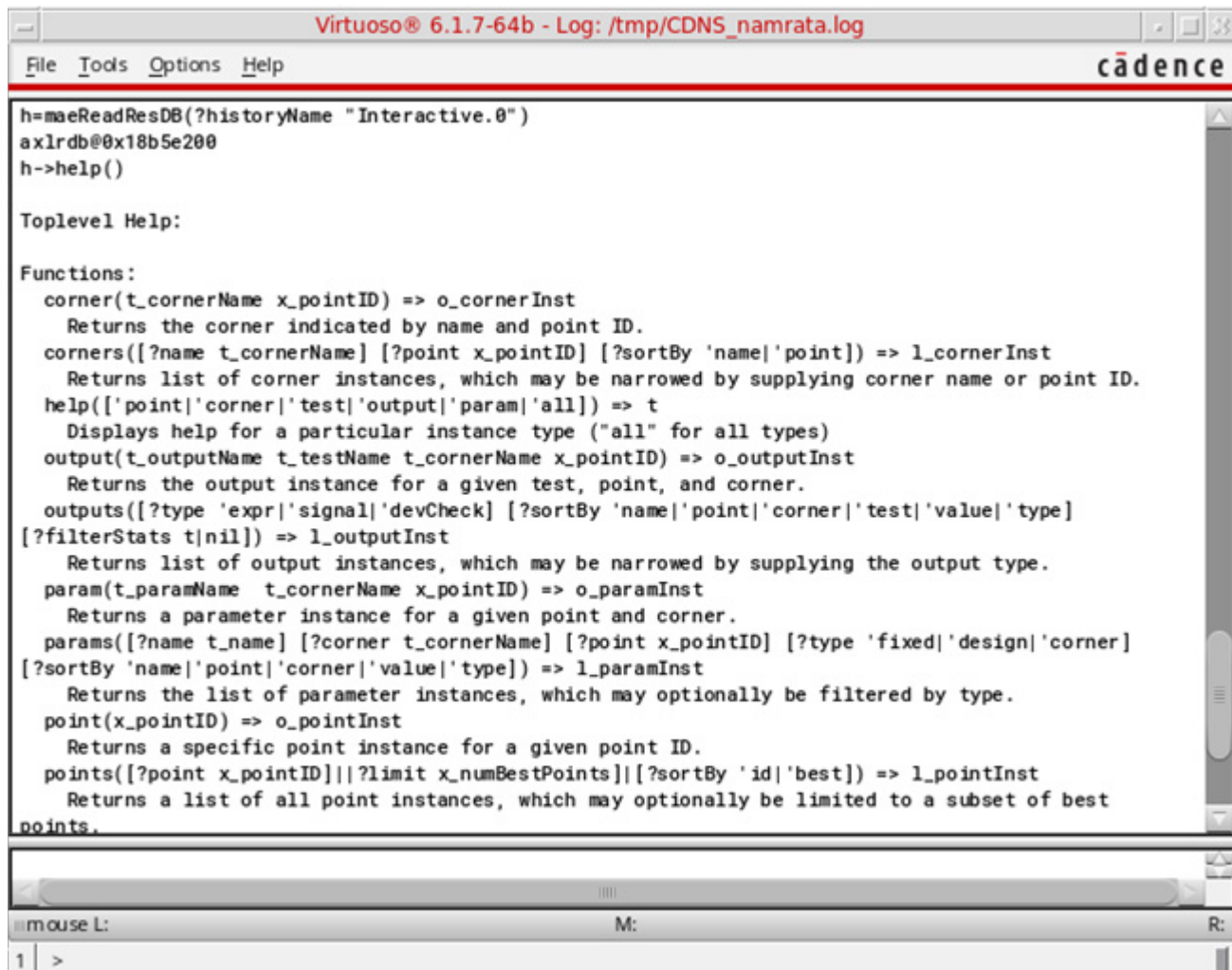
Opens the results database and displays built-in help.

```
rdb = maeReadResDB( ?historyName "Interactive.1" )  
=> axlrdb@0x18b5e200  
rdb -> help()
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Help is displayed in the CIW.



As shown in the help above, you can use the handle to access specific results from the results database.

; The following statement returns the point object for design point 1

```
pt = rdb->point(1)
=> axlrbdb@0x18b5e230
```

; The following code prints corner name, test name, output name and its value
; for each output of type expression

```
foreach(out pt->outputs(?type 'expr ?sortBy 'corner)

printf("corner=%s, test=%s, output=%s, value=%L\n" out->cornerName out->testName
out->name out->value)
)
corner=C0, test=test1, output=VAR("val1"), value=99.22984
corner=C0, test=test1, output=freq_res, value=5.112893e+08
```


Virtuoso ADE SKILL Reference

Maestro Cellview Functions

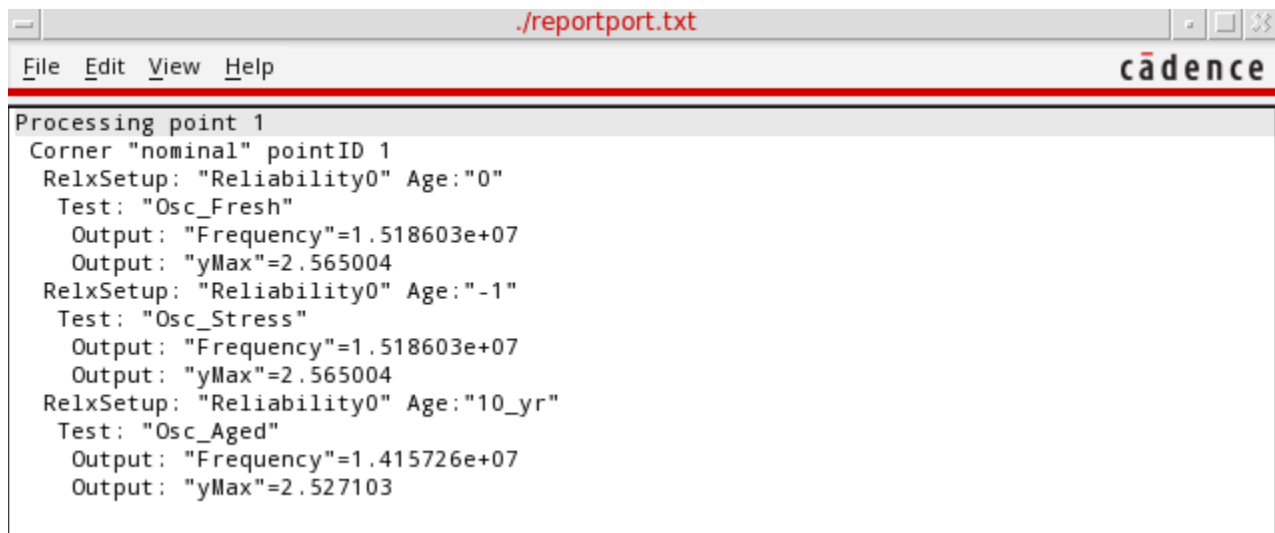
```
corner=C0, test=test2, output=((xmin(mag(IF("/I0/p1")) 1) / 1000000) - 200) * 4),
value=1299.23
corner=C0, test=test2, output=myCalib, value=99.22984
corner=C0, test=test2, output=calib_dummy, value=1
(axlrdbo@0x18476638 axlrdbo@0x18476620 axlrdbo@0x184765f0 axlrdbo@0x184765c
axlrdbo@0x18476590)
```

Prints the results of reliability analysis.

```
rdb=maeReadResDB(?historyName "Interactive.1")
reportport=outfile("./reportport.txt")
foreach(point rdb->points()
fprintf(reportport "Processing point %L\n" point->id)
foreach(corner rdb->corners()
fprintf(reportport " Corner %L pointID %L\n" corner->name corner->pointID)

foreach(relxSetup corner->relxSetups()
fprintf(reportport " RelxSetup: %L Age:%L \n" relxSetup->name relxSetup->age)
foreach(test relxSetup->tests()
fprintf(reportport " Test: %L\n" test->name)
foreach(output test->outputs()
unless(output->value=='Error' || !output->value
fprintf(reportport " Output: %L=%L\n" output->name output->value)
)
)
)
)
)
close(reportport)
view("./reportport.txt")
```

The .txt file that contains the report is displayed as shown below.



Prints the results of reliability analysis when the setup contains aging scenarios:

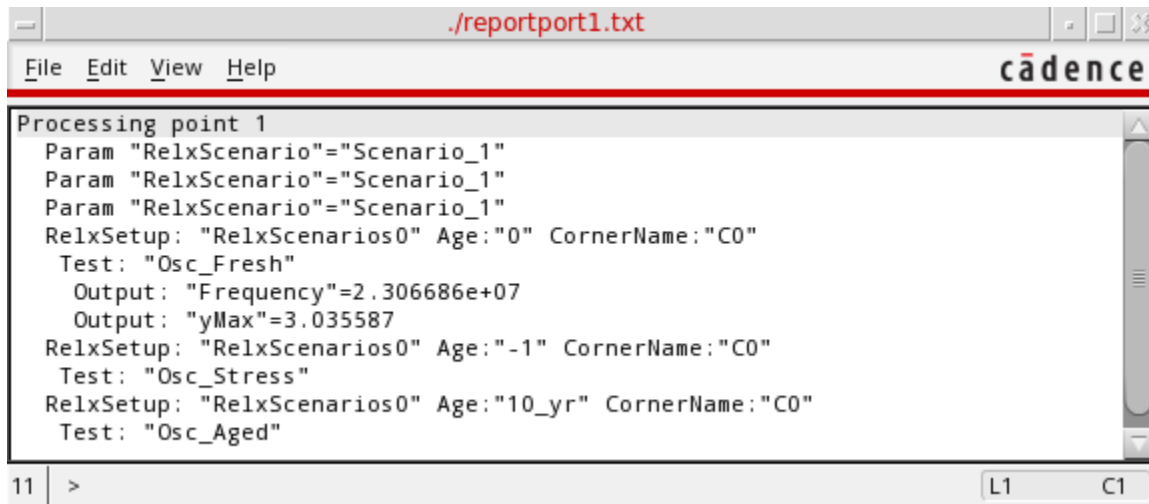
```
rdb1=maeReadResDB(?historyName "Interactive.3")
reportport1=outfile("./reportport1.txt")
foreach(point rdb1->points())
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
fprintf(reportport1 "Processing point %L\n" point->id)
foreach(param point->params()
  when(equal(param->name "RelxScenario")
    fprintf(reportport1 "  Param %L=%L\n" param->name param->value)
  )
)
foreach(relxSetup point->relxSetups()
  fprintf(reportport1 "  RelxSetup: %L Age:%L CornerName:%L\n" relxSetup->name
relxSetup->age relxSetup->cornerName)
  foreach(test relxSetup->tests()
    fprintf(reportport1 "    Test: %L\n" test->name)
    foreach(output test->outputs()
      unless(output->value=='Error || !output->value
        fprintf(reportport1 "      Output: %L=%L\n" output->name output->value)
      )
    )
  )
)
)
close(reportport1)
view("./reportport1.txt")
```

The txt file that contains the report is displayed as shown below.



Returns the *r* and *p* values from the results of Monte Carlo analysis.

```
rdb = (maeReadResDB ?historyName "MonteCarlo.0")
myoutputs = (setof x rdb->outputs(?type 'expr) (and (equal x->name "UGF"))))
myoutputvalues = (mapcar (lambda (x) (x->value)) myoutputs)
yvec = (drCreateVec 'double myoutputvalues)
xvec = (drCreateVec 'double (mapcar (lambda (x) (x->pointID)) myoutputs))
mywave = (drCreateWaveform xvec yvec)
myqq2 = (normalQQ (histogram2D mywave 10 "standard" nil nil))

printf("r = %f\n" myqq2->correlation)
printf("p = %f\n" myqq2->pValue)

=>
srrVec:0x37f331d0
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
srrVec:0x37f331e0
srrWave:0x37f770d0
srrWave:0x37f770f0
r = 0.976763
t
p = 0.053200
t
```

Prints the parameter combinations for all points in the history Interactive.4.

```
rdb=maeReadResDB(?historyName "Interactive.4")
printf("  Params %L\n" mapcar(lambda((x) sprintf(nil "%s=%L" x->name x->value)))
rdb->params()))

=> Params ("age=0" "vcontrol=1.8" "vdd=2" "age=-1" "vcontrol=1.8" "vdd=\"2.2\""
"age=\"5_yr\"" "vcontrol=1.8" "vdd=2" "age=\"10_yr\"" "vcontrol=1.8" "vdd=2"
"age=0" "vcontrol=2" "vdd=2" "age=-1" "vcontrol=2" "vdd=\"2.2\"" "age=\"5_yr\""
"vcontrol=2" "vdd=2" "age=\"10_yr\"" "vcontrol=2" "vdd=2")
```

maeRestoreHistory

```
maeRestoreHistory(  
    t_histName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Restores the given history as the active setup.

If you restore a history saved from ADE Explorer into an ADE Assembler session, all the existing tests are removed and only a single test, pertaining to the history, is displayed.

Arguments

<i>t_histName</i>	Name of the history to be restored.
<i>?session t_sessionName</i>	Name of the Assembler session. If not specified, the current session is used.

Value Returned

<i>t</i>	The specified history is restored as the active setup.
<i>nil</i>	The specified history is not restored.

Examples

Restores the setup from the history `Interactive.1`.

```
maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
maeRestoreHistory("Interactive.1")
```

maeRunLogDiagnostics

```
maeRunLogDiagnostics(  
    [ ?session t_session ]  
    [ ?lib t_libName ]  
    [ ?cell t_cellName ]  
    [ ?view t_viewName ]  
    [ ?directory t_directory ]  
    [ ?historyName t_historyName ]  
    [ ?points l_points ]  
    [ ?testNames l_testNames ]  
    [ ?checks l_checks ]  
    [ ?jobs x_jobs ]  
    [ ?tolerance x_tolerance ]  
    [ ?display g_display ]  
    [ ?log t_log ]  
)  
=> x_ID / nil
```

Description

Runs a list of checks on the log files for the specified history and generates the ADE Diagnostics report. Troubleshooting simulation errors using log files can be a difficult task because ADE Assembler saves multiple log files that are generated by different components of the LSCS architecture. The ADE diagnostics reports provides a detailed analysis of such errors and helps in debugging.

Arguments

`?session t_session`

Name of the maestro session.

`?lib t_libName`

Library name for the maestro cellview associated to the specified history.

`?cell t_cellName`

Cell name for the maestro cellview associated to the specified history.

`?view t_viewName`

View name for the maestro cellview associated to the specified history.

`?directory t_directory`

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Name of the directory exported by the Log packer window that contains all the logs for the specified history.

If all of the following arguments are specified, the precedence order is:

1. `?lib t_libName, ?cell t_cellName, and ?view t_viewName`
2. `?session t_session`
3. `?directory t_directoryName.`

`?historyName t_historyName`

Name of the history for which the ADE diagnostics report is to be generated.

Note: It is mandatory to specify this argument.

`?points l_points`

List of points for which the checks are to be run.

By default, the checks are run for all points.

`?testNames l_testNames`

List of names of the tests for which the checks are to be run.

By default, the checks are run for all tests.

`?checks l_checks`

List of checks or the categories of checks to be run for the generation of the ADE Diagnostics report.

By default, all checks are run.

For more information, see [Checks Performed by ADE Diagnostics](#).

`?jobs x_jobs`

Number of threads to be used for processing.

This argument can be used to speed up the analysis for larger simulation runs.

`?tolerance x_tolerance`

Tolerance in milliseconds (ms). It is used when comparing timestamps to account for clock drift between hosts.

`?display g_display`

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

When set to `t`, the ADE diagnostics report is displayed in the browser after the checks are run.

Specifying the `?display g_display` argument blocks the maestro session until the checks are run. To avoid this, use the function `maeDisplayLogDiagnostics` after `maeRunLogDiagnostics` is run instead of specifying this argument.

`?log t_log`

Name of the file where the output log for the diagnostics run is to be saved.

If not specified, the output is logged to the CIW or `CDS.log`.

Value Returned

`x_ID`

Returns a unique identifier for the diagnostics run. This ID is used with `maeDisplayLogDiagnostics` to display the ADE diagnostics report in a web browser.

`nil`

Indicates an error.

Examples

Displays the ADE diagnostics report in the web browser for the history `Interactive.1` in the cellview `Two_Stage_Opamp/Opamp/maestro`.

```
maeRunLogDiagnostics(?lib "Two_Stage_Opamp" ?cell "OpAmp" ?view "maestro"
?historyName "Interactive.1" ?display t)
=> 1
```

Related Topics

[maeDisplayLogDiagnostics](#)

[Debugging with ADE Diagnostics](#)

[maestro.debug.netlisting](#)

[maestro.debug.simAndEval](#)

[maestro.debug.beanstalk](#)

[maestro.debug.mainVirtuoso](#)

maeWriteDatasheet

```
maeWriteDatasheet(  
  [ ?directory t_directory ]  
  [ ?resultsSummary g_resultsSummary ]  
  [ ?testsSummary g_testsSummary ]  
  [ ?detailedResults g_detailedResults ]  
  [ ?plots g_points ]  
  [ ?designVarsSummary g_designVarsSummary ]  
  [ ?paramsSummary g_paramsSummary ]  
  [ ?cornersSummary g_cornersSummary ]  
  [ ?setupSummary g_setupSummary ]  
  [ ?launchBrowser g_launchBrowser ]  
  [ ?name t_name ]  
  [ ?session t_sessionName ]  
  [ ?historyName t_historyName ]  
)  
=> t / nil
```

Description

Writes the results for the given history in a datasheet.

Arguments

?directory t_directory

Name and path to the target directory where the datasheet is to be saved.

?resultsSummary g_resultsSummary

Boolean to specify whether or not to print a results summary sheet containing specification sheet pass/fail table.

The default value is *t*.

?testsSummary g_testsSummary

Boolean to specify whether or not to print a tests summary sheet containing details about the tests, sweeps, and corners.

The default value is *t*.

?detailedResults g_detailedResults

Name of the history to generate results for all the points.

The default value is *t*.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

<code>?plots <i>g_plots</i></code>	Boolean to specify whether or not to print the plots in the generated datasheet. The default value is <code>t</code>.
<code>?designVarsSummary <i>g_designVarsSummary</i></code>	Boolean to specify whether or not to save the design variable summary in the generated datasheet. The default value is <code>t</code>.
<code>?paramsSummary <i>g_paramsSummary</i></code>	Boolean to specify whether or not to save the parameters summary in the generated datasheet. The default value is <code>t</code>.
<code>?cornersSummary <i>g_cornersSummary</i></code>	Boolean to specify whether or not to save the corners summary in the generated datasheet. The default value is <code>t</code>.
<code>?setupSummary <i>g_setupSummary</i></code>	Boolean to specify whether or not to save the setup summary in the generated datasheet. The default value is <code>t</code>.
<code>?launchBrowser <i>g_launchBrowser</i></code>	Boolean to specify whether or not to launch a browser window to view the generated datasheet. The default value is <code>t</code>.
<code>?name <i>t_name</i></code>	Specifies the name of the top level file and directory created for the datasheet. For example, if <code>?name</code> is set to <code>myDatasheet</code>, the top level file and directory are named as <code>myDatasheet.html</code> and <code>myDatasheet/</code>.
<code>?session <i>t_sessionName</i></code>	Name of the ADE Assembler session. If not specified, the current session is used.
<code>?historyName <i>t_historyName</i></code>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Name of the history for which results are to be printed in a datasheet.

Certain run modes create a group history that contains multiple child histories. For such run modes, provide the name of the group history to create a consolidated datasheet for the group run, and provide the name of that child history to create a datasheet for a specific child history.

Value Returned

<code>t</code>	The results for the given history are written in a datasheet.
<code>nil</code>	The results for the given history are not written in a datasheet.

Examples

Runs a simulation for a maestro cellview and writes the results in a datasheet.

```
maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeRunSimulation()
=> "Interactive.2"
maeWriteDatasheet(?historyName "Interactive.2")
=> t
```

Creates a datasheet for run `Run.1` which is a part of the run plan history `Plan.1`.

```
maeWriteDatasheet(?historyName "Plan.1.Run.1")
=> t
```

For the same example, you can create a datasheet for the complete group history that contains all child histories by using the following code:

```
maeWriteDatasheet(?historyName "Plan.1")
=> t
```

Example script for Viewing Simulation Results

The following example shows how to view the simulation results:

```
; load the setup
maeOpenSetup("solutions" "ampTest" "maestro") > t

; check which tests are there in setup
maeGetSetup() -> ("solutions:ampTest:1" "solutions:ampTest:2")

; Enable a test "solution:ampTest:1"
maeSetSetup(?tests '("solutions:ampTest:1"))

; Set the value of global variable
"CAP" as 5p: maeSetVar("CAP" 5p)

; Disable corners C0 and C1
maeSetSetup(?corners '("C0" "C1") ?enabled nil)

; Run Simulation
maeRunSimulation()
=> "Interactive.2"

; open the results
maeOpenResults(?history "Interactive.2")
=> t

; view the list of outputs in the results
maeGetResultOutputs()

; Read the value of a specific output
maeGetOutputValue("outputA" "testA")
=> 2.2

; Find the best design point
maeGetNBestDesignPoints(?count 2)
=> (8 10)

; close the results
maeCloseResults()
=> t
```

Functions for Checks and Asserts

This topic lists the SKILL functions that let you manage violations reported for checks and asserts.

[maeCloseViolationDb](#)

[maeOpenViolationDb](#)

[maeRefreshViolationDisplay](#)

[maeWaiveViolation](#)

[maeUnWaiveViolation](#)

maeCloseViolationDb

```
maeCloseViolationDb(  
    x_dbId  
)  
=> t / nil
```

Description

Closes an open checks and asserts database using the database ID returned by maeOpenViolationDb.

Ensure that the history is closed before you run this function.

Arguments

<i>x_dbId</i>	An integer identifying the connection for an open checks and asserts database.

Value Returned

t	The open checks and asserts database is closed.
nil	The open checks and asserts database is not closed.

Examples

Opens a database, performs a task on it, and then closes it:

```
let( (dbId)  
    dbId = maeOpenViolationDb( "fnxSession0" "demo_top:1" "Interactive.1")  
    when( dbId != -1  
        maeWaiveViolation( dbId ?objectName "I19.PMO" ?objectType "instance"  
?checkerName "testLib/demo_top/Constr_3" ?analysisName "tran" ?comment "waiving")  
        maeCloseViolationDb(dbId)  
    )  
=> t
```

Related Topics

[maeOpenViolationDb](#)

maeOpenViolationDb

```
maeOpenViolationDb(  
    t_sessionName  
    t_testName  
    t_historyName  
)  
=> x_dbId / nil
```

Description

Opens a connection to the checks and asserts database for the given maestro cellview and returns a unique ID for the connection. You can use this ID to read or modify the database and its attached waiver SQL database using other SKILL functions. You must close this connection using `maeCloseViolationDb` to release the connection and complete the transactions on the database.

Note: Ensure that the history is closed before you run this function.

Arguments

<i>t_sessionName</i>	Name of the ADE Explorer or ADE Assembler session.
<i>t_testName</i>	Name of the test for which the results are to be opened.
<i>t_historyName</i>	Name of the history or child history. If a run saves child histories, specify the name of the child history to open results. You can also use this argument to open the results of a history saved for a run in the run plan.

Value Returned

<i>x_dbId</i>	The ID of the open database connection is returned.
<i>nil</i>	The ID of the open database connection is not returned.

Examples

Opens a database, performs a task on it, and then closes it.

```
let( (dbId)  
    dbId = maeOpenViolationDb( "fnxSession0" "demo_top:1" "Interactive.1")
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
when( dbId != -1
      maeWaiveViolation( dbId ?objectName "I19.PMO" ?objectType "instance"
?checkerName "testLib/demo_top/Constr_3" ?analysisName "tran" ?comment "waiving")
      maeCloseViolationDb(dbId)
    )
```

Related Topics

[maeCloseViolationDb](#)

maeRefreshViolationDisplay

```
maeRefreshViolationDisplay(  
    t_sessionName  
)  
=> t / nil
```

Description

Refreshes the `Checks/Asserts` results view. When it is already open and the database is modified, use this function to refresh the displayed results.

Arguments

<i>t_sessionName</i>	Name of the ADE Explorer or ADE Assembler session.

Value Returned

<i>t</i>	The results view is refreshed.
<i>nil</i>	The results view is not refreshed.

Examples

Refreshes the `Checks/Asserts` results view in the given session.

```
axlGetWindowSession( hiGetCurrentWindow() )  
=> "fnxSession0"  
maeRefreshViolationDisplay("fnxSession0")  
=> t
```


maeWaiveViolation

```
maeWaiveViolation(  
    x_dbId  
    [?objectName t_objectName]  
    [?objectType t_objectType]  
    [?checkerName t_checkerName]  
    [?analysisName t_analysisName]  
    [?time t_time]  
    [?comment t_comment]  
    [?historyName t_historyName]  
    [?designPoint x_designPoint]  
    [?corner t_cornerName]  
  
    )  
=> t / nil
```

Description

Adds a rule to the waivers SQL database that waives a check and assert for the given object.

Arguments

<code>x_dbId</code>	An integer identifying the connection for an open checks and asserts database.
<code>?objectName t_objectName</code>	Name of the net or instance object for which you want to waive a violation. Note: Specify the instance or net name in a format that is used in the netlist, not as shown in the <i>Inst/Net</i> column of the Checks/Asserts results table. For example, if an instance name is I15/I1/NM0, you should write it as I15.I1.NM0.
<code>?objectType t_objectType</code>	Type of the object for which you want to waive a violation.
<code>?checkerName t_checkerName</code>	Name of the checker in a hierarchical format.
<code>?analysisName t_analysisName</code>	Name of the analysis.
<code>?time t_time</code>	A double or a string value with units.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?comment t_comment` A string value indicating the reason for providing a waiver.

Note: You must provide a value for this argument.

`?historyName t_historyName`

Name of the history or the child history for which you want to waive a violation.

Note: The arguments `?historyName t_historyName`, `?designPoint x_designPoint`, and `?corner t_cornerName` are optional arguments and must be specified together only. You can use these arguments only when the variable `waiveViolationsByCorner` is set to `t`.

`?designPoint x_designPoint`

Point ID of the design point for which you want to waive the violation.

`?corner t_cornerName`

Name of the corner for which you want to waive the violation.

Value Returned

`t` A rule is added to the waivers SQL database that waives a check and assert for the given object.

`nil` No rule is added to the waivers SQL database.

Examples

Waives the violations when the following conditions are met:

- Point ID is 1
- Corner name is C1
- Object type is instance and its name is I19.PMok1
- Checker name is testLib/demo_top/Constr_3
- The analysis type is transient.
- History name is Interactive.1

```
let( (dbId)
dbId = maeOpenViolationDb( "fnxSession0" "demo_top:1" "Interactive.1")
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
when( dbId != -1
maeWaiveViolation( dbId ?objectName "I19.PMO" ?objectType "instance"
?checkerName "testLib/demo_top/Constr_3" ?analysisName "tran" ?comment "waiving"
?historyName "Interactive.1" ?designPoint 1 ?corner "C1")
maeCloseViolationDb(dbId)
)
```

maeUnWaiveViolation

```
maeUnWaiveViolation(  
    x_dbId  
    [?objectName t_objectName]  
    [?objectType t_objectType]  
    [?checkerName t_checkerName]  
    [?analysisName t_analysisName]  
    [?time t_time]  
)  
=> t / nil
```

Description

Removes from the waivers SQL database a waive rule matching the given criteria.

Ensure that the history is closed before you run this function.

Arguments

<code>x_dbId</code>	An integer identifying the connection for an open checks and asserts database.
<code>?objectName t_objectName</code>	Name of the net or instance object for which you want to waive a violation. Specify the instance or net name in a format that is used in the netlist, not as shown in the <i>Inst/Net</i> column of the Checks/Asserts results table. For example, if an instance name is I15/I1/NM0, you should write it as I15.I1.NM0.
<code>?objectType t_objectType</code>	Type of the object for which you are waiving a violation.
<code>?checkerName t_checkerName</code>	Name of the checker in hierarchical format.
<code>?analysisName t_analysisName</code>	Name of the analysis.
<code>?time t_time</code>	A double or a string value with units.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The rule matching the given criteria is waived from the waivers SQL database.
<code>nil</code>	No rule is waived from the SQL database.

Examples

Opens a database, removes a waiver database, and then closes it:

```
axlSession = axlGetWindowSession(hiGetCurrentWindow())
dbId = maeOpenViolationDb(axlSession "test_demoTop:1" "Interactive.3")
when( dbId
    ;;
    maeUnWaiveViolation(dbId ?objectName "I8.I0.MP1" ?objectType "instance"
?checkerName "testLib/demo_top/AssertionCheck" ?analysisName "tran" ?time
"101.1p")
    maeCloseViolationDb(dbId)
)
```

Functions to Manage Run Plans

This topic lists the SKILL functions that let you manage run plans in Virtuoso ADE Assembler.

[maeAddPostRunScript](#)

[maeAddPreRunScript](#)

[maeCopySimulationOptions](#)

[maeConvertViewForReferencedAndLocalRunPlanCorners](#)

[maeCreateNewRun](#)

[maeCreateRunCondition](#)

[maeCreateRunCopy](#)

[maeCopyTestToRun](#)

[maeDeleteRun](#)

[maeDeleteRunCondition](#)

[maeDeleteTest](#)

[maeEditSimulationOptions](#)

[maeEnableCorner](#)

[maeEnableGlobalVar](#)

[maeEnableParameter](#)

[maeEnableRun](#)

[maeEnableTests](#)

[maeGetCurrentRunPlanName](#)

[maeGetHistoryNameForCurrentRunInRunPlan](#)

[maelsFirstRunInRunPlan](#)

[maelsFinalRunCompleted](#)

maeAddPostRunScript

```
maeAddPostRunScript(  
    t_runName  
    t_postRunScriptFileName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds the given post-run script to the specified run in the run plan.

Arguments

<i>t_runName</i>	Name of the run to which the post-run script file is to be added.
<i>t_postRunScriptFileName</i>	Path of the post-run script.
<i>?session t_session</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>t</i>	The post-run script is added successfully.
<i>nil</i>	Indicates an error while adding a post-run script.

Examples

Adds a post-run script to run Run.0:

```
maeAddPostRunScript("Run.0" "/home/user/maeBestPoint.il")  
=> t
```

Related Topics

[Executing a Run Plan Based on Conditions](#)

maeAddPreRunScript

```
maeAddPreRunScript(  
    t_runName  
    t_preRunScriptFileName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds the pre-run script to the specified run in the run plan.

Arguments

<i>t_runName</i>	Name of the run to which a pre-run script is to be added.
<i>t_preRunScriptFileName</i>	Path to the pre-run script file.
<i>?session t_sessionName</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>t</i>	Pre-run script added successfully.
<i>nil</i>	Indicates an error while adding a pre-run script.

Examples

Adds a pre-run script to run Run.0:

```
maeAddPreRunScript("Run.0" "/home/user/maeBestPreScript.il")  
=> t
```

Related Topics

[Executing a Run Plan Based on Conditions](#)

maeCopySimulationOptions

```
maeCopySimulationOptions(  
    t_srcRun  
    [ ?destRun t_destRun ] | [ ?all g_all ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Copies the simulation options from the source run to the destination run. Different run modes have different simulation options, therefore, It is mandatory that the source and destination runs have the same run mode. Otherwise, the function reports an error.

Arguments

<i>t_srcRun</i>	Name of the run that is the source of simulation options.
<i>?destRun t_destRun</i>	Name of the destination run to which simulation options are to be copied. If not specified, the function looks for <i>?all</i> . If both are not specified, returns an error.
<i>?all g_all</i>	A Boolean value that specifies whether the simulation options are to be copied to all other runs in run plan that have the same run mode as that of the source run. By default, this argument is set to <i>nil</i> .
<i>?session t_session</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>t</i>	Simulation options are copied successfully.
<i>nil</i>	Simulation options are not copied.

Examples

Copies the simulation options from Run.0 to Run.1:

```
maeCopySimulationOptions("Run.0" ?destRun "Run.1")  
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Copies the simulation options from `Run.0` to all other runs that have the same run mode as that of `Run.0`:

```
maeCopySimulationOptions("Run.0" ?all t)
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeCreateNewRun

```
maeCreateNewRun(  
  [ ?session t_session ]  
  [ ?runName t_runName ]  
  [ ?newRunFromActiveSetup g_newRunFromActiveSetup ]  
)  
=> t / nil
```

Description

Creates a new run in the run plan.

Arguments

?session t_session Name of the session. If not specified, the current session is used.

?runName t_runName Name to be used for the new run created by this function. If no name is specified, the run is named as `Run.x` where, *x* is an incremental number.

?newRunFromActiveSetup g_newRunFromActiveSetup

Specifies whether to copy the setup from the active setup to the newly created run.

- *t*: Copies the setup from the active setup to the newly created run.
- *nil*: Does not copy the setup from the active setup to the newly created run and creates a blank run. This is the default.

Value Returned

t A new run is created.

nil Indicates an error.

Examples

Creates a new run named `myrun` in which the setup is copied from the active setup:.

```
maeCreateNewRun(?runName "myrun" ?newRunFromActiveSetup t)
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Creates a blank new run.

```
maeCreateNewRun(?runName "Run.2")  
=> t
```

Related Topics

[Creating a Run by Copying Another Run](#)

maeCreateRunCondition

```
maeCreateRunCondition(  
    t_sourceRunName  
    [ ?name t_name ]  
    [ ?run t_run ]  
    [ ?test t_test ]  
    [ ?output t_output ]  
    [ ?status t_status ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Adds a run condition to the specified run in the run plan.

Arguments

<i>t_sourceRunName</i>	Name of the run for which a run condition is to be created.
<i>?name t_name</i>	Name of the run condition to be created. When not specified, the default condition name <code>condx</code> is used, where <i>x</i> is in incremental number starting from 1.
<i>?run t_run</i>	Name of the run for which the condition is to be created. The default value for this argument is "".
<i>?test t_test</i>	Name of the test to be checked in the condition. The default value for this argument is "".
<i>?output t_output</i>	Name of the output to be checked in the condition. The default value for this argument is "".
	The valid values are <code>All</code> and <code>Any</code> .
<i>?status t_status</i>	Status to be checked for the new condition. The default value for this argument is "".
	The valid values are <code>pass</code> , <code>fail</code> , and <code>error</code> .
<i>?session t_session</i>	Name of the session. If not specified, the current session is used.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

t	A run condition is created.
nil	Indicates an error.

Examples

Adds a run condition named COND1 for Run.0. In this condition, the tool needs to look for the fail status for any output of the test testA:

```
maeCreateRunCondition("Run.0" ?name "COND1" ?run "Run.1" ?test "testA" ?output  
"Any" ?status "fail")  
=> t
```

Related Topics

[Executing a Run Plan Based on Conditions](#)

maeCreateRunCopy

```
maeCreateRunCopy(  
    ?run t_run  
    [ ?session t_session ]  
    [ ?copyName g_copyName ]  
)  
=> t / nil
```

Description

Creates a copy of the specified run in the run plan.

Arguments

<code>?run t_run</code>	Name of the source run to be copied. By default, this argument is set to <code>nil</code> . If not specified, the function returns an error.
<code>?session t_session</code>	Name of the session. If not specified, the current session is used.
<code>?copyName t_copyName</code>	Name to be assigned to the copied run. If not specified, the name of the copied run is in the format <i>sourceRunName:incrNum</i> where, <i>incrNum</i> starts with 1.

Value Returned

<code>t</code>	A copy of the given run is created.
<code>nil</code>	indicates an error

Examples

Creates a copy of run `Run.0`:

```
maeCreateRunCopy(?run "Run.0")  
=> t  
; The name of the copy of the run is Run.0:1
```

Creates a copy of run `Run.0` and the new run is named `Run.0_copy`:

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeCreateRunCopy(?run "Run.0" ?copyName "Run.0_copy")  
=> t
```

Related Topics

[Creating a Run by Copying Another Run](#)

maeCopyTestToRun

```
maeCopyTestToRun(  
    t_testName  
    [ ?session t_session ]  
    [ ?run t_run ] | [ ?all g_all ]  
)  
=> t / nil
```

Description

Copies one or more tests from the active setup to a run in the run plan.

Arguments

<code>t_testName</code>	Name of the test in the active setup to be copied to a particular run in the run plan.
<code>?session t_session</code>	Name of the session. If not specified, the current session is used.
<code>?run t_run</code>	Name of the run in the run plan where the specified test is to be copied.
<code>?all g_all</code>	A Boolean value that specifies whether to copy the test to all runs available in the given session. By default, this argument is set to <code>nil</code>. The valid values are <code>t</code> and <code>nil</code>. Note: You must specify either <code>?run</code> or <code>?all</code>.

Value Returned

<code>t</code>	The specified test is successfully copied to the run plan.
<code>nil</code>	Indicates an error.

Examples

The following example copies the test AC to run Run.2:

```
maeCopyTestToRun("AC" ?run "Run.2")
```

The following example copies the test AC to all runs in the run plan:

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeCopyTestToRun("AC" ?all t)
```

The following example copies multiple tests to run Run.2:

```
maeCopyTestToRun(list("Test.1" "Test.2" "Test.3") ?run "Run.2")
```

The following example copies multiple tests to runs Run.2 and Run.3:

```
maeCopyTestToRun(list("AC" "AC1" "AC2") ?run list("Run.2" "Run.3"))
```

maeDeleteRun

```
maeDeleteRun(  
    [ ?run l_run ] | [ ?all g_all ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Deletes the specified runs from the run plan.

Arguments

<code>?run l_run</code>	A list containing the names of the runs to be deleted. By default, this argument is set to <code>nil</code>. If not specified, the function does not delete any run and returns <code>nil</code>.
<code>?all g_all</code>	A Boolean value that specifies whether to delete all runs available in the given session. By default, this argument is set to <code>nil</code>. The valid values are <code>t</code> and <code>nil</code>. Note: You must specify either <code>?run</code> or <code>?all</code>.
<code>?session t_session</code>	Name of the session. If not specified, the current session is used.

Value Returned

<code>t</code>	The specified run plan is successfully deleted.
<code>nil</code>	Indicates an error.

Examples

Deletes two runs, `Run.0` and `Run.1`, from the run plan of the active setup:

```
maeDeleteRun(?run '("Run.0" "Run.1"))  
=> t
```

Deletes all runs from the run plan:

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeDeleteRun(?all t)  
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeDeleteRunCondition

```
maeDeleteRunCondition(  
    t_runName  
    [ ?name t_name ] | [ ?all g_all ]  
)  
=> t / nil
```

Description

Deletes a run condition from a run in the run plan.

Arguments

<i>t_runName</i>	Name of the run from which the run condition is to be deleted.
<i>?name t_name</i>	Name of the run condition to be deleted. If not specified, the function looks for <i>?all</i> . If both are not specified, returns an error.
<i>?all g_all</i>	Specifies whether all run conditions are to be deleted from the given run. By default, this argument is set to <i>nil</i> . The valid values are <i>t</i> and <i>nil</i> .

Value Returned

<i>t</i>	Deletes the given run condition from the specified run.
<i>nil</i>	Indicates an error.

Examples

Deletes the run condition named COND1 from run Run.7.

```
maeDeleteRunCondition("Run.7" ?name "COND1")  
=> t
```

Related Topics

[Executing a Run Plan Based on Conditions](#)

maeEditSimulationOptions

```
maeEditSimulationOptions(  
    t_run  
    l_options  
)  
=> t / nil
```

Description

Edits the simulation options for the specified run.

Arguments

<i>t_run</i>	Name of the run for which the simulation options are to be edited.
<i>l_options</i>	A list of name-value pairs of the simulation options to be edited for the run.

Value Returned

<i>t</i>	Simulation options edited successfully.
<i>nil</i>	Simulation options are not edited.

Examples

Edits four simulation options for run Run.1.

```
optionsList = ' ( ("numberOfPoints" "30") ("variation" "Process") ("samplingMethod"  
"lds") ("seed" "12"))  
maeEditSimulationOptions("Run.1" optionsList)  
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeEnableRun

```
maeEnableRun(  
    [ ?run t_run ]  
    [ ?enable g_enable ]  
    [ ?session t_session ]  
    [ ?all g_all ]  
)  
=> t / nil
```

Description

Enable or disables the runs in a run plan.

Arguments

<code>?run <i>t_run</i></code>	Name of run to be enabled or disabled in the run plan. By default, this argument is set to "". If not specified, the function looks for <code>?all</code> . If both are not specified, returns an error.
<code>?enable <i>g_enable</i></code>	A Boolean value that specifies the status to be set for the specified runs in the run plan. By default, this argument is set to <code>t</code> . The valid values are <code>t</code> and <code>nil</code> .
<code>?session <i>t_session</i></code>	Name of the session. If not specified, the current session is used.
<code>?all <i>g_all</i></code>	A Boolean value that specifies whether to enable or disable all runs in the run plan. By default, this argument is set to <code>nil</code> . The valid values are <code>t</code> and <code>nil</code> .

Value Returned

<code>t</code>	The status of the specified runs is successfully changed.
<code>nil</code>	The status of the runs is not changed.

Examples

Enables run `Run.0` in the run plan:

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeEnableRun(?run "Run.0")  
=> t
```

Disables run Run.0 in the run plan:

```
maeEnableRun(?run "Run.0" ?enable nil)  
=> t
```

Enables all runs in the run plan:

```
maeEnableRun(?all t)  
=> t
```

Disables all runs in the run plan:

```
maeEnableRun(?all t ?enable nil)  
=> t
```

Related Topics

[Modifying a Run in the Run Plan](#)

maeGetEnabledRuns

```
maeGetEnabledRuns(  
    [ ?session t_sessionName ]  
)  
=> l_enabledRuns / nil
```

Description

Returns a list of run IDs for the enabled runs in the run plan.

Arguments

?session t_sessionName

Name of the session.

If not specified, the current session is used.

Value Returned

l_enabledRuns

List of run IDs for enabled runs.

nil

No runs are enabled.

Examples

Return a list of run IDs for the enabled runs in the run plan.

```
maeGetEnabledRuns()  
=> (14930 14948)
```

maeGetNumberOfExecutedRuns

```
maeGetNumberOfExecutedRuns(  
    [ ?session t_sessionName ]  
    [ ?runPlanName t_runPlanName ]  
)  
=>x_runsCompleted
```

Description

Returns the number of runs executed in the flow. It only counts a run that has been completed, partial runs are not included. This is exclusive for the fault dropping flow.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

?runPlanName t_runPlanName

Name of the run plan whose status needs to be extracted.

Value Returned

x_runsCompleted Complete runs executed in the flow are returned.

Examples

Returns the number of runs:

```
maeGetNumberOfExecutedRuns(?runPlanName "Plan.0")  
=> 3  
)
```

maeGetHistoryLockFlag

```
maeGetHistoryLockFlag(  
    [ ?historyName t_historyName ]  
    [ ?session t_sessionName ]  
)  
=> n_flag / nil
```

Description

Returns the lock status for the specified history.

Arguments

?historyName t_historyName

Name of this history for which you want to get the lock status.

?session t_sessionName

Name of a maestro session.

If not specified, the currently active session is used.

Value Returned

n_flag

Numeric value that specifies a status:

- 0: Unlocked
- 1: Locked by user
- 2: Locked because it has been referenced in another history
- 3: Locked by user as well as because it has been referenced in another history
- 4: Locked by reuse in variation

nil

The lock status is not returned.

Examples

Returns the lock status of a Monte Carlo run history in the current session:

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeGetHistoryLockFlag("MonteCarlo.0")  
=> 4
```

maeGetRunPlan

```
maeGetRunPlan(  
    [ ?session t_sessionName ]  
    [ ?historyName t_historyName ]  
    [ ?enabled t_enabled ]  
)  
=> l_runPlanNames / nil
```

Description

Returns a list of all the runs available in the run plan for the given session or history.

This function is applicable only for ADE Assembler.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

`?session t_sessionName`

Name of an ADE Assembler session.

If not specified, the currently active session is used.

`?historyName t_historyName`

Name of this history for which you want to get the names of child histories corresponding to each run in the run plan.

Default value: " "

`?enabled t_enabled`

Enabled or disabled status according to which you want to get a list of runs.

Default value: " "

This argument cannot be used with `?historyName`.

Value Returned

`l_runPlanNames` List of the names of run plans

`nil` The list of run plan names is not returned.

Examples

Returns the names of runs for the current session.

```
maeGetRunPlan()  
=> ("Run.0" "Run.1")
```

Returns the names of runs for the given session.

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
maeGetRunPlan(?session sess1)  
=> ("Run.0" "Run.1" "Run.2")
```

Returns the names of runs for the given run plan history.

```
maeGetRunPlan(?historyName "Plan.0")  
=> ("Plan.0.Run.0" "Plan.0.Run.1")
```

maeGetCurrentRunPlanName

```
maeGetCurrentRunPlanName (  
    )  
=> t_runPlanName
```

Description

Returns the name of the history of the current run plan being run.

Arguments

None

Value Returned

<i>t_runPlanName</i>	Name of the current run plan is returned.

Examples

Returns the name of the current run plan being run.

```
Example: maeGetCurrentRunPlanName ()  
=> "Plan.0"
```

maeGetHistoryNameForCurrentRunInRunPlan

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

Description

Returns the history name for the current run in the run plan.

Arguments

None

Value Returned

<i>t_historyName</i>	The history name for the run being executed is returned.
<i>nil</i>	No history available for the current run.

Examples

Returns the history name for the current run in the run plan.

```
maeGetHistoryNameForCurrentRunInRunPlan()  
=> "Plan.0.Run.1"
```


maelsFirstRunInRunPlan

```
maeIsFirstRunInRunPlan(  
    [ ?session t_sessionName ]  
    [ ?historyName t_historyName ]  
)  
=> t / nil
```

Description

Checks if the current run is the first run executed in the run plan. Implemented exclusively for the fault dropping flow.

Arguments

?session t_sessionName

Name of the session.

If not specified, the current session is used.

?historyName t_historyName

History name for the current run being executed.

Value Returned

t The current run is the first executed run in the run plan.

nil The current run is not the first executed run in the run plan.

Examples

Checks if the current run is the first run executed in the run plan.

```
maeIsFirstRunInRunPlan(?historyName "Plan.0.Run.0")  
=> t
```

maelsFinalRunCompleted

```
maelsFinalRunCompleted(  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Checks if the current run is the last run executed in the run plan. A run is said to be the last run if either no further runs exist in the run plan or no undetected faults are found in the current run.

Implemented exclusively for the fault dropping flow.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

<i>t</i>	The current run is the final run executed in the run plan.
<i>nil</i>	The current run is not the final run executed in the run plan.

Examples

Checks if the current run is the last run executed in the run plan.

```
maelsFinalRunCompleted()  
=> t
```

Functions that Provide Support for Run Plan

This topic lists the SKILL functions that provide support for the run plan.

[maeDeleteTest](#)

[maeEnableCorner](#)

[maeEnableGlobalVar](#)

[maeEnableParameter](#)

[maeEnableTests](#)

[maeGetCurrentRunMode](#)

[maeGetSetup](#)

[maeSaveSetupStateForOptimization](#)

[maeSetStartingPointForOptimization](#)

[maeOpenResults](#)

[maeReadResDB](#)

Functions to Work with Plotting Templates

This topic lists the SKILL functions that let you work with plotting templates in maestro cellviews.

[maeGetAllPlottingTemplates](#)

[maePlotWithPlottingTemplate](#)

[maeSaveImagesUsingPlottingTemplate](#)



Video

You can watch a video demonstration of these functions at [SKILL Functions to Work with Plotting Templates](#).

maeGetAllPlottingTemplates

```
maeGetAllPlottingTemplates(  
    [ ?session t_sessionName ]  
)  
=> l_templates / nil
```

Description

Returns a list of all plotting templates saved in the given session.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

l_templates

The list of plotting templates saved in the given session is returned. Each list item contains the template name, history name, time stamp, and a description for the template.

nil

Indicates that there is an error or no plotting template is available.

Examples

Returns a list of templates and use those for plotting for results saved in a history.

```
maeGetAllPlottingTemplates()  
=> (("p2" "Interactive.29" "May 14 12:19:12 2019" "")  
    ("p1" "Interactive.29" "May 14 12:19:58 2019" "test")  
)  
  
maePlotWithPlottingTemplate(?history "Interactive.29" ?name "templateName")  
=> (window:33)
```

maePlotWithPlottingTemplate

```
maePlotWithPlottingTemplate(  
    [ ?session t_sessionName ]  
    [ ?history t_historyName ]  
    [ ?replaceMode t_replaceMode ]  
    [ ?name t_templateName ]  
  
    )  
=> l_windowsPlotted / nil
```

Description

Plots the results for the given history using the specified plotting template.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

`?session t_sessionName`

Name of the maestro session.

If not specified, the current session is used.

`?history t_historyName`

Name of the history for which the results are to be plotted

If not specified, the history saved for the last run is used.

`?replaceMode t_replaceMode`

Specifies the plotting mode.

- `t`: Specifies that the currently open window should be replaced with the plotted template. This is the default.
- `nil`: Specifies that the template should open in a new window.

`?name t_templateName`

Name of the template to be used for plotting.

Value Returned

`l_windowsPlotted` A list of windows plotted with the given template is returned.

`nil` Indicates an error.

Examples

Returns a list of templates and use those for plotting for results saved in a history.

```
maeGetAllPlottingTemplates()  
=> (("p2" "Interactive.29" "May 14 12:19:12 2019" "")  
    ("p1" "Interactive.29" "May 14 12:19:58 2019" "test")  
)  
  
maePlotWithPlottingTemplate(?history "Interactive.29" ?name "p1")  
=> (window:33)
```

maeSaveImagesUsingPlottingTemplate

```
maeSaveImagesUsingPlottingTemplate(  
  [ ?session t_sessionName ]  
  [ ?history t_historyName ]  
  [ ?replaceMode t_replaceMode ]  
  [ ?name t_templateName ]  
  [ ?dir t_dirPath ]  
  [ ?saveEachSubwindowSeparately t_saveEachSubwindowSeparately ]  
)  
=> t / nil
```

Description

Plots and saves the results for the given history using the specified plotting template.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

?history t_historyName

Name of the history for which the results are to be plotted

If not specified, the history saved for the last run is used.

?replaceMode t_replaceMode

Specifies the plotting mode.

- *t*: Specifies that the currently open window should be replaced with the plotted template.
- *nil*: Specifies that the template should open in a new window. This is the default.

?name t_templateName

Name of the template to be used for plotting.

?dir t_dirPath

Path to the directory in which plots are to be saved.

?saveEachSubwindowSeparately t_saveEachSubwindowSeparately

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Specifies whether to save each subwindow in a separate image.

- `t`: Saves each subwindow in a separate image. This is the default.
- `nil`: The subwindows are not saved in separate images.

Value Returned

<code>t</code>	The windows are successfully plotted.
<code>nil</code>	Indicates an error.

Examples

Returns a list of templates and use those for plotting for results saved in a history.

```
maeGetAllPlottingTemplates()  
=> (("p2" "Interactive.28" "May 14 12:19:12 2019" "")  
    ("p1" "Interactive.29" "May 14 12:19:58 2019" "test")  
)
```

Plots the templates and saves the windows as images, for example, window:33.png, in the current project directory.

```
maeSaveImagesUsingPlottingTemplate(?history "Interactive.29" ?replaceMode nil  
?name "p1" ?dir "." ?saveEachSubwindowSeparately t)  
=> t
```

Functions for Sensitivity Analysis Setup

This topic lists the SKILL functions that let you manage the setup in the Sensitivity form for Sensitivity Analysis in ADE Assembler.

[maeSensDeleteModel](#)

[maeSensDeleteModelGroup](#)

[maeSensDeleteParameter](#)

[maeSensDeleteVar](#)

[maeSensEnableDesignVariation](#)

[maeSensEnableStatVariation](#)

[maeSensGetModel](#)

[maeSensGetModelGroup](#)

[maeSensGetModels](#)

[maeSensGetParameter](#)

[maeSensGetParameters](#)

[maeSensGetVar](#)

[maeSensGetVars](#)

[maeSensSetMethod](#)

[maeSensSetModel](#)

[maeSensSetModelGroup](#)

[maeSensSetParameter](#)

[maeSensSetVar](#)

maeSensDeleteModel

```
maeSensDeleteModel(  
    t_modelName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified model from the setup for sensitivity analysis in the current or the specified maestro session.

Arguments

<i>t_modelName</i>	Name of the model to be deleted.
<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>t</i>	The model is deleted.
<i>nil</i>	The model is not deleted. You can review the errors in the log.

Examples

Deletes a model from the run options:

```
maeSensDeleteModel("gpdK045.scs")  
=> t
```

maeSensDeleteModelGroup

```
maeSensDeleteModelGroup(  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes all the model groups specified in the *Model Groups* setting on the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

t

The model group is successfully deleted.

nil

The model group is not deleted. You can review the errors in the log.

Examples

Deletes all the model groups from the run options.

```
maeSensDeleteModelGroup()  
=> t
```

maeSensDeleteParameter

```
maeSensDeleteParameter(  
    t_parameterName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified parameter from the setup for Sensitivity analysis in the current or the specified maestro session.

Arguments

t_parameterName Name of the parameter. The parameter name must contain at least five non-empty string values, each separated by a / in the following format:

library-name/cell-name/view-name/instance-name/property-name

For hierarchical parameters, specify the hierarchical path in *instance-name*. For example, mylib/mycell/view1/instance0/instance1/instance2/length.

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

t The parameter is successfully deleted.

nil The parameter is not deleted. You can review the errors in the log.

Examples

Deletes the specified parameter from the run options.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeSensDeleteParameter("minimal/cm/schematic/NM0/fw")  
=> t
```

maeSensDeleteVar

```
maeSensDeleteVar(  
    t_varName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified variable from the setup for Sensitivity analysis in the current or the specified maestro session. You can also use this function to delete the value for the *Temperature* field in the Sensitivity form.

Arguments

<i>t_varName</i>	Name of the variable to be deleted. To delete the value for temperature, specify "temperature".
<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>t</i>	The variable is deleted.
<i>nil</i>	Indicates an error.

Examples

Deletes the specified variable from the run options.

```
maeSensDeleteVar("vdd")  
=> t
```

maeSensEnableDesignVariation

```
maeSensEnableDesignVariation(  
    g_status  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Selects or deselects the *Enable Design and PVT Variation* check box on the Sensitivity form for sensitivity analysis in the current or the specified maestro session.

Arguments

<i>g_status</i>	Selects or deselects the <i>Enable Design and PVT Variation</i> check box on the Sensitivity form.

<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>t</i>	The <i>Enable Design and PVT Variation</i> check box is selected or deselected successfully.
<i>nil</i>	The selection status of the check box is not altered.

Examples

Clears the *Enable Design and PVT Variation* check box on the Sensitivity form in the active setup.

```
maeSensEnableDesignVariation(nil)  
=> t
```


maeSensEnableStatVariation

```
maeSensEnableStatVariation(  
    g_status  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Selects or deselects the *Enable Variation of Statistical Parameters* check box in the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

g_status Selects or deselects the *Enable Variation of Statistical Parameters* check box on the Sensitivity form.

■ *t*: Selects the check box.

■ *nil*: Deselects the check box.

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

t The *Enable Variation of Statistical Parameters* check box is selected or deselected successfully.

nil The selection status of the check box is not altered.

Examples

Clears the *Enable Variation of Statistical Parameters* check box in the Sensitivity form in the active setup.

```
maeSensEnableStatVariation(nil)  
=> t
```

maeSensGetModel

```
maeSensGetModel(  
    t_modelName  
    [ ?session t_sessionName ]  
)  
=> l_modelDetails / nil
```

Description

Returns a list of valid values and the nominal value for the specified model from the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

<i>t_modelName</i>	Name of the model for which the values are to be returned.
<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>l_modelDetails</i>	A list containing the following items: ■ Valid values for the given model ■ The nominal value specified for the given model
<i>nil</i>	Indicates an error.

Examples

Returns the details of the specified model.

```
maeSensGetModel("gpdK045.scs")  
=> ("ff mc dd" "ff")
```

maeSensGetModelGroup

```
maeSensGetModelGroup(  
    [ ?session t_sessionName ]  
)  
=> l_modelGroupDetails / nil
```

Description

Returns a list of values specified in the *Model Groups* setting on the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

l_modelGroupDetails

A list containing the following two items is returned.

- A list of model group names specified in the *Value* column
- The name of the model group specified in the *Nominal* column

nil Indicates an error.

Examples

Returns and prints the model groups specified in the setup for Sensitivity analysis.

```
;; get and print details of the model group  
  
mgroup = (maeSensGetModelGroup)  
=> ("mg1 mg2" "mg1")  
  
(when mgroup  
  (printf "Sensitivity Model Group value = %s, nominal value = %s\n" (car mgroup)  
    (cadr mgroup))  
)
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
=> Sensitivity Model Group value = mg1 mg2, nominal value = mg1  
=> t
```

maeSensGetModels

```
maeSensGetModels(  
    [ ?session t_sessionName ]  
)  
=> l_modelNames / nil
```

Description

Returns a list containing the names of all enabled models on the Sensitivity form for sensitivity analysis in the current or the specified maestro session.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

l_modelNames

A list with the names of all enabled models.

nil

Indicates an error.

Examples

Returns a list of names of enabled models on the Sensitivity form.

```
maeSensGetModels()  
=> ("gpdK045.scs" "gpdK045_resistor.scs")
```

Deletes the existing models from the Sensitivity form, adds two new models, uses the `maeSensGetModels` function to get all enabled models, and then prints their details.

```
;models  
models = (maeSensGetModels) ;returns the model display names not paths  
(foreach m models  
    (maeSensDeleteModel m)  
)  
  
(maeSensSetModel "spectre/gpdK045.scs" "ss tt ff" ?nominalValue "tt")  
;path is absolute or relative to include path  
(maeSensSetModel "models.scs" "mc" ?nominalValue "mc")  
;path is absolute or relative to include path  
models = (maeSensGetModels) ;returns the model display names not paths  
=> ("cds_ff.scs" "cds_ff_res.scs")
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
(printf "Sensitivity models = %L\n" models)

(foreach m models
  modelval = (maeSensGetModel m)
  (printf "Model %s value = %s, nominal value = %s\n" m (car modelval) (cadr
    modelval)))

=> Model cds_ff.scs value = ff ss fs, nominal value = ff
Model cds_ff_res.scs value = ff_res1 ss_res, nominal value = ss_res
```

maeSensGetParameter

```
maeSensGetParameter(  
    t_parameterName  
    [ ?session t_sessionName ]  
)  
=> l_paramDetails / nil
```

Description

Returns a list of valid values and the nominal value for the specified parameter from the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

t_parameterName Name of the parameter. The parameter name must contain at least five non-empty string values, each separated by a / in the following format:

library-name/cell-name/view-name/instance-name/property-name

For hierarchical parameters, specify the hierarchical path in *instance-name*. For example, mylib/mycell/view1/instance0/instance1/instance2/length.

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

l_paramDetails A list containing the following two items is returned.

- A list of valid values for the given parameter
- The nominal value specified for the given parameter

nil Indicates an error.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Examples

Returns the values set for the given parameter.

```
maeSensGetParameter("minimal/cm/schematic/NM0/fw")  
=> ("120n:10n:400n" "120n")
```


maeSensGetParameters

```
maeSensGetParameters(  
    [ ?session t_sessionName ]  
)  
=> l_parameterNames / nil
```

Description

Returns a list of all enabled parameters on the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

<i>l_parameterNames</i>	A list containing the names of enabled parameters is returned.
<i>nil</i>	Indicates an error.

Examples

Returns a list of names of enabled parameters on the Sensitivity form.

```
maeSensGetParameters()  
=> ("minimal/cm/schematic/NM0/fw" "Two_Stage_Opamp/AmpIn/schematic/M3/fingers"  
    "Two_Stage_Opamp/AmpIn/schematic/M3/w")
```

maeSensGetVar

```
maeSensGetVar(  
    t_varName  
    [ ?session t_sessionName ]  
)  
=> l_varDetails / nil
```

Description

Returns a list of valid values and the nominal value for the specified global variable from the Sensitivity form for sensitivity analysis in the current or the specified maestro session.

Arguments

<i>t_varName</i>	Name of the global variable. For example, "temperature" returns the values set for the <i>Temperature</i> setting.

<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>l_varDetails</i>	A list containing the following items: ■ Valid values for the given variable ■ The nominal value specified for the given variable
<i>nil</i>	Indicates an error.

Examples

Returns the value set for *Temperature* and the variable *rload*.

```
maeSensGetVar("temperature")  
=> ("-40 70 20" "20")  
  
maeSensGetVar("rload")  
=> ("1K 2K" "1K")
```

maeSensGetVars

```
maeSensGetVars(  
    [ ?session t_sessionName ]  
)  
=> l_varNames / nil
```

Description

Returns a list containing the names of all enabled global variables, including *Temperature*, on the Sensitivity form for the Sensitivity Analysis run mode in the current or the specified maestro session.

Arguments

`?session t_sessionName`

Name of the maestro session.

If not specified, the current session is used.

Value Returned

`l_varNames`

A list containing the names of enabled global variables.

`nil`

Indicates an error.

Examples

Returns the names of all global variables in the active setup.

```
maeSensGetVars()  
=> ("temperature" "cload" "rload")
```

maeSensSetMethod

```
maeSensSetMethod(  
    t_methodName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets the value in the *Method* field on the Sensitivity form for Sensitivity analysis in the current or the specified maestro session.

Arguments

<i>t_method</i>	Name of the method to be set. The possible values are: ■ "Hammersley" ■ "OFAT Sweep" ■ "OFAT 3-level"
<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>t</i>	The method is set.
<i>nil</i>	Indicates an error.

Examples

Sets the specified method as the value for the *Method* field on the Sensitivity form.

```
maeSensSetMethod("OFAT Sweep")  
=> t
```

maeSensSetModel

```
maeSensSetModel(  
    t_modelName  
    g_modelValue  
    [ ?nominalValue g_nominalValue ]  
    [ ?session t_sessionName ]  
    [ ?modelTest t_modelTest ]  
    [ ?modelBlock t_modelBlock ]  
)  
=> t / nil
```

Description

Adds a model on the Sensitivity form for Sensitivity Analysis in the current or the specified maestro session.

Arguments

<i>t_modelName</i>	Name of the model.
<i>g_modelValue</i>	Value range, which is the names of sections to be used from the specified model.
<i>?nominalValue g_nominalValue</i>	Nominal value, which is the section name to be used for the nominal run.
<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.
<i>?modelTest t_modelTest</i>	Name of the test for which the model is to be used. The default value is "All". This specifies that the model is associated with all the tests in the cellview.
<i>?modelBlock t_modelBlock</i>	Name of the block for which the model is to be used. The default value is "Global". This specifies that the model is associated with all the design blocks.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Value Returned

<code>t</code>	The model is set successfully.
<code>nil</code>	Indicates an error.

Examples

Sets the model `gpdk045.scs` on the Sensitivity form.

- Sections `ff` and `mc` are used by all the tests.
- Section `mc` is used for the nominal run:

```
maeSensSetModel("gpdk045.scs" "ff mc" ?nominalValue "mc")  
=> t
```

maeSensSetModelGroup

```
maeSensSetModelGroup(  
    g_modelGroupValue  
    [ ?nominalValue g_nominalValue ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a model group on the Sensitivity form for sensitivity analysis in the current or the specified maestro session.

Arguments

g_modelGroupValue Value range. These are the names of model groups.

?nominalValue g_nominalValue

Nominal value, which is the name of model group to be used for the nominal run.

?session t_sessionName

Name of the maestro session.

If not specified, the current session is used.

Value Returned

t The model group is set successfully.

nil Indicates an error.

Examples

Sets the model group `modelGroupA`, `modelGroupB` to the Sensitivity form.

```
maeSensSetModelGroup("modelGroupA modelGroupB" ?nominalValue "modelGroupA")  
=> t
```

maeSensSetParameter

```
maeSensSetParameter(  
    t_parameterName  
    g_parameterValue  
    [ ?nominalValue g_nominalValue ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a parameter on the Sensitivity form for Sensitivity analysis in the current or the specified maestro session. If a parameter already exists with the given name, its value is updated.

Arguments

<i>t_parameterName</i>	Name of the parameter. The parameter name must contain at least five non-empty string values, each separated by a / in the following format: <i>library-name/cell-name/view-name/instance-name/property-name</i> to be modified. For hierarchical parameters, specify the hierarchical path separated by / in <i>instance-name</i> . For example, mylib/mycell/view1/instance0/instance1/instance2/length.
<i>g_parameterValue</i>	Value range for the parameter.
<i>?nominalValue g_nominalValue</i>	Nominal value for the parameter.
<i>?session t_sessionName</i>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<i>t</i>	The parameter is set.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`nil` Indicates an error.

Examples

Set the parameter `NM0 / fw` on the Sensitivity form:

```
maeSensSetParameter("minimal/cm/schematic/NM0/fw" "100n:10n:140n")  
=> t
```

```
maeSensSetParameter("minimal/cm/schematic/NM0/fw" "100n:10n:140n" ?nominalValue  
"120n")  
=> t
```

maeSensSetVar

```
maeSensSetVar(  
    t_varName  
    g_varValue  
    [ ?nominalValue g_nominalValue ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Adds a global variable to the run options for the Sensitivity analysis run mode in the current or the specified maestro session. If a global variable already exists with the specified name, its value is updated. You can also use this function to change the value of *Temperature*.

Arguments

<code>t_varName</code>	Name of the global variable or "temperature".
<code>g_varValue</code>	Value range for the variable or <i>Temperature</i> .
<code>?nominalValue g_nominalValue</code>	Specifies the nominal value.
<code>?session t_sessionName</code>	Name of the maestro session. If not specified, the current session is used.

Value Returned

<code>t</code>	The variable is set successfully to the run options for the Sensitivity analysis run mode.
<code>nil</code>	Indicates an error.

Examples

Set the values for global variables `rload` and `vdd`.

```
maeSensSetVar("rload" "50 70")  
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
maeSensSetVar("vdd" "1 2 3" ?nominalValue "1")  
=> t
```

Functions for XML File Management

This topic lists the SKILL functions that let you work with XML files saved for maestro cellviews.

sevOpenXmlFile

sevConvertStateFormat

sevCloseXmlFile

sevWriteTable

sevReadTable

sevWriteValue

sevReadValue

sevOpenXmlFile

```
sevOpenXmlFile(  
    t_stateFile  
)  
=> x_stateHandle / nil
```

Description

Opens the specified XML state file. If the file does not exist, this function creates a new file.

Arguments

<i>t_stateFile</i>	XML state file that also includes your home directory.

Value Returned

<i>x_stateHandle</i>	The handle for the specified XML state file is returned. This an integer greater than 0.
<i>nil</i>	Indicates an error.

Examples

Opens the specified XML state file.

```
directory = "/home/user"  
// Specifies the user's directory  
stateFile = strcat(directory "/xml.state")  
stateHandle = sevOpenXmlFile(stateFile)  
// The above function statements returns a handle for the XML state file
```

sevConvertStateFormat

```
sevConvertStateFormat(  
    [ ?libs l_libs ]  
    [ ?stateDir t_stateDir ]  
    [ ?viewType t_viewType ]  
    [ ?format t_format ]  
    [ ?backup g_backup ]  
    [ ?recover g_recover ]  
    [ ?autoCheckin g_autoCheckin ]  
    [ ?popUpWarnWin g_popUpWarnWin ]  
=> t / nil
```

Description

Translates all ADE XL states from normal to XML format in the current directory. This function can also be used to do translate a specific parameter set. You can use this function in the OCEAN script and perform the translation in a batch.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Arguments

<code>?libs l_libs</code>	Library for which the state is to be converted. For example, <code>evaluation</code>, <code>list("evaluation" "tes")</code> or <code>all</code>. If you do not specify a library, all libraries under current directory are searched and converted. For read-only states, a warning message is printed. Note: For ADE XL, the search is based on <code>data.sdb</code>. For state view, search is based on <code>ADE_state.info</code>.
<code>?stateDir t_stateDir</code>	State directories that are not under tool default search directory. It works when <code>viewType</code> is set to <code>stateview</code>. It is used to convert state file for ADE L state.
<code>?viewType t_viewType</code>	Sets the state property. The valid values are <code>adexl</code>, <code>stateview</code>, and <code>all</code>. If no view type is specified, <code>adexl</code> view is used.
<code>?format t_format</code>	Specify the format in which the state is to be converted The possible values are <code>xml</code> and <code>files</code>. The default value is <code>xml</code>.
<code>?backup g_backup</code>	Sets the directory used to backup the original state. The possible values are <code>t</code> and <code>nil</code>. The default value is <code>nil</code>. When this argument is set to <code>t</code>, the function creates a backup library of the directory and files, such as <code>data.sdb</code>, <code>test_states</code>, that are present in the view to be changed.
<code>?recover g_recover</code>	Recovers the state automatically with the latest backup data. When both backup and recover arguments are set to <code>t</code>, backup is ignored. The default value is <code>nil</code>.
<code>?autoCheckin g_autoCheckin</code>	Checks in view automatically under the DM environment. The default value is <code>nil</code>.

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

`?popUpWarnWin g_popUpWarnWin`

Controls the display of the warning message when the state starts getting converted. It can be used in converting state by script.

Value Returned

<code>t</code>	All ADE XL states are translated from normal to XML format in the current directory.
<code>nil</code>	The ADE XL states are not translated.

Examples

Converts all `adexl` views under current directory from the `files` format to `xml` format.

```
sevConvertStateFormat()
```

Converts the `adexl` view under `evaluation` library to `files` format and also creates a backup of original state under directory `/home/user/backup`.

```
sevConvertStateFormat(?libs "evaluation" ?format 'files ?backup  
"/home/user/backup")
```

Recovers the `adexl` view under `evaluation` library from backup directory

```
/home/user/backup
```

```
sevConvertStateFormat(?libs "evaluation" ?format 'files ?backup  
"/home/user/backup" ?recover t)
```

Converts the ADE L state under directory `/home/user/artist_state` to the `xml` format.

```
sevConvertStateFormat(?stateDir "/home/user/artist_state" ?viewType 'stateview )
```

Converts the `adexl` view under `evaluation` library to `xml` format and check in the view automatically under DM environment.

```
sevConvertStateFormat(?libs "evaluation" ?autoCheckin t)
```


sevCloseXmlFile

```
sevCloseXmlFile(  
    t_stateFile  
)
```

Description

Closes the specified XML state handle.

Arguments

<i>t_stateFile</i>	State handle for the XML state file.

Value Returned

None

Examples

Closes the XML state file. This example includes a combination of open and close functions.

```
directory = "/home/user"  
// Specifies the user's directory  
stateFile = strcat(directory "/xml.state")  
stateHandle = sevOpenXmlFile(stateFile)  
// The above function statements returns a handle for the XML state file  
...  
sevCloseXmlFile(stateHandle)  
// closes the given XML state handle.
```

sevWriteTable

```
sevWriteTable(  
    x_stateHandle  
    S_componentName  
    o_table  
    [ ?partition l_partitions ]  
)  
=> t / nil
```

Description

Writes the component table in the specified XML state file.

Arguments

<i>x_stateHandle</i>	The handle of a state file.
<i>S_componentName</i>	Name of the XML component whose value you want to write in the table.
<i>o_table</i>	Table that contains information about the specified component.
<i>?partitions l_partitions</i>	List of partitions.

Value Returned

<i>t</i>	The component table is written to the specified XML state file.
<i>nil</i>	Indicates an error.

Examples

Writes the specified component table in a given XML state file.

```
directory = "/home/user"  
stateFile = strcat(directory "/xml.state")  
stateHandle = sevOpenXmlFile(stateFile)  
// The above commands return the state file handle and open the XML state file.  
table = makeTable('tableformat nil)  
table['a] = list(nil)  
table['b] = "a"  
// The above commands creates the table and specify two arguments a and b of type  
list and string respectively.  
sevWriteTable(stateHandle "test" table)
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
// The above command writes the specified table for the component 'test'.  
=> t
```

When you view the contents of the XML state file, the following output appears:

```
<?xml version="1.0"?>  
<statedb ICVersion="IC6.1.6-64b.ADE.634" version="5">xml  
  <component Name="test" Type="skillTable">tableformat  
    <field Name="b" Type="string">"a"</field>  
    <field Name="a" Type="skillDpl"></field>  
  </component>  
</statedb>  
=> t
```

sevReadTable

```
sevReadTable(  
    x_stateHandle  
    S_componentName  
    o_table  
)  
=> t / nil
```

Description

Returns the component table from the specified XML state file.

Arguments

<i>x_stateHandle</i>	The handle of a state file. This can be any integer value.
<i>S_componentName</i>	Name of the XML component whose value you want to read from the table.
<i>o_table</i>	Table that contains the information about the specified component.

Value Returned

t	The component table is returned from the specified XML state file.
nil	Indicates an error.

Examples

Reads the component table for a given XML state file.

```
stateFile = strcat(directory "/xml.state")  
stateHandle = sevOpenXmlFile(stateFile)  
//The above commands return the handle for the XML state file.  
table = makeTable('tableformat nil)  
//The above command creates a table.  
sevReadTable(stateHandle "analyses" table)  
// The above command reads the value of the component analyses from the given table  
for the specified XML state file handle.
```

sevWriteValue

```
sevWriteValue(  
    x_stateHandle  
    t_componentName  
    t_attributeName  
    t_compValue  
)  
=> t / nil
```

Description

Writes a XML component value in the specified XML state file.

Arguments

<i>x_stateHandle</i>	The handle of a state file. This can be any integer value.
<i>t_componentName</i>	Name of the XML component whose value you want to write.
<i>t_attributeName</i>	Name of the attribute of the specified component.
<i>t_compValue</i>	Value you want to write for the specified component.

Value Returned

<i>t</i>	An XML component value is written in the specified XML state file.
<i>nil</i>	Indicates an error.

Examples

Writes the values of the arguments *a* and *b* of component *test*. A combination of write and read commands are used in this example.

```
directory = "/home/user"  
stateFile = strcat(directory "/xml.state")  
// The above commands return the handle of the XML state file.  
stateHandle = sevOpenXmlFile(stateFile)  
// The above commands opens the XML state file handle.  
sevWriteValue(stateHandle "test" "a" list(nil))  
sevWriteValue(stateHandle "test" "b" list('a'))  
// The above commands write the argument values a and b for component test.  
sevReadValue(stateHandle "test" "b")
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

```
// The above command reads the value for argument b of component test.  
sevCloseXmlFile(stateHandle)  
// The above command closes the XML state file handle.
```

When you view the state file contents, the following output appears:

```
<?xml version="1.0"?>  
<statedb ICVersion="IC6.1.6-64b.ADE.634" version="5">xml  
  <test Name="a" Type="skillDpl"></test>  
  <test Name="b" Type="list">(a)</test>  
</statedb>
```

sevReadValue

```
sevReadValue(  
    x_stateHandle  
    t_componentName  
    t_attributeName  
)  
=> t_componentValue / nil
```

Description

Reads the value of an XML component from the specified XML state file.

Arguments

<i>x_stateHandle</i>	The handle of a state file. This can be any integer value.
<i>t_componentName</i>	Name of the XML component whose value you want to read.
<i>t_attributeName</i>	Name of the attribute of the specified component.

Value Returned

<i>t_componentValue</i>	The value of the specified component is returned.
<i>nil</i>	Indicates an error.

Examples

Reads the values of a component test. A combination of write and read commands are used in this example.

```
directory = "/home/user"  
stateFile = strcat(directory "/xml.state")  
// The above commands return the handle of the XML state file.  
stateHandle = sevOpenXmlFile(stateFile)  
// The above commands opens the XML state file handle.  
sevWriteValue(stateHandle "test" "a" list(nil))  
sevWriteValue(stateHandle "test" "b" list('a'))  
// The above commands write the argument values a and b for component test.  
sevReadValue(stateHandle "test" "b")  
// The above command reads the value for argument b of component test.  
sevCloseXmlFile(stateHandle)  
// The above command closes the XML state file handle.
```

Functions to Work with the Locally Scoped Models and Options (MTS Options)

This topic lists the SKILL functions that let you work with locally scoped models and MTS options.

[maeGetMTSMode](#)

[maeSetMTSMode](#)

[maeGetMTSBlock](#)

[maeSetMTSBlock](#)

maeGetMTSMODE

```
maeGetMTSMODE (
    t_testName
    [ ?session t_sessionName ]
)
=> t / nil
```

Description

Indicates whether the locally scoped models and options (MTS options) are enabled for the specified test.

Arguments

<i>t_testName</i>	Name of the test.
<i>?session t_sessionName</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>t</i>	MTS options are enabled for the specified test.
<i>nil</i>	MTS options are disabled for the specified test or there is an error.

Examples

Indicates that MTS options are enabled for the `test_MTS` test:

```
maeGetMTSMODE ("test_MTS")
=> t
```

maeSetMTSMode

```
maeSetMTSMode (
    t_testName
    [ ?session t_sessionName ]
    [ @rest g_mtsMode ]
)
=> t / nil
```

Description

Enables the locally scoped models and options (MTS options) for the specified test.

Arguments

t_testName Name of the test.

?session t_sessionName
Name of the session.

Note: If you do not specify this argument, the currently active session is used.

@rest g_mtsMode
Boolean flag to specify whether to enable or disable the MTS options.
The default value is *t*.

Value Returned

t The MTS options are enabled for the specified test.

nil Indicates an error.

Examples

Enables the MTS options for the `test_MTS` test.

```
maeSetMTSMode("test_MTS" t)
=> t
```

Enables the MTS options for the `test_MTS` test.

```
maeSetMTSMode("test_MTS")
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Disables the MTS options for the `test_MTS` test.

```
maeSetMTSMode("test_MTS" nil)  
=> t
```

maeGetMTSBlock

```
maeGetMTSBlock(  
    t_testName  
    [ ?lib t_lib ]  
    [ ?cell t_cell ]  
    [ ?inst t_instName ]  
    [ ?type t_type ]  
    [ ?session t_sessionName ]  
)  
=> l_mtsBlockInformation / nil
```

Description

Returns the requested MTS-related information about the specified library cell (block) or instance.

Arguments

<i>t_testName</i>	Name of the test.
<i>?lib t_lib</i>	Name of the library.
<i>?cell t_cell</i>	Name of the cell.
<i>?inst t_instName</i>	Name of the instance.
<i>?type t_type</i>	Type of information you want to print for the specified block or instance. ■ enable: Returns <code>t</code> if local scoping is enabled for the specified block or instance. Returns <code>nil</code> if local scoping is disabled for the specified block or instance. ■ modelFiles: Returns the information related to the models files that are locally scoped for the specified block or instance. ■ simOptions: Returns the process parameters with their values that are included in the simulation locally for the specified block or instance. ■ all: Returns all the above information for the specified block or instance. This is the default.
<i>?session t_sessionName</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Name of the session.

Note: If you do not specify this argument, the currently active session is used.

Value Returned

l_mtsBlockInformation

The requested MTS-related information is returned about the specified library cell (block) or the instance.

nil

The MTS-related information is not returned.

Examples

Returns all the MTS-related information for the library cell, `design_45 inv` in the `test_MTS` test.

```
maeGetMTSBlock("test_MTS" ?lib "design_45" ?cell "inv")
=>(nil modelFiles (
  ("gpdk045/gpdk045_v_3_0/models/spectre/gpdk045.scs" "tt")
) simOptions (
  temp("40")
  scale "5")
  scalem "2")
)
enable t
)
```

Returns the values of the process parameters that are included in the simulation for the library cell, `design_45 inv` in the `test_MTS` test, locally.

```
maeGetMTSBlock("test_MTS" ?lib "design_45" ?cell "inv" ?type "simOptions")
=>(temp("40")
  (scale "5")
  (scalem "2")
)
```

maeSetMTSBlock

```
maeSetMTSBlock(  
    t_testName  
    [ ?lib t_lib ]  
    [ ?cell t_cell ]  
    [ ?inst t_instName ]  
    [ ?enable g_enable ]  
    [ ?modelFiles l_modelFiles ]  
    [ ?simOptions l_simOptions ]  
    [ ?session t_sessionName ]  
    [ ?mode t_mode ]  
)  
=> t / nil
```

Description

Enables local scoping for the specified library cell (block) or instance and sets the specified MTS options. This includes the models files to be scoped locally and the process parameters, such as `scale` and `temp` to be included in the simulation locally.

Arguments

<i>t_testName</i>	Name of the test.
?lib <i>t_lib</i>	Name of the library.
?cell <i>t_cell</i>	Name of the cell.
?inst <i>t_instName</i>	Name of the instance.
?enable <i>g_enable</i>	Boolean flag to specify whether to enable local scoping for the specified block or instance. The default value is <code>t</code> .
?modelFiles <i>l_modelFiles</i>	List specifying the model files that will be locally scoped for the specified block or instance.
?simOptions <i>l_simOptions</i>	List specifying the values of the process parameters that will be included in the simulation for the specified block or instance locally.
?session <i>t_sessionName</i>	

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

Name of the session.

Note: If you do not specify this argument, the currently active session is used.

`?mode t_mode`

Specifies whether to merge or overwrite the existing MTS settings.

- `merge`: Merges the specified MTS settings with the existing settings. This is the default.
- `overwrite`: Removes and overwrites the existing MTS settings, including simulation options and model files.

Value Returned

`t`

Local scoping is enabled for the specified library, cell (block), or instance and the specified MTS options are set.

`nil`

Indicates an error.

Examples

Enables local scoping for the library cell, `design_45 inv` in the `test_MTS` test. Additionally, it specifies that the `gpdk045.scs` model file is locally scoped for the specified library cell and the values of the process parameters, `temp` and `scale` are set to 30 and 5, respectively. These parameters will be included in the simulation for the cell locally.

```
maeSetMTSBlock("test_MTS" ?lib "design 45" ?cell "inv" ?enable t ?modelFiles
list(list("gpdk045.scs" "tt")) ?simOptions list(list("temp" "30") list("scale"
"5")))
=> t
```

Enables local scoping for the instance, `I8` in the `test_MTS` test. Additionally, it specifies that the `gpdk045.scs` model file is locally scoped for the specified instance and the values of the process parameters, `temp` and `scale` are set to 30 and 5, respectively. These parameters will be included in the simulation for the instance locally.

```
maeSetMTSBlock("test_MTS" ?inst "I8" ?enable t ?modelFiles list(list("gpdk045.scs"
"tt")) ?simOptions list(list("temp" "30") list("scale" "5")))
=> t
```

If you have another instance, `I0`, under the `I8` instance in your `test_MTS` test, the following example enables local scoping for the instance, `I0`.

```
maeSetMTSBlock("test_MTS" ?inst "I8/I0" ?enable t ?modelFiles
list(list("gpdk045.scs" "tt")) ?simOptions list(list("temp" "30") list("scale"
"5")))
=> t
```

Virtuoso ADE SKILL Reference

Maestro Cellview Functions

The following example disables local scoping for the library cell, `design_45 inv` in the `test_MTS` test.

```
maeSetMTSBlock("test_MTS" ?lib "design_45" ?cell "inv" ?enable nil )
=> t
```

Enables local scoping at instance `I11` in the test `test_MTS` and adds model files `gpd045_mos.scs` and `gpd045.scs`.

```
models='("./gpd045_mos.scs" "ff") ("./gpd045.scs" "tt")'
maeSetMTSBlock("test_MTS" ?inst "I11" ?enable t ?modelFiles models)
```

Enables local scoping at instance `I11` in the test `test_MTS` and adds model files `gpd045_mos.scs` and `gpd045.scs`. Note that the existing MTS setup for instance `I11` is overwritten.

```
models='("./gpd045_mos.scs" "ff") ("./gpd045.scs" "tt")'
maeSetMTSBlock("test_MTS" ?inst "I11" ?enable t ?modelFiles models ?mode
"overwrite")
```

Disables local scoping and removes the existing MTS setup, including simulation options and model files for instance `I11` in the test `test_MTS`.

```
maeSetMTSBlock("test_MTS" ?inst "I11" ?enable nil ?mode "overwrite")
```

Fault Simulation Functions

Use the following SKILL functions to manage fault simulation runs in ADE Assembler:

getFaultTranData

maeAddFaultRule

maeAddFaultsToFaultGroup

maeAddUndetectedFaultsFromHistoryToGroup

maeChangeTypeFromExprToFaultExpr

maeClearExistingFaultsForRevalidation

maeCreateOrRenameFaultGroup

maeDeleteFaultGroup

maeDeleteFaultRule

maeEditFaultRule

maeGenerateFuSaReport

maeGetDUTForFaults

maeGetFaultGroups

maeGetFaultGroupToRun

maeGetFaultRule

maeGetFaultRules

maeGetFaultRunModeOptions

maeGetFaults

Virtuoso ADE SKILL Reference

Fault Simulation Functions

maeGetFaultSamplingOptions

maeGetGlobalFaultOptions

maeRunFaultSimulationWithFaultDroppingForActiveTests

maeSetDUTForFaults

maeSetDUTForFaults

maeSetFaultAnalysisType

maeSetFaultDFARunModeOptions

maeSetFaultGroupToRun

maeSetFaultSamplingOptions

maeSetFaultTFARunModeOptions

maeSetGlobalFaultOptions

maeSetGlobalFaultOptions

maeGetNumberOfUndetectedFaultsFromHistory

maeMergeFaultHistories

maePrintFaultDroppingStatistics

maeSaveFaultsRunCount

maeSwitchActiveFaultGroupForCurrentRun

maeSetFaultGroupToRun

getFaultTranData

```
getFaultTranData(  
    t_netName | t_terminalName  
    n_faultTime  
)  
=> x_voltage / o_currentWaveform / nil
```

Description

Returns the waveform data for a fault analysis run at the given time.

Arguments

<i>t_netName</i>	Name of a net for which voltage data is to be returned. Default: " "
<i>t_terminalName</i>	Name of a terminal for which current waveform data is to be returned. Default: " "
<i>n_faultTime</i>	Time point in a transient fault analysis for which the data is to be returned. This argument is relevant for a TFA run only. If used with DFA or a regular transient analysis run, fault time is ignored. Default: " "

Value Returned

<i>x_voltage</i>	Returns the voltage data from the simulation result.
<i>o_currentWaveform</i>	Returns a waveform object with current data. A waveform object represents simulation results that can be displayed as a series of points on a grid. A waveform object identifier looks like this: srrWave:XXXXX.
<i>nil</i>	Returns <i>nil</i> and an error message if no value can be returned.

Examples

The following example returns the current waveform for /I1/I2/net5:

```
getFaultTranData("/I1/I2/net5" 1n)
=> srrWave:0x38871020
```

maeAddFaultRule

```
maeAddFaultRule(  
    [ ?session t_sessionName ]  
    [ ?faultType t_faultType ]  
    [ ?resistance t_resistance ]  
    [ ?weightExpr t_weightExpr ]  
    [ ?weightFactor t_weightFactor ]  
    [ ?inst t_instance ]  
    [ ?excludeInst t_excludeInst ]  
    [ ?excludeSubckt t_excludeSubckt ]  
    [ ?faultDevices t_faultDevices ]  
    [ ?pinNames t_pinNames ]  
    [ ?extraOptions t_extraOptions ]  
    [ ?useNetlistSyntax g_useNetlistSyntax ]  
    [ ?enableIEEE2427Mode g_enableIEEE2427Mode ]  
    [ ?enableFaultCollapse g_enableFaultCollapse ]  
)  
=> t / nil
```

Description

Creates a new fault rule with the given set of properties.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

`?faultType t_faultType`

Type of the fault to be inserted.

Possible values:

- "Bridge"
- "Open (Terminal based)"
- "Stuck-at"
- "Custom"

Default value: "Bridge"

`?resistance t_resistance`

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Resistance value for the fault rule. When not specified, the resistance value is taken from the global preferences for faults.

Possible values: A positive numeric value in string format.

`?weightExpr t_weightExpr`

An expression that defines the fault weighting function. When not specified, the weight expression is taken from the global preferences for faults.

Default value is taken from global fault options.

`?weightFactor t_weightFactor`

Weight factor for the fault rule. When not specified, the weight factor is taken from the global preferences for faults.

Possible values: A positive numeric value in string format.

`?inst t_instance`

Name of an instance below which you need to inject faults in design hierarchy.

Default: "*"

`?excludeInst t_excludeInst`

A space-separated list of instance paths to be excluded while injecting faults.

`?excludeSubckt t_excludeSubckt`

A space-separated list of subcircuits to be excluded while injecting faults.

`?faultDevices t_faultDevices`

A space-separated list of fault devices on which you want to inject faults.

Default value: "*"

`?pinNames t_pinNames`

A space-separated list of pin names on which you want to inject faults.

value: "*"

`?extraOptions t_extraOptions`

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Additional info analysis options in the syntax supported by the simulator.

`?useNetlistSyntax` *g_useNetlistSyntax*

A Boolean value that specifies whether to use the netlist format for instance names.

Default value: `nil`

`?enableIEEE2427Mode` *g_enableIEEE2427Mode*

A Boolean value that enables the IEEE 2427 mode in Spectre info analysis.

Default value: `nil`

`?enableFaultCollapse` *g_enableFaultCollapse*

A Boolean value that enables fault collapsing for the fault rule. This option is overridden by the fault collapsing options provided in the global preferences.

Value Returned

`t` A fault rule is successfully created.

`nil` In case of an error.

Example

The following example shows how to add or create a new fault rule with all default values:

```
maeAddFaultRule()  
=> t  
; a new fault rule is added to the maestro session
```

The following example shows how to add or create a new fault rule to insert bridges of resistance 150 ohms in the design hierarchy under instance I3:

```
maeAddFaultRule(?faultType "Bridge" ?weightExpr "w * 2" ?inst "I3" ?resistance  
"150")  
=> t
```

The following example shows how to add a fault rule to insert a stuck-at fault of resistance 100 ohms in the design hierarchy under instance M5:

```
maeAddFaultRule(?faultType "Stuck-at" ?resistance "100" ?inst "M5")
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Related Topics

[Injecting Faults in a Design](#)

maeAddFaultsToFaultGroup

```
maeAddFaultsToFaultGroup(  
    [ ?session t_sessionName ]  
    [ ?faultsName l_faultsName ]  
    [ ?faultGroupsName l_faultGroupsName ]  
)  
=> t / nil
```

Description

Adds the given faults to one or more fault groups.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?faultsName l_faultsName

A list containing the names of the fault rules to be added to the groups.

?faultGroupsName l_faultGroupsName

A list containing the names of the fault groups to which the given faults are to be added.

If no group is found with the given name, a new group is created.

Value Returned

t

Faults are successfully added to the groups.

nil

If the given fault rule is not found in the setup.

Example

The following example shows how to add the fault rules `RB1` and `rule2` to the fault group named `Group1`:

Virtuoso ADE SKILL Reference

Fault Simulation Functions

```
maeAddFaultsToFaultGroup(?faultsName '("RB1" "rule2") ?faultGroupsName  
'("Group1"))  
=>
```

Related Topics

[Injecting Faults in a Design](#)

[Creating Fault Groups](#)

[maeCreateOrRenameFaultGroup](#)

maeAddUndetectedFaultsFromHistoryToGroup

```
maeAddUndetectedFaultsFromHistoryToGroup(  
    [ ?session t_sessionName ]  
    [ ?faultGroupName t_faultGroupName ]  
    [ ?historyName t_historyName ]  
)  
=> t / nil
```

Description

Adds the undetected faults from a history to the specified group in the active setup.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?faultGroupName t_faultGroupName

Name of the fault group in which undetected faults are to be added. If a group is not found with the given name, a new group is created.

?historyName t_historyName

Name of the history from which undetected faults are to be extracted.

Value Returned

t Undetected faults are added to the specified group.

nil If the group name or history is not found in the active setup.

Example

The following example shows how to add undetected faults to a group:

```
maeAddUndetectedFaultsFromHistoryToGroup(?session maeSession ?faultGroupName  
"Group1" ?historyName "FaultSimulation.1.RO")  
=> t
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Related Topics

[Checking Fault Detection Rate](#)

maeChangeTypeFromExprToFaultExpr

```
maeChangeTypeFromExprToFaultExpr(  
    t_sessionName  
    ?historyName t_historyName  
    ?corner t_corner  
)  
=> t / nil
```

Description

Provides an automated way to create fault expressions from the expressions that passed in a Single Run, Sweeps and Corners history.

Arguments

<i>t_sessionName</i>	Name of a session. Default: Current session
<i>?historyName t_historyName</i>	A valid history of the Single Run, Sweeps and Corners run. Default: ""
<i>?corner t_corner</i>	Name of a corner from the given history.

Value Returned

<i>t</i>	All expressions that passed in the given history are changed to fault expressions in the Outputs Setup.
<i>nil</i>	The type of expressions is not changed.

Examples

The following example shows how to convert passed expressions of a history to fault expressions in the outputs setup of the current session:

```
maeChangeTypeFromExprToFaultExpr("fnxSession0" ?historyName "Interactive.0")  
=> t  
  
maeChangeTypeFromExprToFaultExpr("fnxSession0" ?historyName "Interactive.0"  
?corner "C0")  
=> t
```

maeClearExistingFaultsForRevalidation

```
maeClearExistingFaultsForRevalidation(  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Clears the cache for all fault rules that exist in the active fault setup.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

Value Returned

<code>t</code>	The cache is cleared for all fault rules.
<code>nil</code>	In case of an error.

Example

The following example shows how to clear cache for all fault rules that exist in the active fault setup of the current session:

```
maeClearExistingFaultsForRevalidation()  
=> t
```

Related Topics

[Clearing Fault Cache](#)

maeCopyFaultGroups

```
maeCopyFaultGroups (
    [ ?groupNamesList l_groupNames ]
)
=> t / nil
```

Description

Creates a copy of the specified fault groups and saves those with the default names *source-group-name_copy_incremental-num*.

Arguments

?groupNamesList l_groupNames

A list containing the names of fault groups to be copied.

Value Returned

t	When a copy of the specified fault groups are saved.
nil	When the specified fault group is not found,

Example

The following example shows how to create copies of the specified fault groups:

```
maeCopyFaultGroups(?groupNamesList '("Group1" "Group2"))
=> t
; The function creates fault groups with names Group1_copy_0 and Group2_copy_0.
```

Related Topics

[Creating a Fault Group by Copying An Existing Group](#)

[maeCreateOrRenameFaultGroup](#)

[maeDeleteFaultGroup](#)

maeCreateOrRenameFaultGroup

```
maeCreateOrRenameFaultGroup(  
    t_newGroupName  
    [ ?oldGroupName t_oldGroupName ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Creates or renames a group with the specified name in the active fault setup.

Arguments

t_newGroupName

Name of the new group.

Note: This is a mandatory argument. Ensure that this value does not match with any of the existing group names.

?oldGroupName t_oldGroupName

Name of an existing group that you want to rename. Provide this argument only when you want to rename an existing group.

?session t_sessionName

Name of a session.

Default: Current session

Value Returned

t

Creates or renames a group in the active fault setup

nil

If another group with the same name already exists in the setup

Example

The following example shows how to rename an existing group named Group1 to Group2:

```
maeCreateOrRenameFaultGroup("Group2" ?oldGroupName "Group1")  
=t
```


Virtuoso ADE SKILL Reference

Fault Simulation Functions

Related Topics

[Creating Fault Groups](#)

maeDeleteFaultGroup

```
maeDeleteFaultGroup(  
    t_groupName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified fault group from the active fault setup.

Arguments

<i>t_groupName</i>	Name of the existing fault group to be deleted
<i>?session t_sessionName</i>	Name of a session
	Default: Current session

Value Returned

<i>t</i>	Fault group is deleted successfully.
<i>nil</i>	In case of an error.

Example

The following example shows how to delete an existing fault group named Group1:

```
maeDeleteFaultGroup("Group1")  
=> t
```

Related Topics

[Deleting Fault Rules and Fault Groups](#)

maeDeleteFaultRule

```
maeDeleteFaultRule(  
    t_faultRuleName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Deletes the specified fault rule from the active fault setup.

Arguments

<i>t_faultRuleName</i>	Name of the fault rule to be deleted.
<i>?session t_sessionName</i>	Name of a session. Default: Current session

Value Returned

<i>t</i>	When the fault rule is deleted successfully.
<i>nil</i>	In case of an error.

Example

The following example shows how to delete an existing fault rule named RB1:

```
maeDeleteFaultRule("RB1")  
=> t
```

Related Topics

[Deleting Fault Rules and Fault Groups](#)

maeEditFaultRule

```
maeEditFaultRule(  
    t_faultRuleName  
    [ ?session t_sessionName ]  
    [ ?resistance t_resistance ]  
    [ ?weightExpr t_weightExpr ]  
    [ ?weightFactor t_weightFactor ]  
    [ ?inst t_instance ]  
    [ ?excludeInst t_excludeInst ]  
    [ ?excludeSubckt t_excludeSubckt ]  
    [ ?faultDevices t_faultDevices ]  
    [ ?pinNames t_pinNames ]  
    [ ?extraOptions t_extraOptions ]  
    [ ?useNetlistSyntax g_useNetlistSyntax ]  
    [ ?enableIEEE2427Mode g_enableIEEE2427Mode ]  
    [ ?enableFaultCollapse g_enableFaultCollapse ]  
)  
=> t / nil
```

Description

Edits the given fault rule by changing the specified properties. The function updates only the given properties. The values of other properties remain unchanged.

Arguments

t_faultRuleName Name of the fault rule to be edited.

?session *t_sessionName*

Name of a session.

Default: Current session

?resistance *t_resistance*

Resistance value for the fault rule. When not specified, the resistance value is taken from the global preferences for faults.

Possible values: A positive numeric value in string format.

?weightExpr *t_weightExpr*

An expression that defines the fault weighting function. When not specified, the weight expression is taken from the global preferences for faults.

Default value is taken from global fault options.

Virtuoso ADE SKILL Reference

Fault Simulation Functions

`?weightFactor t_weightFactor`

Weight factor for the fault rule. When not specified, the weight factor is taken from the global preferences for faults.

Possible values: A positive numeric value in string format.

`?inst t_instance`

Name of an instance below which you need to inject faults in design hierarchy.

Default: "*"

`?excludeInst t_excludeInst`

A space-separated list of instance paths to be excluded while injecting faults.

`?excludeSubckt t_excludeSubckt`

A space-separated list of subcircuits to be excluded while injecting faults.

`?faultDevices t_faultDevices`

A space-separated list of fault devices on which you want to inject faults.

Default value: "*"

`?pinNames t_pinNames`

A space-separated list of pin names on which you want to inject faults.

value: "*"

`?extraOptions t_extraOptions`

Additional info analysis options in the syntax supported by the simulator.

`?useNetlistSyntax g_useNetlistSyntax`

A Boolean value that specifies whether to use the netlist format for instance names.

Default value: nil

`?enableIEEE2427Mode g_enableIEEE2427Mode`

Virtuoso ADE SKILL Reference

Fault Simulation Functions

A Boolean value that enables the IEEE 2427 mode in Spectre info analysis.

Default value: `nil`

`?enableFaultCollapse` *`g_enableFaultCollapse`*

A Boolean value that enables fault collapsing for the fault rule. This option is overridden by the fault collapsing options provided in the global preferences.

Value Returned

`t` When the specified fault rule is edited successfully.

`nil` In case of an error.

Example

The following example modifies the weight expression, instance name, and resistance values for a fault rule named RB5:

```
maeEditFaultRule("RB5" ?weightExpr "w * 1" ?inst "I3" ?resistance "250")
=> t
```

Related Topics

[Using Fault Rules](#)

[maeAddFaultRule](#)

maeEnableFaults

```
maeEnableFaults(  
    [ ?session t_sessionName ]  
    [ ?groupNamesList l_groupNamesList ]  
    [ ?ruleNamesList l_ruleNamesList ]  
    [ ?enable g_enable ]  
    [ ?enableAll g_enableAll ]  
    [ ?disableAll g_disableAll ]  
)  
=> list(libName cellName viewName) / nil
```

Description

Enables or disables the given fault rules and groups.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?groupNamesList l_groupNamesList

A space-separated list of fault group names that need to be enabled or disabled. It is mandatory to provide at least one of the two arguments, *?groupNamesList* and *?ruleNamesList*.

?ruleNamesList l_ruleNamesList

A space-separated list of fault group names that need to be enabled or disabled. It is mandatory to provide at least one of the two arguments, *?groupNamesList* and *?ruleNamesList*.

?enable g_enable

The enabled or disabled status to be set for the given list of faults or groups.

Default value: t

?enableAll g_enableAll

Virtuoso ADE SKILL Reference

Fault Simulation Functions

A Boolean value that specifies whether to enable all fault rules and groups in the active setup.

Default value: `nil`

You can set either `?enableAll` or `?disableAll` to `t` at the same time.

```
?disableAll t_disableAll
```

A Boolean value that specifies whether to disable all fault rules and groups in the active setup.

Default value: `nil`

Note: You can set either `?enableAll` or `?disableAll` to `t` at the same time.

Value Returned

<code>t</code>	When the specified rules and groups are enabled or disabled, as specified.
<code>nil</code>	In case of an error.

Example

The following example shows how to enable two fault rules and two groups:

```
maeEnableFaults(?ruleNamesList '("RB1" "ROT1") ?groupNamesList '("Group1"
"Group2"))
=> t
```

The following example code disables two fault rules and a group:

```
maeEnableFaults(?ruleNamesList '("RB1" "RB2") ?groupNamesList '("grp1") ?enable
nil)
t
```

Related Topics

[Previewing Faults in a Fault Group](#)

[maeGetFaults](#)

maeGetDUTForFaults

```
maeGetDUTForFaults(  
    [ ?session t_sessionName ]  
)  
=> l_LCVDetails / nil
```

Description

Returns a list containing the library, cell, and view name of the design under test in the active fault setup.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

Value Returned

`l_LCVDetails`

A list containing the library, cell, and view name of the design under test for the fault simulation.

`nil`

In case of an error.

Example

The following example shows how to get the design under test from the fault setup:

```
maeGetDUTForFaults  
=> ("AMP" "pll_lpf_amp_sim" "schematic")
```

Related Topics

[DUT for Faults](#)

maeGetFaultGroups

```
maeGetFaultGroups (
    [ ?session t_sessionName ]
)
=> l_faultGroupNames/ nil
```

Description

Returns a list of fault groups that exist in the active fault setup.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

Value Returned

<i>l_faultGroupNames</i>	A list of fault groups that exist in the active fault setup.
<i>nil</i>	In case of an error.

Example

The following example shows how to get a list of fault groups from the active fault setup:

```
maeGetFaultGroups ()
=> ("Group1" "Group2")
```

Related Topics

[Creating Fault Groups](#)

[maeAddFaultsToFaultGroup](#)

[maeCreateOrRenameFaultGroup](#)

maeGetFaultGroupToRun

```
maeGetFaultGroupToRun(  
    [ ?session t_sessionName ]  
)  
=> t_faultGroupToRun / nil
```

Description

Returns the name of the fault group selected for the fault simulation run. This value is specified by the `faultGroupToRun` parameter.

Arguments

<code>?session t_sessionName</code>	Name of a session.
	Default: Current session

Value Returned

<code>t_faultGroupToRun</code>	Name of the fault group.
<code>nil</code>	In case of an error.

Example

The following example shows how to retrieve the current `faultGroupToRun` parameter:

```
maeGetFaultGroupToRun()  
=> "Rules/Ind. Faults"
```

Related Topics

[Creating Fault Groups](#)

[maeCreateOrRenameFaultGroup](#)

[maeCopyFaultGroups](#)

maeGetFaultRule

```
maeGetFaultRule(  
  [ faultRuleName t_faultRuleName]  
  [ ?session t_sessionName ]  
)  
=> l_faultRule / nil
```

Description

Returns the properties set for the fault rule with the given name.

Arguments

<code>?faultRuleName t_faultRuleName</code>	Name of the fault rule.
<code>?session t_sessionName</code>	Name of a session.
	Default: Current session

Value Returned

<code>l_faultRule</code>	A disembodied property list.
<code>nil</code>	A fault rule with given name is not found.

Example

The following example shows how to return the properties of a fault rule named RB5:

```
maeGetFaultRule("RB5")  
=> (nil faultRuleName "RB5" faultType "Bridge" resistance "250" weightExpr "w * 1"  
weightFactor "1" inst "I3" excludeInst "" excludeSubckt "" faultDevices "NM0"  
pinNames "*" extraOptions "" useNetlistSyntax nil enableIEEE2427Mode nil  
enableFaultCollapse t)
```

Related Topics

Using Fault Rules

maeCreateOrRenameFaultGroup

Virtuoso ADE SKILL Reference

Fault Simulation Functions

maeCopyFaultGroups

maeGetFaultRules

```
maeGetFaultRules(  
    [ ?session t_sessionName ]  
)  
=>l_faultRuleNames / nil
```

Description

Returns a list of the names of fault rules in the fault setup.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

Value Returned

<i>l_faultRuleNames</i>	List of fault rule names.
<i>nil</i>	In case of an error.

Examples

```
maeGetFaultRules()  
=> ("RB1" "RB2" "RB3" "RB4" "RB5" "RB6" "ROT1" )
```

Related Topics

[Using Fault Rules](#)

[maeAddFaultRule](#)

maeGetFaultRunModeOptions

```
maeGetFaultRunModeOptions(  
    [ ?session t_sessionName ]  
    [ ?faultAnalysisType t_faultAnalysisType ]  
)  
=> l_runModeOptions / nil
```

Description

Retrieves the run options for the fault simulation.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?faultAnalysisType t_faultAnalysisType

Type of fault analysis for which the run options are to be retrieved. By default, returns data for the currently selected analysis type.

Possible Values:

- "DFA"
- "TFA"

Value Returned

l_runModeOptions List of options set for the specified or the currently selected analysis type.

nil In case of an error.

Example

The following example shows how to retrieve the run mode options:

```
maeGetFaultRunModeOptions(?faultAnalysisType "DFA")  
=> (nil analysisType "DFA" runNominal t)
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Related Topics

[Setting up Run Options for a Fault Simulation](#)

[maeSetFaultDFARunModeOptions](#)

[maeSetFaultTFARunModeOptions](#)

maeGetFaults

```
maeGetFaults(  
    [ ?session t_sessionName ]  
    [ ?activeGroupToRun g_activeGroupToRun ]  
    [ ?faultRuleName t_faultRuleName ]  
    [ ?groupName t_groupname ]  
)  
=> t_faultsFileName / nil
```

Description

Runs Spectre *info* analysis to generate the faults for the specified fault group or fault rule. The generated faults are saved in a file. To view the generated faults, you can either open the file or use the *Preview Faults* commands in the Fault Setup assistant.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?activeGroupToRun g_activeGroupToRun

A Boolean value that specifies whether to get faults for the currently selected group.

Possible values: *t* and *nil*

Default value: *t*

If *g_activeGroupToRun* is set to *nil*, either *t_faultRuleName* or *t_groupName* is required.

?faultRuleName t_faultRuleName

Name of an existing fault rule for which faults need to be generated.

?groupName t_groupName

Name of an existing group for which faults need to be generated.

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Value Returned

<i>t_faultsFileName</i>	The path to the fault file containing the generated faults.
<i>nil</i>	If the specified fault group is not found.

Example

The following example shows how to get faults for the group RB1:

```
maeGetFaults(?faultRuleName "RB1")
```

```
=> INFO (ADE-3071): Simulation completed successfully.  
reading simulation data...  
...successful.
```

To view the faults generated, open
/usePath/simulation/libName/cellName/viewName/results/maestro/.tmpADEDir_username/
DCGain/libName_cellName_viewName_spectre/faultInfo/psf/ruleFaults.scs file.

Related Topics

[Previewing Faults in a Fault Group](#)

maeGetFaultSamplingOptions

```
maeGetFaultSamplingOptions(  
    [ ?session t_sessionName ]  
)  
=> l_samplingOptions / nil
```

Description

Returns the sampling options set for fault simulation.

Arguments

<code>?session t_sessionName</code>	Name of a session
	Default: Current session

Value Returned

<code>l_samplingOptions</code>	A disembodied property list of fault sampling options and their values.
<code>nil</code>	When the given session is not found.

Examples

The following example shows how this function returns fault sampling options:

```
maeGetFaultSamplingOptions()  
(nil enableSampling t_samplingMethod "randomweighted" sampleRatio "50" seedValue  
"1" confidenceValue 97.0 )
```

Related Topics

[Specifying Fault Sampling Options](#)

maeGetGlobalFaultOptions

```
maeGetGlobalFaultOptions(  
    [ ?session t_sessionName ]  
)=> l_globalOptions / nil
```

Description

Returns the global fault options set in the Virtuoso ADE Fault Setup Global Preferences form. These options are applicable to all faults unless overridden in individual faults or fault rules.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

Value Returned

l_globalOptions A disembodied property list of global fault options.

nil When the given session is not found.

Example

The following example shows how to return global fault options set in the currently active session:

```
maeGetGlobalFaultOptions()  
=> (nil defaultBridgeResistance "100" defaultOpenResistance "10K"  
defaultStuckAtResistance "100" defaultWeightExpr "1" defaultWeightFactor "1"  
defaultIndividualFaultWeight "1" stuckAtGround "/vin" stuckAtSupply "/vss"  
applyFaultCollapsing "All Fault rules")
```

Related Topics

[Specifying Global Preferences for Fault Setup](#)

maeRunFaultSimulationWithFaultDroppingForActiveTests

```
maeRunFaultSimulationWithFaultDroppingForActiveTests(  
    [ ?session t_sessionName ]  
)  
=> t_historyName / nil
```

Description

Runs the fault dropping flow in batch mode for the enabled tests in the given session.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

Value Returned

`t` Name of the history in which the results of fault simulation are saved.

`nil` When the simulation is not run successfully.

Examples

The following example code runs the fault simulation with fault dropping enabled:

```
maeRunFaultSimulationWithFaultDroppingForActiveTests()  
=> "FaultSimulation.1"
```

Related Topics

[Automatic Fault Dropping Using a Run Plan](#)

maeSetDUTForFaults

```
maeSetDUTForFaults(  
    t_libName  
    t_cellName  
    t_viewName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets the specified design under test for the fault setup.

Arguments

<i>t_libName</i>	Name of a library.
<i>t_cellName</i>	Name of a cell.
<i>t_viewName</i>	Name of a view.
<i>?session t_sessionName</i>	Name of a session. Default: Current session

Value Returned

<i>t</i>	The design is successfully set for the fault setup.
<i>nil</i>	The specified design is not found.

Examples

The following example shows how to set the design for fault simulations:

```
maeSetDUTForFaults("Two_Stage_Opamp" "DualAmp" "schematic")  
=> t
```

Related Topics

Setting up Faults

maeSetFaultAnalysisType

```
maeSetFaultAnalysisType(  
    [ ?analysisType t_analysisType ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets the analysis type for the fault simulation run mode.

Arguments

?analysisType t_analysisType

Name of the analysis type.

Possible values:

- "DFA"
- "TFA"

?session t_sessionName

Name of a session.

Default: Current session

Value Returned

<i>t</i>	The analysis type for fault simulation is updated successfully.
<i>nil</i>	The analysis type or the maestro session is invalid.

Examples

The following example explains how to set the analysis type for the fault simulation run mode:

```
maeSetFaultAnalysisType("DFA")  
=> t
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Related Topics

[Setting up Run Options for a Fault Simulation](#)

[maeSetFaultDFARunModeOptions](#)

[maeSetFaultTFARunModeOptions](#)

maeSetFaultDFARunModeOptions

```
maeSetFaultDFARunModeOptions(  
    [ ?session t_sessionName ]  
    [ ?extraOptions t_extraOptions ]  
    [ ?runNominal g_runNominal ]  
)  
=> t / nil
```

Description

Sets the run mode options for the Direct Fault Analysis (DFA) to be run for fault simulation.

Arguments

?session t_sessionName

Name of a session

Default: Current session

?extraOptions t_extraOptions

Specifies the options other than *?runNominal*. These options are directly passed to Spectre. You must ensure the correctness of the names and values of these options.

Default Value: " "

?runNominal g_runNominal

A Boolean value that specifies whether to run simulation for nominal or not. Possible values are *t* and *nil*

Default Value: *nil*

Value Returned

t Run mode options are successfully set for Direct Fault Analysis.

nil When invalid values are provided for the options.

Examples

The following example explains how to set run mode options for DFA in the fault setup:

Virtuoso ADE SKILL Reference

Fault Simulation Functions

```
maeSetFaultDFARunModeOptions(?extraOptions "faultduplicate=yes" ?runNominal t)  
=> t
```

Related Topics

[Setting up Run Options for a Fault Simulation](#)

[maeSetFaultTFARunModeOptions](#)

maeSetFaultSamplingOptions

```
maeSetFaultSamplingOptions(  
    [ ?session t_sessionName ]  
    [ ?enableSampling g_enableSampling ]  
    [ ?samplingMethod t_samplingMethod ]  
    [ ?sampleByNumOrRatio t_sampleByNumOrRatio ]  
    [ ?sampleNum t_sampleNum ]  
    [ ?sampleRatio t_sampleRatio ]  
    [ ?seedValue t_seedValue ]  
    [ ?confidenceValue t_confidenceValue ]  
)  
=> t / nil
```

Description

Sets the sampling options for the fault setup.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?enableSampling g_enableSampling

A Boolean value that specifies whether to enable sampling or not. Possible values are *t* and *nil*.

Default value: *t*

?samplingMethod t_samplingMethod

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Sampling method.

Possible values:

- "random"
- "randomuniform"
- "weightsorted"
- "randomweighted"

Default value: "random"

For details on these sampling methods, see [Fault Selection and Sampling](#) in *Spectre Classic Simulator*, *Spectre APS*, *Spectre X*, and *Spectre XPS User Guide*.

`?sampleByNumOrRatio t_sampleByNumOrRatio`

Specifies how you want to choose samples.

Possible values:

- "Number": Uses a specific number of faults in each sample.
- "Ratio": Uses a percentage or ratio of the possible samples.

Default value: "Number"

`?sampleNum t_sampleNum`

Number of samples to be created.

Default value: "1"

Note: Specify this argument only when `?sampleByNumOrRatio` is set to "Number" and `?sampleRatio` is not specified.

`?sampleRatio t_sampleRatio`

Virtuoso ADE SKILL Reference

Fault Simulation Functions

The ratio of the possible samples to be created. Specify a value between 0 and 100.

Default value: "1"

Note: Specify this argument only when `?sampleByNumOrRatio` is set to "Ratio" and `?sampleNum` is not specified.

`?seedValue t_seedValue`

A seed value greater than 0.

Default value: "1"

`?confidenceValue t_confidenceValue`

Value for confidence interval between 0 and 100.

Default value: "97.8"

Value Returned

`t` Sampling options are updated successfully.

`nil` In case of an error.

Examples

The following example explains how to set sampling options for the current fault setup:

```
maeSetFaultSamplingOptions( ?enableSampling t ?samplingMethod "random"  
?sampleByNumOrRatio "Ratio" ?sampleRatio "2" ?seedValue "12")  
=> t
```

Related Topics

[Specifying Fault Sampling Options](#)

maeSetFaultTFARunModeOptions

```
maeSetFaultTFARunModeOptions (  
    [ ?session t_sessionName ]  
    [ ?extraOptions t_extraOptions ]  
    [ ?simulationMethod t_simulationMethod ]  
    [ ?maxIterations t_maxIterations ]  
    [ ?leadTime t_leadTime ]  
    [ ?faultPointsMethod t_faultPointsMethod ]  
    [ ?faultTimePoints g_faultTimePoints ]  
    [ ?start_step_stop l_start_step_stop ]  
)  
=> t / nil
```

Description

Sets the run mode options for Transient Fault Analysis to be run for fault simulation.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

`?extraOptions t_extraOptions`

Extra options for Transient Fault Analysis to be passed to Spectre directly.

Default value: " "

`?simulationMethod t_simulationMethod`

The simulation method for Transient Fault Analysis.

Possible values:

- "linear" (default)
- "onestep"
- "testpoint"
- "timezero"
- "maxiters"
- "leadtime"

Virtuoso ADE SKILL Reference

Fault Simulation Functions

?maxIterations *t_maxIterations*

The maximum number of iterations greater than 0. Set this argument only when the simulation method is "maxiters".

Default value: "50"

?leadTime *t_leadTime*

The lead time value greater than 0. Set this argument only when the simulation method is "leadtime".

Default value: "50u"

?faultTimePointsMethod *t_faultTimePointsMethod*

The method to set fault points.

Possible values:

- "Start/Step/Stop"
- "Fault Time Points"

Default value: ""

?faultTimePoints *g_faultTimePoints*

A string or list of strings to be set when ?faultTimePointsMethod is set to "Fault Time Points".

Default value: ""

?start_step_stop *l_start_step_stop*

A list of three numeric values greater than 0 in string format to be used as start, step, and stop values when ?faultTimePointsMethod is set to "Start/Step/Stop" .

Note: The total of start and step time should be less than the stop time.

Default value: A blank list.

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Value Returned

<code>t</code>	Run mode options for Transient Fault Analysis are set successfully.
<code>nil</code>	Returns <code>nil</code> if there is an error.

Examples

The following example code sets up the run mode options for Transient Fault Analysis in the fault setup:

```
maeSetFaultTFARunModeOptions(?simulationMethod "maxiters" ?maxIterations "50"  
?faultPointsMethod "Start/Step/Stop" ?start_step_stop '("3" "4" "11"))  
=> t
```

Related Topics

[Run Options for Transient Fault Analysis](#)

[maeSetFaultDFARunModeOptions](#)

Virtuoso ADE SKILL Reference

Fault Simulation Functions

maeSetGlobalFaultOptions

```
maeSetGlobalFaultOptions (  
    [ ?session t_sessionName ]  
    [ ?defaultBridgeResistance t_defaultBridgeResistance ]  
    [ ?defaultOpenResistance t_defaultOpenResistance ]  
    [ ?defaultStuckAtResistance t_defaultStuckAtResistance ]  
    [ ?stuckAtGround t_stuckAtGround ]  
    [ ?stuckAtSupply t_stuckAtSupply ]  
    [ ?defaultWeightExpr t_defaultWeightExpr ]  
    [ ?defaultWeightFactor t_defaultWeightFactor ]  
    [ ?defaultIndividualFaultWeight t_defaultIndividualFaultWeight ]  
    [ ?applyFaultCollapsing t_applyFaultCollapsing ]  
)  
=> t / nil
```

Description

Sets global preferences for fault simulations.

Arguments

?session t_sessionName

Name of a session.

Default: Current session

?defaultBridgeResistance t_defaultBridgeResistance

Default resistance value greater than 0 for bridge faults.

Default value: "100"

?defaultOpenResistance t_defaultOpenResistance

Default resistance value greater than 0 for open faults.

Default value: "10K"

?defaultStuckAtResistance t_defaultStuckAtResistance

Default resistance value greater than 0 for stuck-at faults.

Default value: "100"

?stuckAtGround t_stuckAtGround

Stuck at ground node for stuck-at faults.

Default value: ""

Virtuoso ADE SKILL Reference

Fault Simulation Functions

`?stuckAtSupply t_stuckAtSupply`

Stuck at supply node for stuck-at faults.

Default value: ""

`?defaultWeightExpr t_defaultWeightExpr`

Default weight expression for all fault rules.

Default value: "1"

`?defaultWeightFactor t_defaultWeightFactor`

Default weight factor greater than 0 for all fault rules.

Default value: "1"

`?defaultIndividualFaultWeight t_defaultIndividualFaultWeight`

Default weight greater than 0 for individual faults.

Default value: "1"

`?applyFaultCollapsing t_applyFaultCollapsing`

Fault collapsing policy for fault rules.

Possible values:

- "All Fault rules"
- "Rule Specific"
- "None"

Default value: "All Fault rules"

Value Returned

`t` Global preferences for fault simulation are set successfully.

`nil` An invalid value is specified for a global option or the given session is not found.

Example

The following example shows how to set the global fault options:

```
maeSetGlobalFaultOptions(?defaultBridgeResistance "100" ?defaultOpenResistance  
"10K" ?defaultStuckAtResistance "100" ?stuckAtGround "/vin" ?stuckAtSupply "/vss")
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

```
?defaultWeightExpr "1" ?defaultWeightFactor "1" ?defaultIndividualFaultWeight "1"  
?applyFaultCollapsing "All Fault rules")
```

=> t

Related Topics

[Specifying Global Preferences for Fault Setup](#)

maeGenerateFuSaReport

```
maeGenerateFuSaReport(  
    [ ?session t_sessionName ]  
    [ ?historyName t_historyName ]  
    [ ?outFile t_outFile ]  
    [ ?checkerExprList l_checkerExprList ]  
    [ ?mode t_mode ]  
    [ ?modeID x_modeID ]  
    [ ?confidenceLevel n_confidenceLevel ]  
    [ ?viewFuSaReport g_viewFuSaReportValue ]  
)  
=> t / nil
```

Description

Generates a functional safety report for a fault simulation run.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

`?historyName t_historyName`

Name of the history for which you want to generate the report.

Default: " ", which implies that results from the last run fault simulation are used for the report.

`?outFile t_outFile` Path of the file in which the report is to be written.

Default value: ". /FuSARepor.txt"

`?checkerExprList l_checkerExprList`

List of the names of the checker expressions to be used to check the output of the checker blocks.

Default value is " ". You must specify at least one expression.

Virtuoso ADE SKILL Reference

Fault Simulation Functions

<code>?mode <i>t_mode</i></code>	The failure mode to be printed in the report. ■ SPF: Single Point Failure ■ LFC: Latent Fault Continuous ■ LFP: Latent Fault Periodic Default value is " ". You must specify a valid mode to run the function successfully.
<code>?modeID <i>x_modeID</i></code>	An integer ID of the failure mode being addressed with this report. Default value: 0
<code>?confidenceLevel <i>n_confidenceLevel</i></code>	A confidence level, in percentage, to show the corresponding confidence intervals in the fault detection matrix. Default value is 0. You must specify a positive floating value to run the function successfully.
<code>?viewFusaReport <i>g_viewFusaReportValue</i></code>	Specifies whether to open the report after it is created. Default value is <code>nil</code>.

Value Returned

<code>t</code>	A functional safety report is generated successfully.
<code>nil</code>	The functional safety report was not generated or could not be saved at the given file path.

Example

The following example shows how to generate a functional safety report:

```
maeGenerateFuSaReport( ?outFile "./report1.txt" ?checkerExprList  
  ("problemDetected2") ?mode "LFP" ?confidenceLevel 95 )
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

=>t
; Saves the following report:

Related Topics

[Running Functional Safety Checks](#)

maeGetNumberOfUndetectedFaultsFromHistory

```
maeGetNumberOfUndetectedFaultsFromHistory(  
    [ ?session t_sessionName ]  
    [ ?historyName t_historyName ]  
)  
=> l_UDFaults / nil
```

Description

Returns the list of undetected faults for the history provided.

Arguments

`?session t_sessionName`

Name of a session.

Default: Current session

`?historyName t_historyName`

History name for which list of undetected faults is to be retrieved.

Value Returned

`l_UDFaults`

List of undetected faults for the provided history name.

`nil`

No undetected faults present.

Example

The following example shows how to return the list of undetected faults:

```
maeGetNumberOfUndetectedFaultsFromHistory(?historyName "FaultSimulation.1")  
=> ("ROTr1_open_1" "ROTr1_open_2" "ROTr1_open_3" "ROTr1_open_4")
```

Related Topics

[Checking Fault Detection Rate](#)

maeMergeFaultHistories

```
maeMergeFaultHistories(  
    [ ?session t_sessionName ]  
    [ ?historiesToMerge l_historiesToMerge ]  
    [ ?finalHistoryName t_finalHistoryName ]  
    [ ?logFile t_logFile ]  
)  
=> t/ nil
```

Description

Merges fault histories into one.

Arguments

`?session t_sessionName`

Name of the session

Default: Current session

`?historiesToMerge l_historiesToMerge`

List of histories that are to be merged.

`?finalHistoryName t_finalHistoryName`

History name in which all the histories are to be merged.

`?logFile t_logFile` A file to print the logs of merging. The default file name is faultDropping.logs.

Note: Prints are disabled for now.

Value Returned

`t` Fault histories are merged.

`nil` Merging failed with a relevant error message.

Example

The following example shows how to merge fault histories:

Virtuoso ADE SKILL Reference

Fault Simulation Functions

```
maeMergeFaultHistories(?historiesToMerge '("Plan.0.Run.0" "Plan.0.Run.1"  
"Plan.0.Run.2" "Plan.0.Run.3") ?finalHistoryName "Plan.0.Run.3" ?logFile  
"_faultDropping.logs")  
=> t
```

Related Topics

[Checking Fault Detection Rate](#)

maePrintFaultDroppingStatistics

```
maePrintFaultDroppingStatistics(  
    [ ?session t_sessionName ]  
    [ ?historiesName l_historiesName ]  
    [ ?logFile t_logFile ]  
)  
=> t
```

Description

Prints the formatted results obtained after execution of the fault dropping flow. It also writes the same to the log file in the current work directory. This is exclusive for the fault dropping flow.

Arguments

`?session t_sessionName`

Name of the session

Default: Current session

`?historiesName l_historiesName`

Name of the history containing final results post fault dropping flow is completed.

`?logFile t_logFile`

Name of the log file to be written in the current work directory. The default file name is `_faultDropping.logs`.

Value Returned

`t` Formatted results are printed.

Examples

The following example explains how to print the formatted results:

```
maePrintFaultDroppingStatistics( ?historyName "Plan.2.Run.7" ?logFile  
    "_faultDropping.logs")  
=>
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

```
Fault Dropping : Sep 11 12:37:15 2020 : -----Fault Dropping Run
Summary-----
Fault Dropping : Sep 11 12:37:15 2020 : Number of expected simulations to run was
12.
Fault Dropping : Sep 11 12:37:15 2020 : Number of simulations saved is 6.
Fault Dropping : Sep 11 12:37:15 2020 : Number of corners run in the setup is 0.
Fault Dropping : Sep 11 12:37:15 2020 : Cumulative coverage is 100.0.
Fault Dropping : Sep 11 12:37:15 2020 : Cumulative weighted coverage is 100.0.
Fault Dropping : Sep 11 12:37:15 2020 : Fault simulation has been run for all active
tests. Results available in Plan.2.Run.7.
Fault Dropping : Sep 11 12:37:15 2020 : -----Fault Dropping Run Flow
Completed-----
t
```

Related Topics

[Automatic Fault Dropping Using a Run Plan](#)

maeSaveFaultsRunCount

```
maeSaveFaultsRunCount(  
    [ ?session t_sessionName ]  
    [ ?faultsCount x_faultsCount ]  
)  
=> x_faultsCount
```

Description

Saves the faults count globally for further use. If a fault count already exists, it adds the provided value to the existing value. This is exclusive for the fault dropping flow.

Arguments

`?session t_sessionName`

Name of the session

Default: Current session

`?faultsCount x_faultsCount`

Number of faults to be saved.

Value Returned

`x_faultsCount`, Final faults count saved.

Examples

The following example explains how to save the fault counts:

```
maeSaveFaultsRunCount(?faultsCount 100)  
=> 108
```

Related Topics

[Checking Fault Detection Rate](#)

maeSwitchActiveFaultGroupForCurrentRun

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

Description

Sets the fault group to run for the current run in run plan. This is exclusive to the fault dropping flow.

Arguments

<i>t_historyName</i>	Fault group to run for the current run.

Value Returned

<i>t</i>	Fault group to run is set.
<i>nil</i>	Fault group to run couldn't be set with a relevant error message.

Examples

The following example explains how to set the fault group to run for the current run:

```
maeSwitchActiveFaultGroupForCurrentRun("Plan.0.Run.2")  
=> t
```

Related Topics

[Checking Fault Detection Rate](#)

maeSetFaultGroupToRun

```
maeSetFaultGroupToRun(  
    [ ?session t_sessionName ]  
    [ ?faultGroupToRun t_faultGroupToRun ]  
)  
=> t / nil
```

Description

Sets the fault group to run for the active fault setup.

Arguments

?session t_sessionName

Name of a session

Default: Current session

?faultGroupToRun t_faultGroupToRun

Specify the group to set as the active group to run fault simulation.

Possible values: "Rules/Ind. Faults" or any other valid group name.

Default Value: "Rules/Ind. Faults"

Value Returned

t The `faultGroupToRun` parameter is updated.

nil Returns `nil` if invalid *t_faultGroupToRun* is specified.

Examples

The following example explains how to set the `faultGroupToRun` parameter for the active fault setup:

```
Example: maeSetFaultGroupToRun(?faultGroupToRun "Group1")  
=> t
```

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Related Topics

[Creating Fault Groups](#)

Virtuoso ADE SKILL Reference

Fault Simulation Functions

Session Functions

Every instance of ADE Explorer and ADE Assembler has a session associated with it. If these applications are opened in the UI, you can access the session by using the `axlGetWindowSession` function. It returns the session of the active ADE window.

If you are using a SKILL script in the non-GUI mode, you need to create a new session by using the `maeOpenSetup` for a maestro cellview and `axlCreateSession` for an adexl cellview.

By using the handle to the ADE session, you can access the existing setup database or create a new database.

This topic lists the SKILL functions that let you work with ADE sessions.

[axlCloseSession](#)

[axlCloseSessionInWindow](#)

[axlCreateSession](#)

[axlGetMainSetupDB](#)

[axlGetSessionCellName](#)

[axlGetSessionLibName](#)

[axlGetSessionViewName](#)

[axlGetSessionWindowNumber](#)

[axlGetToolSession](#)

[axlGetWindowSession](#)

[axlGetCurrentResultSimulationHost](#)

[axlIsSessionReadOnly](#)

Virtuoso ADE SKILL Reference

Session Functions

[axlIsValidAXLSession](#)

[axlMainAppSaveSetup](#)

[axlNoSession](#)

[axlRemoveSetupState](#)

[axlSaveSetupState](#)

[axlSessionConnect](#)

[axlSessionDisconnect](#)

[axlSessionRegisterCreationCallback](#)

[axlSessionSignalList](#)

[axlSessionSignalSignature](#)

[axlSetMainSetupDB](#)

[axlSetMainSetupDBLCV](#)

[axlSetupStates](#)

[axlSuppressPersistedQuestionDialog](#)

[axlShowPersistedQuestionDialog](#)

[axlToolSetOpPointInfo](#)

Related Topics

[maeOpenSetup](#)

[Working with Signals or Triggers](#)

[Signals or Triggers for ADE Sessions](#)

axlCloseSession

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

Description

Closes the specified session.

Arguments

<i>t_session</i>	Name of the ADE session you want to close.

Value Returned

<i>t</i>	The specified session is closed.
<i>nil</i>	The specified session is not closed.

Examples

Closes the session `session0`.

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

Virtuoso ADE SKILL Reference

Session Functions

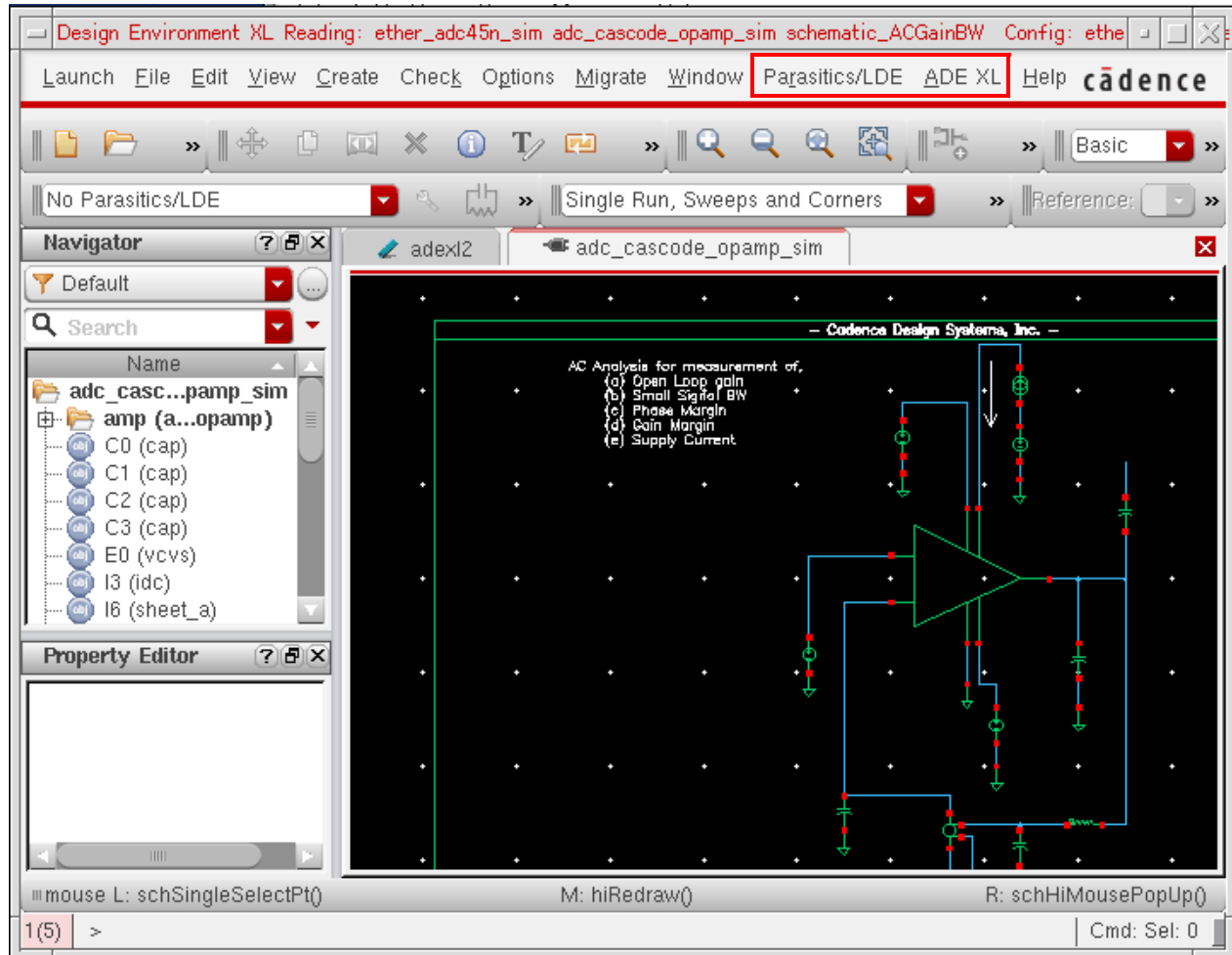
axlCloseSessionInWindow

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

Description

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

When you open a schematic or a layout in a new tab in a window for a maestro cellview, the menus for that maestro cellview are also displayed on the schematic or layout tab, as highlighted in the figure shown below.



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

Virtuoso ADE SKILL Reference

Session Functions

Arguments

<i>w_window</i>	The window ID of the schematic or layout tab that has associated maestro cellview. If not specified, the assistants for the maestro cellview are closed from the current window.

Value Returned

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

Examples

Closes the menus and assistants from the current window.

```
axlCloseSessionInWindow( )  
=> t
```

axlCreateSession

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

Description

Creates a new session with the specified name. You can use this function to create a new session before running a SKILL code for a maestro cellview in the non-GUI mode.

When running a script in the standalone mode, after creating a new session, you can set up a database for the session by using the [axlSetMainSetupDBLCV](#) function.

Note: If the GUI is already open and you need to run SKILL commands from the CIW, you do not need to create a new session. Instead, you can only get a handle to the already open session by using the [axlGetWindowSession](#) function.

Arguments

<i>t_session</i>	Name to be used for the new ADE session.

Value Returned

<i>t_sessionName</i>	The name of the session is returned if the session is successfully created.
<i>nil</i>	The name of the session is not returned.

Examples

Creates a new session with the name `mysession`.

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

Creates a new session with a random number in the session name.

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

Creates a new session and add an output expression:

Virtuoso ADE SKILL Reference

Session Functions

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

Related Topics

[axlGetWindowSession](#)

[axlSetMainSetupDBLCV](#)

axlGetMainSetupDB

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

Description

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

Arguments

<i>t_session</i>	ADE Session name.

Value Returned

<i>x_mainSDB</i>	Handle to the setup database.
<i>nil</i>	Unsuccessful operation.

Examples

Returns a handle to the setup database associated with the current session:

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

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

axlGetSessionCellName

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

Description

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

Arguments

<i>g_value</i>	Name of the ADE session or handle to the setup database.

Value Returned

<i>t_cellName</i>	Name of the cell corresponding to the given session name or setup database is returned.
<i>nil</i>	Indicates an error.

Examples

Returns the cell name associated with a session by using the session name.

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

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

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

Related Topics

[axlGetMainSetupDB](#)

[axlGetWindowSession](#)

axlGetSessionLibName

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

Description

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

Arguments

<i>g_value</i>	Name of the ADE session or handle to the setup database.

Value Returned

<i>t_libName</i>	Name of the library corresponding to the given session name.
<i>nil</i>	Unsuccessful operation.

Examples

Returns the library name associated with a session by using the ADE session name.

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

Returns the library name associated with a session by using the handle to database of that session.

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

Related Topics

[axlGetMainSetupDB](#)

[axlGetWindowSession](#)

axlGetSessionViewName

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

Description

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

Arguments

<i>g_value</i>	Name of the ADE session or handle to the setup database.

Value Returned

<i>t_viewName</i>	Name of the view corresponding to the given session name is returned.
<i>nil</i>	Indicates an error.

Examples

Returns the view name associated with a session by using the name of that session.

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

Returns the view name associated with a session by using the handle to database of that session.

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

Related Topics

[axlGetMainSetupDB](#)

[axlGetWindowSession](#)

axlGetSessionWindowNumber

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

Description

Returns a unique integer representing a number corresponding to a given session name.

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

Arguments

<i>t_sessionName</i>	Name of the ADE session.

Value Returned

<i>x_number</i>	The unique integer representing a number corresponding to the given session name is returned.
<i>nil</i>	Indicates an error.

Examples

Returns the session ID for the session corresponding to the current window.

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

Related Topics

[axlGetWindowSession](#)

axlGetToolSession

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

Description

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

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

Arguments

<i>t_sessionName</i>	Name of the ADE session.
<i>t_testName</i>	Name of the test.
<i>?history x_history</i>	ID of the history. Use this argument when you want to work on the given history run of the test. This history must be opened before using this function. If the specified history is not open, <code>axlGetToolSession</code> returns <code>nil</code> . To open a history, use axlViewHistoryResults .

Value Returned

<i>g_sessionID</i>	Returns a unique in-memory ID.
<i>nil</i>	Unsuccessful operation.

Examples

Adds a model library file and section for a test in an ADE Assembler session. The script gets access to the test session identifier, `testsess`, by using the `axlGetToolSession` function. It

Virtuoso ADE SKILL Reference

Session Functions

further uses `testsess` to remove all the model library selections for that test and adds new model file sections.

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

Uses the history for a particular test to get access to the unique identifier for a test in that history. You can use this identifier to further work with the OASIS session.

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

axlGetWindowSession

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

Description

Returns the ADE session associated with a window.

There is a session object associated with each instance of ADE Explorer and ADE Assembler. If you have a session open, you can retrieve that session by using this function.

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

Arguments

<i>w_window</i>	Window ID

Value Returned

<i>t_sessionName</i>	Returns the ADE session name.
<i>nil</i>	Unsuccessful operation.

Examples

Returns the ADE session of the active window:

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

axlGetCurrentResultSimulationHost

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

Description

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

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

Arguments

<i>t_sessionName</i>	Specifies the name of the ADE session from which the simulation was run.

Value Returned

<i>t_hostName</i>	Returns the host name corresponding to the specified session.

Examples

Prints the simulation host name.

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


axlIsSessionReadOnly

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

Description

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

Arguments

<i>t_axlSession</i>	Specifies the name of the ADE session.

Value Returned

<i>t</i>	The specified session is opened in the read-only mode.
<i>nil</i>	The specified session is opened in the edit mode.

Examples

Indicates that `session0` is opened in the read-only mode.

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

axlIsValidAXLSession

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

Description

Checks the validity of the specified session.

Arguments

<i>t_session</i>	Name of the ADE session to be validated.

Value Returned

<i>t</i>	The specified session exists.
<i>nil</i>	The specified session does not exist.

Examples

Indicates that the specified session, *session1*, is valid.

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

axlMainAppSaveSetup

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

Description

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

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

Arguments

<i>t_sessionName</i>	Name of the session.

Value Returned

<i>t</i>	The ADE state and setup database information associated with the given session is saved.
<i>nil</i>	The ADE state and setup database information associated with the given session is not saved.

Examples

Creates a new session, adds a new variable, and saves the details from the currently active session to the disk.

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

axlNoSession

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

Description

Returns *t* if there is no session in the current window.

Arguments

<i>w_window</i>	Name of the window.

Value Returned

<i>t</i>	There is no session in the current window.
<i>nil</i>	There is a session in the current window.

Examples

Specifies if there is any active session in the current window.

```
axlNoSession( )  
=> t
```

axlRemoveSetupState

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

Description

Removes the specified setup state for the given session.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>t_stateName</i>	Name of the state.

Value Returned

<i>t</i>	The specified setup state is removed for the given session.
<i>nil</i>	The specified setup state is not removed.

Examples

Removes the `state1` state that is saved for session `session1`.

```
session = (axlGetWindowSession)  
"session1"  
(axlSetupStates session)  
("state1")  
  
(axlRemoveSetupState session "state1")  
=> t
```

axlSaveSetupState

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

Description

Saves a setup state for the specified session.

Virtuoso ADE SKILL Reference

Session Functions

Arguments

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

Value Returned

<i>t</i>	The setup is saved.
<i>nil</i>	The setup is not saved.

Examples

Virtuoso ADE SKILL Reference

Session Functions

Saves the corner changes from a session to a saved state.

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

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

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

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

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

Saves all the tags from the current session to a state, use the all tag.

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


axlSessionConnect

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

Description

Registers a SKILL callback to be connected to a known signal or trigger emitted from an ADE session.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>t_signalName</i>	Name of the signal or trigger emitted by the ADE session for which to register a callback. To see the list of signals emitted by the ADE sessions, see <code>axlSessionSignalList</code> .
<i>s_callbackFunc</i>	Symbol representing the callback function to be called when the signal is emitted.

Value Returned

<i>t</i>	A callback is successfully registered.
<i>nil</i>	A callback is not registered.

Examples

Connects a custom function, `_myRunModeChangedCB`, with the `runModeChanged` trigger.

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

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

Virtuoso ADE SKILL Reference

Session Functions

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

Related Topics

[axlSessionRegisterCreationCallback](#)

[axlSessionSignalList](#)

[axlSessionSignalSignature](#)

[axlSessionDisconnect](#)

[Working with Signals or Triggers](#)

axlSessionDisconnect

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

Description

Disconnects the specified SKILL callback connected to one or more known signals emitted by the ADE Explorer and ADE Assembler session.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>s_callbackFunction</i>	Symbol representing the callback function to be disconnected.

Value Returned

<i>t</i>	Acallback is successfully disconnected from a signal.
<i>nil</i>	A callback is not disconnected.

Examples

Disconnects the `myFunc` callback from the attached signal or trigger:

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

Related Topics

[axlSessionConnect](#)

[axlSessionRegisterCreationCallback](#)

[axlSessionSignalList](#)

[axlSessionSignalSignature](#)

Virtuoso ADE SKILL Reference

Session Functions

Working with Signals or Triggers

axlSessionRegisterCreationCallback

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

Description

Registers a SKILL function as callback for an event. The callback is called whenever that event occurs.

Arguments

s_callbackFunction

SKILL symbol representing the callback function to be called upon creation of a new session.

Value Returned

t	Successful operation
nil	Unsuccessful operation

Examples

Registers a callback that disables all corners after ADE is launched:

```
(defun ADEexampletrigger_postInstall_disable_corners (session)  
    (let (sdb)  
        sdb = (axlGetMainSetupDB session) ; setup DB handle  
        (axlSetAllCornersEnabled sdb nil)  
    )  
)  
  
(axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session  
"postInstall" 'ADEexampletrigger_postInstall_disable_corners)))
```

Related Topics

[axlSessionConnect](#)

Virtuoso ADE SKILL Reference

Session Functions

axlSessionDisconnect

axlSessionSignalList

axlSessionSignalSignature

axlSessionSignalList

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

Description

Returns a list of all the signals or triggers that are emitted from the given session. You can create custom callback functions to be executed when these events are triggered.

Arguments

<i>t_session</i>	Name of the session.

Value Returned

<i>l_signals</i>	List of signals that are returned from the ADE session.
<i>nil</i>	Unsuccessful registration.

Examples

Returns the list of available signals:

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

Related Topics

[axlSessionConnect](#)

[axlSessionRegisterCreationCallback](#)

[axlSessionSignalSignature](#)

[axlSessionDisconnect](#)

[Working with Signals or Triggers](#)

[Signals or Triggers for ADE Sessions](#)

axlSessionSignalSignature

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

Description

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

Arguments

<i>t_session</i>	String representing the ADE session.
<i>t_signal</i>	Name of a known signal that is emitted by session.

Value Returned

<i>t_signature</i>	The signature of the given signal is returned.
<i>nil</i>	The signature of the given signal is not returned.

Examples

Returns the signature of the `runModeChanged` signal:

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

Related Topics

[axlSessionConnect](#)

[axlSessionRegisterCreationCallback](#)

[axlSessionSignalList](#)

[axlSessionSignalList](#)

[Working with Signals or Triggers](#)

axlSetMainSetupDB

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

Description

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

Arguments

<code>t_session</code>	Name of the session.
<code>t_setupdbPath</code>	Path to a setup database file located in the cellview. The setup database is typically named as <code>data.sdb</code> .

Value Returned

<code>x_hsdb</code>	The setup database handle is returned.
<code>nil</code>	The setup database handle is not returned.

Examples

Creates a new session, `data_session`, and then sets a new database for that.

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

Related Topics

[axlCreateSession](#)

axlSetMainSetupDBLCV

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

Description

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

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

Virtuoso ADE SKILL Reference

Session Functions

Arguments

<i>t_session</i>	Name of the session.
<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>?mode t_mode</i>	Access mode
	Valid values:
■	"a" for append mode. This is the default mode.
■	"r" for read-only mode

Value Returned

<i>x_mainSDB</i>	Handle to the setup database is returned.
<i>nil</i>	Handle to the setup database is not returned.

Examples

Creates a new session, `session0`, and sets up a database to the existing database for the `adexl` view of the `myCell` cell in the `myLib` library.

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

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

axlSetupStates

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

Description

Retrieves a list of setup states from the given session.

Arguments

<i>t_session</i>	Name of the session.

Value Returned

<i>l_states</i>	List of states.

Examples

Returns a list of existing states for the current session and then uses one of the states to modify corners.

```
session = (axlGetWindowSession)  
"session1"  
  
x_mainSDB=axlGetMainSetupDB(session)  
1001  
axlSetupStates(session)  
("state1" "tCorners_state")  
; load one state  
axlLoadSetupState(session "tCorners_state" "All" 'overwrite)  
t  
  
; add a corner  
c1 = axlPutCorner( x_mainSDB "c1" )  
3841  
  
; add variables to the corner  
axlPutVar( c1 "VDD" "1.06 1.08 2.1" )  
3843  
axlPutVar( c1 "temp" "-40 0 75" )  
3845  
  
; save the state  
axlSaveSetupState(session "tCorners_state" `("corners"))  
=> t
```

axlSuppressPersistedQuestionDialog

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

Description

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

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

Arguments

<i>x_msgID</i>	ID of the message to be suppressed.

Value Returned

<i>t</i>	The question dialog specified for the specified <i>msgID</i> is suppressed.
<i>nil</i>	The question dialog is not suppressed.

Examples

Suppresses the question dialog for *msgID* 1234 in the current Virtuoso session.

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

The question dialog is not displayed for *msgID* 1234.

axlShowPersistedQuestionDialog

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

Description

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

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

Arguments

<i>x_msgID</i>	ID of the message to be shown.

Value Returned

<i>t</i>	The suppressed question dialog is displayed for a specified <i>msgID</i> .
<i>nil</i>	The suppressed question dialog is not displayed.

Examples

Shows the question dialog for *msgID* 1234 in the current Virtuoso session:

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

The question dialog be shown for *msgID* 1234.

axlToolSetOpPointInfo

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

Description

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

Arguments

<code>g_sessionID</code>	ID of the current session.
<code>t_testName</code>	Name of the test.
<code>?instanceName t_instanceName</code>	Name of the instance.
<code>?parameters t_parameters</code>	Operating point parameters related to the specified instance.

Value Returned

<code>o_sevOpPoint</code>	Returns the signal object when adding the signal output to the test successfully.
<code>nil</code>	Returns <code>nil</code> if the operation is unsuccessful.

Examples

Adds an `oppoint` type signal to the output setup.

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

Working with Signals or Triggers

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

To execute callbacks or slots for signals, perform the following tasks:

1. Define a callback function

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

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

2. Connect the defined callback function with the signal

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

3. Register the callback when a session is launched

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



Tip

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

The following examples show how you can register callbacks and call them at runtime from a session:

Example 1: To automatically disable corners when ADE is launched

Add the following code in `.cdsinit` to register a callback that disables all corners after ADE is launched:

```
(defun ADEexampletrigger_postInstall_disable_corners (session)
  (let (sdb)
    sdb = (axlGetMainSetupDB session) ; setup DB handle
    (axlSetAllCornersEnabled sdb nil)
  )
)
```


Virtuoso ADE SKILL Reference

Session Functions

```
(axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session
"postInstall" 'ADEexampletrigger_postInstall_disable_corners)))
```

Example 2: To send an email after a run is finished

The following example connects a procedure to the runFinished trigger to send an email on completion of a simulation run:

```
(defun ADEexampletrigger_runFinished_email (session _runid histid _errorid)
  (let (email history lib cell view message command)

    email = "user@cadence.com"

    ;send email with subject "ADE run finished - <history> - <lib> <cell> <view>"
    history = (axlGetHistoryName histid)
    lib = (axlGetSessionLibName session)
    cell = (axlGetSessionCellName session)
    view = (axlGetSessionViewName session)

    (sprintf message "ADE run finished - %s - %s %s %s" history lib cell view)

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

    (sprintf command "mutt -s '%s' %s < /dev/null" message email)
    (system command)

    ; print message to CIW/CDS.log
    (printf "\n%s\n" message)
  )

  (axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session
"runFinished" 'ADEexampletrigger_runFinished_email)))
```

Example 3: To automatically print the job policy settings to CDS.log when a run starts

```
(defun ADEexampletrigger_runStarted_print_jobpolicy (_session _runid histid)
  (let (jp)

    (printf "\nStarting ADE run %s\n" (axlGetHistoryName histid))
    (when (maeGetJobControlMode) == "LSCS"
      jp = (maeGetJobPolicy ?jobType "netlisting")
      (printf "Job Policy (Netlisting):\n")
      (foreach p jp->?
        (printf "%s=%s\n" p (get jp p))
      )
    )
    jp = (maeGetJobPolicy)
    (printf "\nJob Policy (Simulation):\n")
    (foreach p jp->?
      (printf "%s=%s\n" p (get jp p))
    )
  )

)
```

Virtuoso ADE SKILL Reference

Session Functions

```
(axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session
"runStarted" 'ADEexampletrigger_runStarted_print_jobpolicy)))
```

Example 4: To automatically change the job policy per ADE test based on the simulation setup

```
(defun ADEexampletrigger_preRun_pertest_jobpolicy (@rest _args)
  (let (tests cmdbase jp unimode)

; Simplistic example to set a per-test job policy based on simulation options
; This is an LSF bsub command example where the -J option (job name) is modified
; per test

cmdbase = "bsub -q cic -K"
  tests = (maeGetSetup ?typeName "tests" ?enabled t)

  (foreach test tests
    unimode = (asiGetHighPerformanceOptionVal (maeGetTestSession test)
'uniMode)

    (if (maeHasTestJobPolicy test) then
      jp = (maeGetJobPolicy ?testName test)
      else
        jp = maeGetJobPolicy()
      ; Could choose to load from a named policy, see maeGetJobPolicyByName
    )

    (case unimode
      ("Spectre"
        jp->distributionmethod = "Command"
        jp->jobsubmitcommand = (sprintf nil "%s -J SPECTRE" cmdbase)
        jp->name = "LSF Spectre"
        (maeSetJobPolicy jp ?testName test)
      )
      ("APS"
        jp->distributionmethod = "Command"
        jp->jobsubmitcommand = (sprintf nil "%s -J APS" cmdbase)
        jp->name = "LSF APS"
        (maeSetJobPolicy jp ?testName test)
      )
      ("Spectre X"
        jp->distributionmethod = "Command"
        jp->jobsubmitcommand = (sprintf nil "%s -J SPECTREX" cmdbase)
        jp->name = "LSF SpectreX"
        (maeSetJobPolicy jp ?testName test)
      )
    )
  )
)

(axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session
"preRun" 'ADEexampletrigger_preRun_pertest_jobpolicy)))
```

Virtuoso ADE SKILL Reference

Session Functions

Example 5: To automatically perform a task after the setup is saved

```
(defun ADEexampletrigger_setupSaved_doaction (_session)
  printf("\nSetup is saved!!\n")
)

(axlSessionRegisterCreationCallback (lambda (session) (axlSessionConnect session
"setupSaved" 'ADEexampletrigger_setupSaved_doaction)))
```

Example 6: To automatically add a variable to an ADE test when created

```
(defun ADEexampletrigger_postCreatedTest_addvar (_session testname)
  (maeSetVar "VDD" "1.1" ?typeName "test" ?typeValue (list testname))
)

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

Signals or Triggers for ADE Sessions

The following table lists all the signals or triggers available for ADE sessions.

Signal	Change in State
cornersUpdated	When corners are updated. Syntax: cornersUpdated(t_sessionName)
createdTest	When a test is created. Syntax: createdTest(t_sessionName t_testName)
downloadFinishedConclusion	When download of simulation data has finished. Syntax: downloadFinishedConclusion(t_sessionName x_runId x_runhsdb)
eligibleReferenceHistoryItemsChanged	When there is a change in the list of history items that are eligible to be used as reference history items. Syntax: eligibleReferenceHistoryItemsChanged(t_sessionName)

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
<code>initializedTest</code>	When a test is enabled. Syntax: <pre>initializedTest(t_sessionName t_testName)</pre>
<code>netlistProgress</code>	When the netlisting job completes a point netlist. Syntax: <pre>netlistProgress(t_sessionName x_runid)</pre>
<code>ocnPostRunCommandWrite</code>	After writing the <code>ocnxlRun</code> command in the OCEAN script being written from the tool. Syntax: <pre>ocnPostRunCommandWrite(t_sessionName g_filePointer)</pre>
<code>ocnPreRunCommandWrite</code>	Before writing the <code>ocnxlRun</code> command in the OCEAN script being written from the tool. Syntax: <pre>ocnPreRunCommandWrite(t_sessionName g_filePointer)</pre>
<code>parametersUpdated</code>	When the values of parameters are updated. Syntax: <pre>parametersUpdated(t_sessionName)</pre>
<code>pointCompleted</code>	When the simulation of a point is completed. Syntax: <pre>pointCompleted(t_sessionName x_historyHSDB x_pointId t_testName t_cornerName)</pre>
<code>pointSimulationCompleted</code>	When simulation for a point is completed. Syntax: <pre>pointSimulationCompleted(t_sessionName x_historyHSDB t_testName x_pointId)</pre>
<code>postCloseCellView</code>	After the cellview is closed. Syntax: <pre>postCloseCellView(t_sessionName t_lib t_cell t_view t_mode)</pre>

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
postCreateDatasheet	After a datasheet is created. Syntax: <pre>postCreateDatasheet(<i>t_sessionName</i> <i>x_hsdb</i> <i>t_datasheetDir</i>)</pre>
postCreateHistoryEntry	After a history item is created. Syntax: <pre>postCreateHistoryEntry(<i>t_sessionName</i> <i>x_hsdb</i>)</pre>
postCreateMainSetupDB	After a setup database is created. Syntax: <pre>postCreateMainSetupDB(<i>t_sessionName</i> <i>x_hsdb</i>)</pre>
postCreatedTest	After a test is created. Syntax: <pre>postCreatedTest(<i>t_sessionName</i> <i>t_testName</i>)</pre>

Virtuoso ADE SKILL Reference

Session Functions

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

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
postLoadMainSetupDB	After a setup database is loaded Syntax: <pre>postLoadMainSetupDB(t_sessionName x_hsdb)</pre>
postLoadSetupState	After a setup state is loaded. Syntax: <pre>postLoadSetupState(t_sessionName t_stateName)</pre>
postRestoreHistory	After a history item is restored. Syntax: <pre>postRestoreHistory(t_sessionName x_hsdb)</pre>
postSaveSetupState	After a setup state is saved. Syntax: <pre>postSaveSetupState(t_sessionName t_stateName l_saveTags)</pre>
postSaveSimulationResults	After saving the simulation results for a history to the given destination directory. Syntax: <pre>postSaveSimulationResults(t_sessionName x_sdbHandle t_historyName t_destinationDir x_copyPSFResults)</pre> where, <i>x_copyPSFResults</i> takes the Boolean value specified for the <i>Copy PSF Results</i> option on the Save Results form, which opens when you save results for a history.
postSwitchMaestroMode	When switching from ADE Explorer to Assembler or vice versa. Syntax: <pre>postSwitchMaestroMode(t_sessionName t_oldMode t_newMode)</pre>
postViewHistoryResults	After the results for a history item are displayed. Syntax: <pre>postViewHistoryResults(t_sessionName x_historyHSDB t_resultsDBPath)</pre>

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
<code>preCreateDatasheet</code>	Before a datasheet is created. Syntax: <pre>preCreateDatasheet(t_sessionName x_hsdb t_datasheetDir)</pre>
<code>preCreateHistoryEntry</code>	Before a history item is created. Syntax: <pre>preCreateHistoryEntry(t_sessionName x_hsdb)</pre>
<code>preDestroySession</code>	Before closing a session. Syntax: <pre>preDestroySession(t_sessionName)</pre>
<code>preExportResults</code>	Before exporting the results data in the HTML or CSV format. Syntax: <pre>preExportResults(t_sessionName x_historyHSDB t_exportedType t_exportedFilePath)</pre> where, <code>t_exportedType</code> specifies the name of the user interface from where you are exporting results. For example, a Results view or other specification analysis views. Possible values that this argument can take are listed below: ■ Detail ■ Detail - Transpose ■ Optimization ■ Summary ■ Yield ■ Spec Summary ■ Spec Comparison ■ Spec Comparison View

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
preImportSetup	Before a setup is imported. Syntax: <pre>preImportSetup(t_sessionName t_newSetupPath l_importTags t_historyName t_operation)</pre> For the list of available tags to be used for l_importTags, refer to axlImportSetup.
preInstallCellView	Before opening a cellview. Syntax: <pre>preInstallCellView(t_sessionName t_lib t_cell t_view t_mode)</pre>
preLoadSetupState	Before a setup state is loaded. Syntax: <pre>preLoadSetupState(t_sessionName t_stateName)</pre>
preRemoveTest	Before a test is deleted. Syntax: <pre>preRemoveTest(t_sessionName t_testName)</pre>
preRestoreHistory	Before a history item is restored. Syntax: <pre>preRestoreHistory(t_sessionName x_hsdb)</pre>
preRun	Before a simulation run is started. Syntax: <pre>preRun(t_sessionName x_setupdb t_mode t_testName)</pre>
preSaveSetup	When saving a maestro view. Syntax: <pre>preSaveSetup(t_sessionName)</pre>

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
preSaveSetupState	Before a setup state is saved. Syntax: <pre>preSaveSetupState(t_sessionName t_stateName l_saveTags)</pre> For the list of available tags to be used for <code>l_saveTags</code>, refer to <code>axlSaveSetupState</code>.
preSaveSimulationResults	Before saving the simulation results. Syntax: <pre>preSaveSimulationResults(t_sessionName x_sdbHandle t_historyName t_destinationDir x_copyPSFResults)</pre> where, <code>x_copyPSFResults</code> takes the Boolean value specified for the <i>Copy PSF Results</i> option on the Save Results form, which opens when you save results for a history.
preViewHistoryResults	Before the results for a history item are viewed. Syntax: <pre>preViewHistoryResults(t_sessionName x_historyHSDB t_resultsDBPath)</pre>
referenceHistoryItemChanged	When the reference history item is changed. Syntax: <pre>referenceHistoryItemChanged(t_name)</pre>
removedHistoryEntry	When a history item is deleted. Syntax: <pre>removedHistoryEntry(t_sessionName x_hsdb)</pre>
removedTest	When a test is deleted. Syntax: <pre>removedTest(t_sessionName t_testName)</pre>
renamedHistoryEntry	When a history item is renamed. Syntax: <pre>renamedHistoryEntry(t_sessionName x_hsdb t_oldName)</pre>

Virtuoso ADE SKILL Reference

Session Functions

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

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
runFinishedPrePlot	Before plotting the results of a simulation. Syntax: <pre>runFinishedPrePlot(t_sessionName x_runId x_runhsdb)</pre>
runModeChanged	When the simulation run mode is changed. Syntax: <pre>runModeChanged(t_sessionName t_newRunMode)</pre>
runOptionsUpdated	When the run options are updated. Syntax: <pre>runOptionsUpdated(t_sessionName)</pre>
runPaused	When a simulation run is stopped. Syntax: <pre>runPaused(t_sessionName x_runId x_isPaused)</pre>
runProgress	When simulation for a design point is complete. Syntax: <pre>runProgress(t_sessionName x_runId x_numFinished x_numSubmitted)</pre>
runStartPreSDBCommit	Before checkpoint setup when a run is started. Syntax: <pre>runStartPreSDBCommit(t_sessionName x_hsdb)</pre>
runStarted	When a simulation run is started. Syntax: <pre>runStarted(t_sessionName x_runId x_runhsdb)</pre>
setSessionSetupDB	After the creation of a session. Syntax: <pre>setSessionSetupDB(t_sessionName x_hsdb)</pre> If you want to perform a custom action after the ADE session and setup database is created, it is recommended to register a callback for <code>postCreateSessionDB</code> instead of <code>setSessionSetupDB</code>.

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
setupSaved	This trigger is called in the following cases: ■ When a <code>maestro</code> view is saved using the ADE session - <i>Save</i> menu command ■ When an <code>adexl</code> view is saved using the <i>File - Save</i> menu command ■ When a simulation run starts ■ When the setup is saved while exiting the <code>maestro</code> or <code>adexl</code> cellview Note: This trigger works only in the write mode. Syntax: <pre>setupSaved(t_sessionName)</pre>
startedNewManualTuningRun	When the manual tuning run is started. Syntax: <pre>startedNewManualTuningRun(t_sessionName x_runId x_runhsdb)</pre>
testComplete	When the simulations for all sweeps and corners for a test have completed. Syntax: <pre>testComplete(t_sessionName x_runId x_runhsdb t_testName)</pre>
testStatusUpdated	After a test is enabled or disabled. Syntax: <pre>testStatusUpdated(t_sessionName t_testName x_enabled)</pre>
updatedSetupStates	After the list of setup states is changed. Syntax: <pre>updatedSetupStates(t_sessionName)</pre>
updatedMAAnyResults	After the result for an output measured across any dimension is calculated. Syntax: <pre>updatedMAAnyResults(x_hsdb t_testName)</pre>

Virtuoso ADE SKILL Reference

Session Functions

Signal	Change in State
<code>updatedMACResults</code>	After the result for an output measured across corners is calculated. Syntax: <pre>updatedMACResults(x_hbdb t_testName)</pre>
<code>updatedMASResults</code>	After the result for an output measured across sweeps is calculated. Syntax: <pre>updatedMASResults(x_hbdb t_testName)</pre>
<code>updatedTest</code>	After any changes are made to the test. For example, a change in the test output, variable value, analysis, temperature, model file, or simulation file. The event is not triggered when there is a change in the status of an analysis or design variable, or when the state or active setup is saved. Syntax: <pre>updatedTest(t_sessionName t_testName)</pre>
<code>useIncrementalChanged</code>	When the options for incremental simulation are modified Syntax: <pre>useIncrementalChanged()</pre>
<code>variablesUpdated</code>	When variable values are updated Syntax: <pre>variablesUpdated(t_sessionName)</pre>

Setup Database Functions

This topic lists the SKILL functions that let you manage the setup database in ADE Explorer and ADE Assembler.

[axlCloseSetupDB](#)

[axlCommitSetupDB](#)

[axlCommitSetupDBAndHistoryAs](#)

[axlCommitSetupDBas](#)

[axlDiffSetup](#)

[axlDeleteNote](#)

[axlGetActiveSetup](#)

[axlGetCopyRefResultsOption](#)

[axlGetActiveSetup](#)

[axlGetEnabled](#)

[axlExportSetup](#)

[axlGetActiveSetup](#)

[axlGetHistoryGroupChildrenEntry](#)

[axlGetNote](#)

[axlGetPointNetlistDir](#)

[axlGetPointPsfDir](#)

[axlGetPointRunDir](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

axlGetPointTroubleshootDir

axlGetResultsLocation

axlGetReuseNetlistOption

axlGetScript

axlGetSessionFromSetupDB

axlGetScriptPath

axlGetScripts

axlGetSessionFromSetupDB

axlGetSetupDBDir

axlGetSetupInfo

axlGetTopLevel

axlGetUseIncremental

axlImportSetup

axlLoadSetupState

axlNewSetupDB

axlNewSetupDBLCV

axlPutNote

axlPutScript

axlPutTest

axlRemoveElement

axlResetActive

axlSaveSetup

axlSaveSetupToLib

axlSDBDebugPrint

Virtuoso ADE SKILL Reference

Setup Database Functions

[axlSDBGetChild](#)

[axlSDBGetChildren](#)

[axlSDBGetChildVal](#)

[axlSDBGetExtension](#)

[axlSDBGetName](#)

[axlSDBGetValue](#)

[axlSDBHp](#)

[axlSDBPutExtension](#)

[axlSDBSetChild](#)

[axlSDBSetMultipleEntry](#)

[axlSDBSetValue](#)

[axlSetAllSweepsEnabled](#)

[axlSetEnabled](#)

[axlSetReferenceHistoryItemName](#)

[axlSetReuseNetlistOption](#)

[axlSetUseIncremental](#)

[axlSetScriptPath](#)

[axlWriteDatasheet](#)

[axlWriteDatasheetForm](#)

[maeGetReferenceHistories](#)

axlCloseSetupDB

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

Description

Closes an open setup database.

Arguments

<i>x_sdb</i>	Setup database

Value Returned

<i>t</i>	The open setup database is closed.
<i>nil</i>	The open setup database is not closed.

Examples

Closes the specified setup database *sdbh*.

```
axlCloseSetupDB(sdbh)  
=>t
```

axlCommitSetupDB

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

Description

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

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

Arguments

<i>x_hsdb</i>	Setup database handle

Value Returned

<i>x_database</i>	The setup database handle for the saved setup database file is returned.

Examples

Returns the setup database handle for the saved setup database file.

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

axlCommitSetupDBAndHistoryAs

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle
<i>t_setupdbName</i>	Name of the new setup database file to be created.

Value Returned

<i>x_hsdb</i>	The setup database handle is returned for the newly created setup database file.

Examples

Returns the setup database handle for the newly created setup database file.

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

axlCommitSetupDBas

```
axlCommitSetupDBas(  
    x_hsdb  
    t_setupdbName  
)  
=> t / nil
```

Description

Saves the setup database under a new name.

Arguments

<i>x_hsdb</i>	Setup database handle
<i>t_setupdbName</i>	Name of the new setup database file.

Value Returned

<i>t</i>	The setup database is saved in a file with the specified name.
<i>nil</i>	Indicates an error.

Examples

Saves the setup database in a new file named `data.sdb`.

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

axlDiffSetup

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

Description

Compares two setup databases and reports the differences.

Arguments

<i>x_handlea</i>	Setup handle
<i>x_handleb</i>	Setup handle

Value Returned

<i>l_diffs</i>	The list of differences between the two given setup databases is returned.

Examples

Compares the two specified setup databases and reports the differences.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

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

axlDeleteNote

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

Description

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

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.
<i>t_item</i>	Type of the element from which you need to delete a note. The valid values are "test", "history", "corner", "globalvar", "parameters", or "relxanalysis".
<i>t_name</i>	Name of the element for which you are deleting a note.

Value Returned

t	The specified note is deleted.
nil	Indicates an error.

Examples

Deletes a note for corner C0.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
axlDeleteNote(x_mainSDB "corner" "C0")  
=> t
```

Related Topics

[axlPutNote](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

Adding Notes to a Test

axlGetCopyRefResultsOption

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle

Value Returned

<i>t</i>	Indicates that the simulation results need to be copied from the reference history.
<i>nil</i>	Indicates that the simulation results need to be moved from the reference history.

Examples

Specifies that the simulation results need to be copied from the reference history.

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

axlGetElementParent

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

Description

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

Arguments

<i>x_element</i>	Handle for the setup database element .

Value Returned

<i>x_parent</i>	Handle for the parent of <i>x_element</i> . For example, if <i>x_element</i> is the handle to a variable's value, <i>x_parent</i> is the handle to the variable; if <i>x_element</i> is the handle to a variable, <i>x_parent</i> is the handle to the set of variables; and so on up to the top-level setup database handle.

g_errorOrZero

Examples

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

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

axlGetHistoryEntry

axlGetEnabled

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

Description

Checks whether a setup database element is enabled or not.

Arguments

<i>x_element</i>	Handle for the setup database element.

Value Returned

t	Element is enabled.
nil	Element is not enabled.

Examples

Indicates that the setup database element is not enabled.

```
axlGetEnabled(1021)  
=> nil
```

axlGetLocalResultsDir

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

Description

Returns the local results directory associated with a run on a remote machine.

Arguments

<i>x_historyHandle</i>	Handle to a history item.

Value Returned

<i>t_dirPath</i>	Path to the local results directory associated with a run on a remote machine.
nil	The handle to the history is invalid.

Examples

Returns the path to the local results directory associated with the specified run.

```
session = axlGetWindowSession()  
"session1"  
sdb = (axlGetMainSetupDB session)  
1675  
h = axlGetHistoryEntry(sdb "Interactive.0")  
1712  
  
axlGetLocalResultsDir(h)  
=> "/tmp/machineName_username_134646275"
```

axlIsLocalResultsDir

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

Description

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

Arguments

x_historyHandle Handle to the history item.

Value Returned

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

Examples

Indicates that the Use Local Simulation Results Directory flag is enabled for the specified history item.

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

axlExportSetup

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

Description

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

Arguments

<i>t_session</i>	Session name
<i>x_hsdb</i>	Handle to a setup database.
<i>t_path</i>	Setup path
<i>l_tags</i>	Setup handle
	The pre-defined values are vars, tests, parameters, corners, runoptions and scripts.

Value Returned

<i>t</i>	The setup from the currently loaded setup database is exported to a different file.
<i>nil</i>	The setup from the currently loaded setup database is not exported.

Examples

Exports the setup from the currently loaded setup database to a different file.

```
session = axlGetWindowSession()  
"session1"  
sdb=axlGetMainSetupDB(session)  
2141  
axlExportSetup(session sdbh "/tmp/exported.sdb" `("vars"))  
=> t
```


Virtuoso ADE SKILL Reference

Setup Database Functions

```
exported.sdb:
<?xml version="1.0"?>
<setupdb>exported
  <active>Active Setup
    <vars>
      <var>CLoad
        <value>1.5p</value>
      </var>
      <var>RLoad
        <value>10M</value>
      </var>
      <var>VDC
        <value>2.7</value>
      </var>
    </vars>
  </active>
  <history>History</history>
</setupdb>
```

axlGetHistoryGroupChildren

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

Description

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

Arguments

<i>x_element</i>	Handle for the setup database element.

Value Returned

<i>l_children</i>	A list that contains the following: ■ A handle to all history entries in the history group ■ A list of names of all the history entries.

Examples

Returns a list containing a handle to all history children entries in the history group `Group.0` and a list of names of all the history children entries

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

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

axlGetActiveSetup

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

Description

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

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.

Value Returned

<i>x_activeSetup</i>	Handle to the active setup.

Examples

Returns the status of the *Overwrite History* save option.

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

axlGetHistoryGroupChildrenEntry

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

Description

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

Arguments

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

Value Returned

<i>x_history</i>	The handle to a history entry is returned.
0	The handle to a history entry is not returned.

Examples

Finds the specified history entry and returns the handle to it.

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

[axlGetActiveSetup](#)

axlGetNote

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

Description

Returns notes from the specified test, history, corner, parameters, variable or Reliability Analysis setup.

Arguments

<i>x_hsdb</i>	Handle to the main setup database.
<i>t_item</i>	Type of the element from which you want to get notes. The valid values are "test", "history", "corner", "globalvar", "parameters", or "relxanalysis".
<i>t_name</i>	Name of the element from which you want to get notes.

Value Returned

<i>t_note</i>	Notes from the specified element are returned.
<i>nil</i>	No notes are returned.

Examples

Retrieves a note for the corner C0:

```
axlGetNote(1001 "corner" "C0")  
=> "Notes -- name : Corner C0, temp : 30, CAP : 2p"
```

Related Topics

[axlDeleteNote](#)

[axlPutNote](#)

axlGetPointNetlistDir

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

Description

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

Arguments

<i>x_historyID</i>	History ID
<i>t_testName</i>	Name of the test.
<i>?cornerName t_cornerName</i>	Name of the corner. If a corner name is specified then point ID must also be specified. If both <i>?cornerName</i> and <i>?pointID</i> arguments are not specified, top PSF level netlist directory is returned.
<i>?designPointId x_designPointId</i>	Point ID

Value Returned

<i>t_pointNetlistDir</i>	The path to the netlist directory is returned.
<i>nil</i>	The path to the netlist directory is not returned.

Examples

Returns the path to the netlist directory.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/psf/op  
amplib:ampTest:1/netlist"  
axlGetPointNetlistDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")  
"Interactive.12") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)  
=>  
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/2/opam  
plib:ampTest:1/netlist"
```

axlGetPointPsfDir

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

Description

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

Arguments

<i>x_historyID</i>	History ID
<i>t_testName</i>	Name of the test.
<i>?cornerName t_cornerName</i>	Name of the corner. If a corner name is specified then point ID must also be specified. If both <i>?cornerName</i> and <i>?pointID</i> arguments are not specified, top psf level netlist directory is returned.
<i>?designPointId x_designpointId</i>	Point ID

Value Returned

<i>t_pointPsfDir</i>	The path to the point psf directory is returned.
<i>nil</i>	The path to the point psf directory is not returned.

Examples

Returns the path to the point psf directory.

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


Virtuoso ADE SKILL Reference

Setup Database Functions

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/psf/op  
amplib:ampTest:1/psf"  
axlGetPointPsfDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")  
"Interactive.12") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)  
=>  
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12/2/opam  
plib:ampTest:1/psf"
```

axlGetPointRunDir

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

Description

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

Arguments

<i>x_historyID</i>	History ID
<i>t_testName</i>	Name of the test.
<i>?cornerName t_cornerName</i>	Name of the corner. If a corner name is specified then point ID must also be specified. If both <i>?cornerName</i> and <i>?pointID</i> arguments are not specified, top psf level netlist directory is returned.
<i>?designPointId x_designPointId</i>	Point ID

Value Returned

<i>t_pointRunDir</i>	The path to the point run directory is returned.
<i>nil</i>	The path to the point run directory is not returned.

Examples

Returns the path to the point run directory.

```
axlGetPointRunDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")  
"Interactive.12") "opamplib:ampTest:1")  
=> "/net/...../simulation/opamplib/ampTest/adex11/results/data/Interactive.12/"
```

Virtuoso ADE SKILL Reference

Setup Database Functions

```
psf/opamplib:ampTest:1/"
axlGetPointRunDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)
=> "/net/...../simulation/opamplib/ampTest/adex11/results/data/Interactive.12/
2/opamplib:ampTest:1/"
```

axlGetPointTroubleshootDir

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

Description

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

Arguments

<i>x_historyID</i>	History ID
<i>t_testName</i>	Name of the test.
<i>?cornerName t_cornerName</i>	Name of the corner. If a corner name is specified then point ID must also be specified. If both <i>?cornerName</i> and <i>?pointID</i> arguments are not specified, top psf level netlist directory is returned.
<i>?designPointId x_designPointId</i>	Point ID

Value Returned

<i>t_trblDir</i>	The path to the troubleshoot directory is returned.
<i>nil</i>	The path to the troubleshoot directory is not returned.

Examples

Returns the path to the troubleshoot directory.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

```
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12.TS.0/p
sf/opamplib:ampTest:1/troubleshoot"
axlGetPointTroubleshootDir(axlGetHistoryEntry(axlGetMainSetupDB("session0")
"Interactive.12.TS.0") "opamplib:ampTest:1" ?cornerName "C0_0" ?designPointId 1)
=>
"/net/...../simulation/opamplib/ampTest/adexl1/results/data/Interactive.12.TS.0/2
/opamplib:ampTest:1/troubleshoot"
```

axlGetResultsLocation

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

Description

Returns the results location for the specified setup database.

Note: If the variable `saveResDir` is set to a location other than the default setting, `axlGetResultsLocation` use sits setting to determine the results location. But if it is set to the default location, `axlGetResultsLocation` uses the `saveDir` setting to determine the results location.

Arguments

<code>x_hsdb</code>	Setup database handle

Value Returned

<code>t_resultsLocation</code>	The results location is returned which includes a directory named from the setup database name prefix.
<code>nil</code>	Indicates an error.

Examples

If you do not set the `saveDir` environment variable in your `.cdsenv`:

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

If you set the `saveDir` environment variable to `RESULTS`, the results change as follows:

```
adexl.results saveDir string "RESULTS"  
=> "RESULTS/myLib/myCell/adexl/results/data"
```

Here is another example where `saveDir` is not set:

```
resultsLoc = (axlGetResultsLocation (axlGetHistoryEntry (axlGetMainSetupDB  
axlGetWindowSession() ) "Interactive.0" ) )  
=> "opamplib/ampTest/adexl/results/data"
```

Virtuoso ADE SKILL Reference

Setup Database Functions

Related Topics

[adexl.results saveDir](#)

[adexl.results saveResDir](#)

[axlCreateSession](#)

[axlGetHistoryEntry](#)

[axlGetMainSetupDB](#)

[axlSetMainSetupDB](#)

axlGetReuseNetlistOption

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle to active setup or checkpoint.

Value Returned

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

Examples

Indicates that the option to use reference netlist is enabled.

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


axlGetScript

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

Description

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

Arguments

<i>x_element</i>	Setup database element handle.
<i>t_scriptName</i>	Script name

Value Returned

<i>x_script</i>	The script handle is returned.
nil	The script handle is not returned.

Examples

Finds the script `myScript.nam` and returns its handle 1045.

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

axlGetScriptPath

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

Description

Returns the path of a script.

Arguments

<i>x_script</i>	Script handle

Value Returned

<i>t_path</i>	The script path is returned.
<i>nil</i>	The script path is not returned.

Examples

Returns the path to the script with the handle 1045.

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

axlGetScripts

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

Description

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

Arguments

<i>x_element</i>	Handle to the setup database element.

Value Returned

<i>l_scripts</i>	A list containing a handle to all scripts for this database entry is returned along with a list of all script names.
<i>nil</i>	Indicates an error.

Examples

Returns a list containing the handle to all scripts for the database entry 1045 along with a list of all script names.

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

axlGetSessionFromSetupDB

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

Description

Determines the session associated with the provided setup database handle.

Arguments

<i>x_hsdb</i>	Determines the session associated with the handle to a setup database.

Value Returned

<i>t_sessionName</i>	The session associated with given provided setup database handle is returned.
<i>nil</i>	Indicates an error.

Examples

Returns the name of the session associated with the setup database handle 1001.

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

axlGetSetupDBDir

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

Description

Returns the path to the directory of the specified setup database.

Arguments

<i>x_hsdb</i>	Handle to the specified setup database.

Value Returned

<i>t_dir</i>	The path to the setup database directory is returned.
<i>nil</i>	The path to the setup database directory is not returned.

Examples

Returns the path to the directory of the specified setup database.

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

axlGetSetupInfo

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

Description

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

Arguments

t_sessionName Name of the active session

?testName t_testName

Test name for which the setup information is to be retrieved. If the test name is not provided, the tool displays the setup information for all the tests.

The default value is " ".

Value Returned

r_setupInfo The setup information containing the count of corners, sweeps, and data points is returned.

nil Indicates an error.

Examples

Returns the setup information for the active setup:

```
session = axlGetWindowSession()  
=> "session0"  
  
x_mainSDB = axlGetMainSetupDB(session)  
=>1001  
  
;; display the total number of points, corners, and data points for all the corners  
axlGetSetupInfo(session)  
=> (nil Corners 6 SweepPoints 1 DataPoints 7)  
;  
;; display the number of points, corners, and data points for the AC test
```

Virtuoso ADE SKILL Reference

Setup Database Functions

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

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

axlGetTopLevel

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

Description

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

Arguments

<i>x_element</i>	Setup database element handle

Value Returned

<i>x_hsdb</i>	The handle to the setup database containing the specified element is returned.
<i>g_errorOrZero</i>	Error message if input argument is invalid or zero for otherwise unsuccessful operation.

Examples

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

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

[axlGetHistoryEntry](#)

axlGetUseIncremental

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle to the active setup or checkpoint.

Value Returned

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

Examples

Specifies that using reference results as cache during incremental run is enabled.

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

axlImportSetup

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

Description

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

Arguments

<i>t_session</i>	Session Name
<i>t_path</i>	Setup path
<i>l_tags</i>	List of tags.
<i>t_historyName</i>	Name of the history from where you need to import the setup.
<i>s_operation</i>	Operation Type
	The valid values are "merge", "retain" or "overwrite".

Value Returned

<i>t</i>	The setup is imported from the file.
<i>nil</i>	The setup is not imported from the file.

Examples

Imports the setup from the given file.

```
session = axlGetWindowSession(hiGetCurrentWindow())  
"session1"  
axlImportSetup(session "/tmp/exported.sdb" ("vars") "" "retain")  
=> t
```

axlLoadSetupState

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

Description

Loads a setup state.

Virtuoso ADE SKILL Reference

Setup Database Functions

Arguments

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

Value Returned

<i>t</i>	The setup state is loaded.
<i>nil</i>	The setup state is not loaded.

Examples

Loads the setup state.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

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

axlNewSetupDB

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

Description

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

Arguments

<i>t_setupdbName</i>	Name of the setup database.

Value Returned

<i>x_hbdb</i>	The setup database handle is returned.
nil	The setup database handle is not returned.

Examples

Opens the given setup database and returns its handle.

```
axlNewSetupDB( "data" )  
=> 1
```

axlNewSetupDBLCV

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

Description

Creates a new setup database in the specified location. It automatically overwrites any existing setup database in any of the above mentioned locations.

Arguments

<i>t_libraryName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.

Value Returned

<i>t</i>	A new setup database is created in the specified location.
<i>nil</i>	No setup database is created.

Examples

Creates a new setup database in the specified location.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

axlPutNote

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

Description

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

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.
<i>t_item</i>	Type of the element to which you need to add a note. The valid values are "test", "history", "corner", "globalvar", "parameters", or "relxanalysis".
<i>t_name</i>	Name of the element to which you are adding a note.
<i>t_note</i>	Text to be added to the note. A note can contain a maximum of 512 characters.

Value Returned

<i>t</i>	A note is added to the given test, history, corner, parameters, variable or Reliability Analysis setup.
<i>nil</i>	No note is added.

Examples

Adds a note for corner C0.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
axlPutNote(x_mainSDB "corner" "C0" "this is a fast corner")  
=> t
```


Virtuoso ADE SKILL Reference

Setup Database Functions

Related Topics

[axlDeleteNote](#)

[Adding Notes to a Test](#)

axlPutScript

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

Description

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

Arguments

<i>x_element</i>	Setup database element handle
<i>t_scriptName</i>	Script name
<i>t_path</i>	Path to be set for the script.

Value Returned

<i>x_script</i>	The handle to the script.
<i>nil</i>	Indicates an error.

Examples

Inserts or locates a script with handle 1104, assigning `scriptname` and `/path/to/script/file` as its name and path, and returning the script's handle.

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

axlPutTest

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

Description

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

Arguments

<code>x_hsdb</code>	Setup database handle
<code>t_test</code>	Test name
<code>t_tool</code>	Tool name

Value Returned

<code>x_test</code>	The test handle is returned.
<code>nil</code>	Indicates an error.

Examples

Inserts or locates a script into the setup database `data_sdb` and returns the handle.

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

axlRemoveElement

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

Description

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

Arguments

<i>x_element</i>	Setup database element handle

Value Returned

t	The specified element and all its children are removed from the setup database.
nil	Indicates an error.

Examples

Removes the given element and all its children from the setup database,

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

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

[axlGetHistoryEntry](#)

axlResetActive

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

Description

Resets the active setup database.

Arguments

<i>x_hsdb</i>	Setup database handle

Value Returned

<i>t</i>	The active setup database is reset.
<i>nil</i>	The active setup database is not reset.

Examples

Resets the setup database with handle 1003.

```
axlResetActive( 1003 )  
=> t
```

axlSaveSetup

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

Description

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

Arguments

None

Value Returned

t	The setup database file and test state files for the current window are saved.
nil	The setup database file and test state files are not saved.

Examples

Saves the setup database file and test state files for the current window.

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

axlSaveSetupToLib

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle
<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.

Value Returned

<i>t</i>	The setup database is saved to the specified location.
<i>nil</i>	The setup database is not saved.

Examples

Saves the setup database to the specified location.

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

axlSDBDebugPrint

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

Description

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

Arguments

<i>x_SDBH</i>	Setup database handle

Value Returned

<i>x_SDBH</i>	The handle of the supplied <i>x_SDBH</i> is returned but this value is inconsequential as this function is called for the printed tree.

Examples

Prints the in-memory database rooted at the supplied handle

```
;; given mainSDBH pointing to current setup and some variables  
varsSDBH = car( axlGetVars(mainSDBH))  
axlSDBDebugPrint(varsSDBH) ->  
1297--\vars|''  
    1304---\var|RES  <enabled: 1>  
        0---->value|4k  
    1338---\var|BOB  
        0---->value|1 2 3
```


axlSDBGetChild

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

Description

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

Arguments

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

Value Returned

<i>x_hbdb</i>	Setup database handle to the child element.
nil	Unsuccessful operation.

Examples

Returns the setup database handle to a single child element.

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

axlSDBGetChildren

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

Description

Returns child elements of the provided setup database handle.

Arguments

<i>x_SDBH</i>	Setup database handle
<i>t_name</i>	Optional filtration by element name. If not provided, all child elements of the parent will be returned.
<i>t_value</i>	Optional filtration by element value. If provided, <i>t_name</i> must also be specified.
<i>S_returnAs</i>	Format of the return value. ■ 'handles: Returns a list of the setup database handles of the child elements. This is the default. ■ 'assoc: Returns a list of name-value pairs, such as ((elt1Name elt1Val) (elt2Name elt2Val)).

Value Returned

<i>l_children</i>	A list corresponding to the value set for <i>s_returnAs</i> is returned.
<i>nil</i>	Indicates an error,

Examples

Returns child elements of the given setup database handle.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

```
axlSDBGetChildren(extensionSDBH ?name "King" ?returnAs 'assoc)
=> (("King" "Louis XIV"))
axlSDBSetMultipleEntry( "King")
axlSDBSetChild( extensionSDBH "King" "Henry VII")
axlSDBGetChildren(extensionSDBH ?name "King" ?returnAs 'handles)
=> '(1852 1854)
axlSDBGetChildren(extensionSDBH ?name "King" ?returnAs 'assoc)
=> (("King" "Louis XIV") ("King" "Henry VII"))
axlSDBGetChildren(extensionSDBH ?name "King" ?value "Louis XIV" ?returnAs 'assoc)
=> (("King" "Louis XIV"))
```

axlSDBGetChildVal

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

Description

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

Arguments

<i>x_SDBH</i>	Setup database handle
<i>t_name</i>	Name of the element.

Value Returned

<i>t_value</i>	The value for a single child element is returned.
<i>nil</i>	Indicates an error.

Examples

Returns the value for a single child element.

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

axlSDBGetExtension

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

Description

Returns the named setup database extension.

Arguments

<i>x_mainSDBH</i>	Main setup database handle.
<i>t_extensionName</i>	Name of an extension previously added with <code>axlSDBPutExtension()</code> .

Value Returned

<i>x_SDBH</i>	The setup database handle of the named extension is returned.
nil	The setup database handle of the named extension is not returned.

Examples

Returns the named setup database extension.

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

axlSDBGetName

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

Description

Returns the name of the provided setup database handle.

Arguments

<i>x_SDBH</i>	Setup database handle

Value Returned

<i>t_name</i>	The name of the given setup database handle is returned.
<i>nil</i>	The name of the given setup database handle is not returned.

Examples

Returns the name of the given setup database handle.

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

axlSDBGetValue

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

Description

Returns the value of the provided setup database handle.

Arguments

<i>x_SDBH</i>	Setup database handle.

Value Returned

<i>t_value</i>	The value of the given setup database handle is returned.
<i>nil</i>	The value of the given setup database handle is not returned.

Examples

Returns the value of the given setup database handle.

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

axlSDBHp

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

Description

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

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

Arguments

<i>g_potentialSDBH</i>	Setup database handle

Value Returned

<code>t</code>	Indicates that the setup database handle is valid.
<code>nil</code>	Indicates that the setup database handle is invalid.

Examples

Indicates that the setup database handle is valid.

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

To remove the variable:

```
axlRemoveElement(varSDBH)
```

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


axlSDBPutExtension

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

Description

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

Arguments

<i>x_mainSDBH</i>	Main setup database handle.
<i>t_extensionName</i>	Name of the extension.

Value Returned

<i>x_SDBH</i>	The setup database handle of the named extension is returned.
nil	The setup database handle of the named extension is not returned.

Examples

Adds an extension to the ADE setup database.

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

axlSDBSetChild

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

Description

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

Virtuoso ADE SKILL Reference

Setup Database Functions

Arguments

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

Value Returned

<i>x_SDBH</i>	The setup database handle of the child element is returned.
<i>nil</i>	The setup database handle of the child element is not returned.

Examples

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

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

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

Single-entry:

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

Virtuoso ADE SKILL Reference

Setup Database Functions

Since `King` is a single-entry name (the default), there can only be one child of `extensionSDBH` with `King` as its name. Attempting to specify another child element will update the value of the pre-existing element, and not create a new element. name can be made multi-entry by calling `axlSDBSetMultipleEntry`, after which calls to `axlSDBSetChild` with same name and different value will create new child elements.

For example:

```
axlSDBSetChild( extensionSDBH "King" "Henry VIII")
;; create an aggregating child under king. This isn't necessary but keeps the
database tidy
;; we are not supplying a value because this is only demarcating a collection
ministersSDBH = axlSDBSetChild( kingSDBH "Ministers" "")
;; set the Minister tag as multi-entry. No session or SDB handle is provided, and
this affects *all* open ADE sessions and databases in the current
;; virtuoso invocation.
axlSDBSetMultipleEntry( "Minister")
;; OK, now we can add multiple ministers
axlSDBSetChild( ministersSDBH "Minister" "Alice")
axlSDBSetChild( ministersSDBH "Minister" "Bob")
axlSDBSetChild( ministersSDBH "Minister" "Eve")
```

axlSDBSetMultipleEntry

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

Description

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

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

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

Virtuoso ADE SKILL Reference

Setup Database Functions

Arguments

<i>t_name</i>	Name of the element to be set as multi-entry. Following are the naming conventions: ■ Must start with an uppercase letter ■ Can contain letters, digits, underscores, hyphens, and periods ■ Cannot contain whitespace Note: Names beginning with lowercase letters are reserved for future ADE expansion.

Value Returned

<i>t</i>	The setup database handle is set as a multi-entry element.
<i>nil</i>	The setup database handle is not set as a multi-entry element.

Examples

Sets the setup database `Ministers` as a multiple entry name.

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

axlSDBSetValue

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

Description

Sets the value of the provided setup database handle.

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

Arguments

<i>x_SDBH</i>	Setup database handle
<i>t_newValue</i>	The new value to be set to the setup database handle.

Value Returned

<i>t</i>	The specified value is set for the setup database handle.
<i>nil</i>	The specified value is not set for the setup database handle.

Examples

Sets the value of the setup database handle `KingSDBH` to `myVDC`.

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

axlSetAllSweepsEnabled

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle
<i>g_enableStatus</i>	Selects or deselects the <i>Point Sweeps</i> check box in the Run Summary assistant pane. ■ 1: Selects the Point Sweeps check box. ■ 0: Deselects the Point Sweeps check box.

Value Returned

t	The <i>Point Sweeps</i> check box is selected or deselected.
nil	The status of the <i>Point Sweeps</i> check box is not altered.

Examples

Selects the *Point Sweeps* check box.

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

Deselects the *Point Sweeps* check box.

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


axlSetCopyRefResultsOption

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

Description

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

Arguments

<i>x_hsdb</i>	Setup database handle
<i>g_value</i>	Boolean value
■ <i>t</i> :	Specifies that the simulation results need to be copied.
■ <i>nil</i> :	Specifies that they need to be moved.

Value Returned

<i>x_hsdb</i>	The setup handle is returned.
0	The setup handle is not returned.

Examples

Specifies that simulation results must be copied from the reference history.

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

axlSetEnabled

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

Description

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

Arguments

<i>x_element</i>	Setup database element handle
<i>g_enable</i>	Enables the flag. ■ <i>t</i>: Enables the flag. ■ <i>nil</i>: Disables the flag.

Value Returned

<i>t</i>	The specified setup database is enabled or disabled.
<i>nil</i>	Indicates an error.

Examples

Uses the `axlSetEnabled` function to enable or disable different elements in the setup database.

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

Virtuoso ADE SKILL Reference

Setup Database Functions

```
axlSetEnabled( axlGetVar( x_mainDB param ) nil ))  
=> ("CAP" "R0" "R1" "LENGTH")
```

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

[axlGetTests](#)

[axlGetTestaxlGetVars](#)

[axlGetVar](#)

axlSetReferenceHistoryItemName

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

Description

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

Arguments

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

Value Returned

<i>x_hsdb</i>	The setup handle for the newly set history name is returned.
0	The setup handle is not returned.

Examples

Lets you reuse the netlist from the specified reference history.

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

Related Topics

[axlSetReuseNetlistOption](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

[axlSetUseIncremental](#)

[axlGetResultsLocation](#)

[Running an Incremental Simulation](#)

axlSetReuseNetlistOption

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

Description

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

Arguments

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

Value Returned

<i>x_hsdb</i>	The setup handle is returned if the option to use the reference netlist is set successfully
0	The setup handle is not returned.

Examples

Uses the `axlSetReuseNetlistOption` function to reuse netlist from a reference history for a new simulation.

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

Related Topics

[axlSetReferenceHistoryItemName](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

[axlSetUseIncremental](#)

[axlGetReuseNetlistOption](#)

[Running an Incremental Simulation](#)

axlSetUseIncremental

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

Description

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

Arguments

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

Value Returned

<i>x_hsdb</i>	The setup handle is returned if the option to use the reference results as cache during incremental run is set successfully
0	The setup handle is not returned.

Examples

Uses the `axlSetUseIncremental` function to use the reference results as cache during an incremental run.

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

Related Topics

[axlSetReferenceHistoryItemName](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

[axlSetReuseNetlistOption](#)

[Running an Incremental Simulation](#)

axlSetScriptPath

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

Description

Sets the path of a script.

Arguments

<i>x_script</i>	The handle to the script for which the path is to be set.
<i>t_path</i>	Specifies the path to be set for the given script.

Value Returned

t	The path for the script is set successfully.
nil	The path is not set for the script.

Examples

Sets the path for the given script.

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

axlWriteDatasheet

```
axlWriteDatasheet(  
    t_axlSession  
    x_historyEntry  
    [ ?directory t_directory ]  
    [ ?resultsSummary g_resultsSummary ]  
    [ ?testsSummary g_testsSummary ]  
    [ ?detailedResults g_detailedResults ]  
    [ ?plots g_plots ]  
    [ ?designVarsSummary g_designVarsSummary ]  
    [ ?paramsSummary g_paramsSummary ]  
    [ ?cornersSummary g_cornersSummary ]  
    [ ?setupSummary g_setupSummary ]  
    [ ?schematicDiagrams g_schematicDiagrams ]  
    [ ?launchBrowser g_launchBrowser ]  
    [ ?name t_name ]  
)  
=> t / nil
```

Description

Creates a datasheet for the specified history entry.

Virtuoso ADE SKILL Reference

Setup Database Functions

Arguments

<code>t_axlSession</code>	Name of the session.
<code>x_historyEntry</code>	Integer value representing the history entry.
<code>?directory t_directory</code>	Target directory for the datasheet. Default Value: <code>libName/cellName/adexl/datasheets</code>
<code>?resultSummary g_resultsSummary</code>	Boolean to specify whether or not you want to print a results summary sheet containing specification sheet pass/fail table. Default Value: <code>t</code>
<code>?testSummary g_testsSummary</code>	Boolean value to specify whether or not you want to print a tests summary sheet containing details about the tests, sweeps, and corners. Default Value: <code>t</code>
<code>?detailedResults g_detailedResults</code>	Boolean value to specify whether or not you want to generate results for all points. Default Value: <code>t</code>
<code>?plots g_plots</code>	Boolean value to specify whether or not you want to print the plots in the generated datasheet. Default Value: <code>t</code>
<code>?designVarsSummary g_designVarsSummary</code>	Boolean value to specify whether or not you want to generate results for all points. Default Value: <code>t</code>
<code>?paramsSummary g_paramsSummary</code>	Boolean value to specify whether or not you want to save the parameters summary in the generated datasheet. Default Value: <code>t</code>
<code>?cornersSummary g_cornersSummary</code>	

Virtuoso ADE SKILL Reference

Setup Database Functions

Boolean value to specify whether or not you want to save the corners summary in the generated datasheet.

Default Value: `t`

`?setupSummary` *g_setupSummary*

Boolean value to specify whether or not you want to save the setup summary in the generated datasheet.

Default Value: `t`

`?schematicDiagrams` *g_schematicDiagrams*

Boolean value to specify whether or not you want to save the schematic diagrams in the generated datasheet.

Default Value: `t`

`?launchBrowser` *g_launchBrowser*

Boolean value to specify whether or not you want to launch a browser window to view the generated datasheet.

Default Value: `t`

`?name` *t_name*

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

Value Returned

`t`

A datasheet is created for the specified history entry.

`nil`

No datasheet is created for the specified history entry.

Examples

Creates a datasheet for the specified history entry.

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

axlWriteDatasheetForm

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

Description

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

Arguments

<i>t_axlSession</i>	String value representing the session name.
<i>x_historyEntry</i>	Integer value representing the history entry.

Value Returned

<i>t</i>	The form is opened.
<i>nil</i>	The form is not opened.

Examples

Opens the form that lets you specify the options for generating a datasheet.

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

maeGetReferenceHistories

```
maeGetReferenceHistories(  
    x_hsdb  
)  
=> l_referenceHistoryNames
```

Description

Returns a list of names of the reference histories used in the active setup or the given history.

Arguments

<i>x_hsdb</i>	Setup database handle to the active setup or history.

Value Returned

<i>l_referenceHistoryNames</i>	List of the names of reference histories.

Examples

Returns the reference histories set in the active setup.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
  
maeGetReferenceHistories(data_sdb)  
=> (22552 ("Interactive.1" "Interactive.4") )
```

Returns the reference histories set in the active setup.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
h=axlGetHistoryEntry(data_sdb "Interactive.2")  
=> 2140  
maeGetReferenceHistories(h)  
=> (2148  
    ("Nominal_initial" "Interactive.0")  
)
```

Related Topics

[axlCreateSession](#)

Virtuoso ADE SKILL Reference

Setup Database Functions

axlGetMainSetupDB

Variable Functions

The following SKILL functions let you manage the variables associated with the setup database, corner variables, and a history checkpoint. Use these functions to add a variable, set or get its value, or set default variables.

[axlGetVars](#)

[axlGetVar](#)

[axlGetVarValue](#)

[axlPutVar](#)

[axlGetAllVarsDisabled](#)

[axlSetAllVarsDisabled](#)

[axlSetDefaultVariables](#)

[axlSetDesignVariablePerTest](#)

[maeExportDesignVariables](#)

[maeImportDesignVariables](#)

axlGetVar

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

Description

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

Arguments

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

Value Returned

<i>x_var</i>	Handle to the specified variable.
<i>nil</i>	The command was unsuccessful.

Examples

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

```
s1 = axlGetWindowSession()  
=> "session0"  
  
x_mainSDB = axlGetMainSetupDB( s1 )  
=> 1001  
  
axlGetVars(x_mainSDB)  
=> (1862  
    ("IREF" "SIDDQ" "VDD" "VIN_CM" "testVar"))  
  
var1 = axlGetVar(x_mainSDB "testVar")  
=> 1952  
axlRemoveElement(var1)  
=> t
```

Virtuoso ADE SKILL Reference

Variable Functions

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

[axlGetVars](#)

axlGetVars

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

Description

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

Arguments

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

Value Returned

<i>l_vars</i>	List containing a handle to all global variables for the database element and a list of all global variable names.
<i>nil</i>	The command was unsuccessful.

Examples

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

```
s1 = axlGetWindowSession()  
=> "session0"  
x_mainSDB=axlGetMainSetupDB(s1)  
=> 1001  
  
axlGetVars(x_mainSDB)  
=>(1862  
    ("IREF" "SIDDQ" "VDD" "VIN_CM"))
```

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

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

Virtuoso ADE SKILL Reference

Variable Functions

```
=> (1003
    ("C0_VDD_1.6_Temp" "C1_VDD_2.0_Temp" "C1_VDD_2.2_Temp" )
)
c1 = axlGetCorner(x_mainSDB "C0_VDD_1.6_Temp")
=> 1984

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

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

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

x_mainSDB=axlGetMainSetupDB( s1 )
=> 1001

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

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

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

[axlGetCorners](#)

[axlGetCorner](#)

[axlGetHistoryEntry](#)

axlGetVarValue

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

Description

Returns value of the given variable.

Arguments

<i>x_varHandle</i>	Handle to a variable.

Value Returned

<i>t_value</i>	Value of the variable.
<i>nil</i>	Handle to the variable is not valid.

Examples

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

```
s1 = axlGetWindowSession()  
"session0"  
  
x_mainSDB=axlGetMainSetupDB( s1 )  
=>1001  
  
axlGetVars(x_mainSDB)  
  
=>(15615  
    ("IREF" "VDD" "VIN_CM" "_sim_time" "SIDDQ")  
)  
  
v2=axlGetVar(x_mainSDB "VDD")  
=>15622  
  
v2=axlGetVarValue(v2)  
=>"1.1 1.2"
```

Related Topics

[axlGetWindowSession](#)

Virtuoso ADE SKILL Reference

Variable Functions

axlGetMainSetupDB

axlGetVars

axlGetVar

axlPutVar

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

Description

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

Arguments

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

Value Returned

<i>x_varHandle</i>	Handle to the variable added or modified.
nil	The command was unsuccessful.

Examples

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

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

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

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


Virtuoso ADE SKILL Reference

Variable Functions

```
axlPutVar(x_mainSDB "IREF" "55u")  
=>1863
```

The following example shows how to add a configuration sweep variable.

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

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

[axlGetCorner](#)

axlGetAllVarsDisabled

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

Description

Returns the status for inclusion of the global variables in a simulation. In the GUI, this is the selection status of the *Global Variables* check box in the Data View assistant pane of ADE Assembler.

Arguments

<i>x_mainSDB</i>	Setup database handle.

Value Returned

<i>t</i>	The <i>Global Variables</i> check box in the Data View assistant pane is not selected.
<i>nil</i>	The <i>Global Variables</i> check box in the Data View assistant pane is selected.

Examples

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

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

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

axlSetAllVarsDisabled

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

Description

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

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>g_enableStatus</i>	Option for setting the selection status. ■ 0: Selects the <i>Global Variables</i> check box. ■ 1: Deselects the <i>Global Variables</i> check box.

Value Returned

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

Examples

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

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

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

Virtuoso ADE SKILL Reference

Variable Functions

```
s1 = axlGetWindowSession()  
=> "session0"  
  
x_mainSDB=axlGetMainSetupDB("s1")  
=> 1001  
  
axlSetAllVarsDisabled(x_mainSDB 1)  
=> t
```

axlSetDefaultVariables

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

Description

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

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

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

Arguments

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

Value Returned

t	A set of default variables were created.
nil	The command was unsuccessful.

Examples

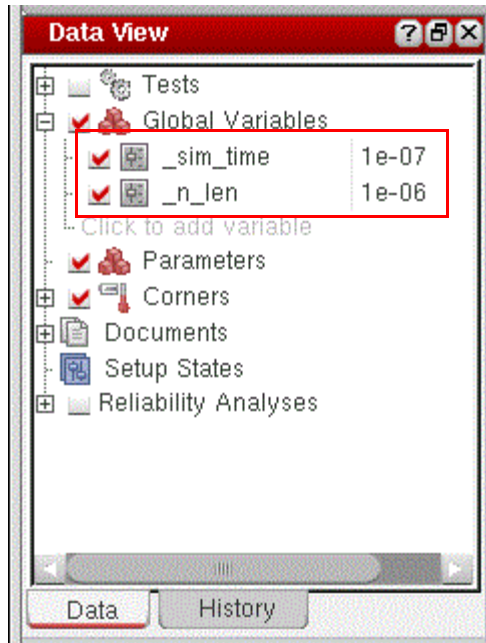
If you add the following statement in the .cdsinit file, it adds two variables, `_n_len` and `_sim_time`, with their default values to the `myDemoLib` library.

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

Virtuoso ADE SKILL Reference

Variable Functions

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



axlSetDesignVariablePerTest

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

Description

Enables or disables a design variable for a specific test. In the Setup assistant of ADE Explorer or the Data View assistant of ADE Assembler, it selects or clears the check box next to a design variable under a test.

Arguments

<i>x_mainSDB</i>	Handle to the main setup database
<i>t_varName</i>	Name of the design variable
<i>t_testName</i>	Name of the test
<i>?enabled g_enableStatus</i>	Enable status to be set for the given design variable Default value: t

Value Returned

t	The design variable was enabled or disabled for the test.
nil	The command was unsuccessful.

Examples

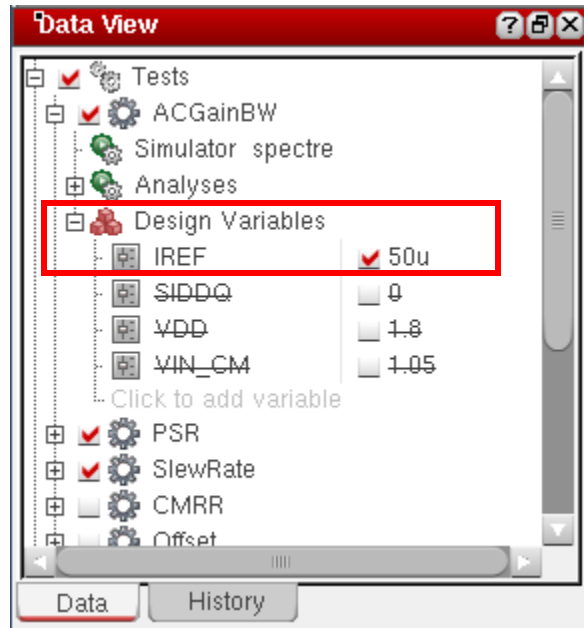
The following example enables a design variable, IREF, for the ACGainBW test.

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

Virtuoso ADE SKILL Reference

Variable Functions

The check box next to the `IREF` variable in the *Design Variables* list for the `ACGainBW` test is selected.



Related Topics

[axlGetEnabledGlobalVarPerTest](#)

maeExportDesignVariables

```
maeExportDesignVariables(  
    t_fileName  
    [ ?exportTest t_exportTest ]  
    [ ?all g_all ]  
)  
=> t / nil
```

Description

Exports the design variables from the specified test to the given CSV file.

Arguments

<i>t_fileName</i>	Name of the CSV file.
<i>?exportTest t_exportTest</i>	Name of the test for which you want to export variables. The default value is " ".
<i>?all g_all</i>	Specifies whether to export the design variables for all tests to the given CSV file. Valid values are <i>t</i> and <i>nil</i> . The default value is <i>nil</i> .

Value Returned

<i>t</i>	The design variables are successfully exported to the given CSV file.
<i>nil</i>	The design variables are not exported to the given CSV file.

Examples

Exports design variables from test *TRAN* to the CSV file *TRAN.csv*.

```
maeExportDesignVariables("TRAN.csv" ?exportTest "TRAN")
```

Exports design variables from all tests to the CSV file *allTests.csv*.

```
maeExportDesignVariables("allTests.csv" ?all t)
```

Virtuoso ADE SKILL Reference

Variable Functions

Related Topics

[maelImportDesignVariables](#)

maeImportDesignVariables

```
maeImportDesignVariables(  
    t_fileName  
    [ ?importFromTest t_importFromTest ]  
    [ ?importToTest t_importToTest ]  
    [ ?all g_all ]  
)  
=> t / nil
```

Description

Imports the design variables from the specified CSV file to the given test.

Arguments

<i>t_fileName</i>	Name of the CSV file from which variables are to be imported.
<i>?importFromTest t_importFromTest</i>	Name of the source test. If the CSV file contains variables from multiple tests, only the variables for the source test are imported. The default value is " ".
<i>?importToTest t_importToTest</i>	Name of test to which variables are to be imported. The default value is " ".
<i>?all g_all</i>	Specifies whether to import the design variables from the specified CSV file to all tests found in the Data View assistant. Valid values are <i>t</i> and <i>nil</i>. The default value is <i>nil</i>.

Value Returned

<i>t</i>	The design variables are successfully imported from the specified CSV file.
<i>nil</i>	The design variables are not imported from the specified CSV file.

Virtuoso ADE SKILL Reference

Variable Functions

Examples

The following example imports design variables for test `TRAN1` from `allTests.csv` into test `TRAN`:

```
maeImportDesignVariables("allTests.csv" ?importFromTest "TRAN1" ?importToTest  
"TRAN")
```

The following example imports the design variables from `allTests.csv` for the tests found in the Data View assistant.

```
maeImportDesignVariables("allTests.csv" ?all t)
```

Related Topics

[maeExportDesignVariables](#)

Parameter Functions

Use the following SKILL functions to work with the device parameters in the setup database of a maestro cellview:

[axlGetParameters](#)

[axlGetParameter](#)

[axlGetParameterValue](#)

[axlGetAllParametersDisabled](#)

[axlRegisterCustomDeviceFilter](#)

[axlSetParameter](#)

[axlSetAllParametersDisabled](#)

axlGetParameters

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

Description

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

Arguments

<i>x_mainSDB</i>	Handle to the setup database.

Value Returned

<i>l_parameterPaths</i>	List of paths to all the parameters found in the setup database.
<i>nil</i>	No parameters are found.

Examples

The following example retrieves a list of paths to all the parameters in the current session.

```
s1 = axlGetWindowSession()  
=> "session0"  
  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
  
axlGetParameters(x_mainSDB)  
("Two_Stage_Opamp/OpAmp/schematic/M10/l" "Two_Stage_Opamp/OpAmp/schematic/M10/m"  
"Two_Stage_Opamp/OpAmp/schematic/M9/l"  
)
```

Related Topics

[Working with Device Instance Parameters](#)

axlGetParameter

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

Description

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

Arguments

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

Value Returned

<i>x_parameterHandle</i>	Handle to the parameter.
0	Unsuccessful, when the parameter is not found.

Examples

The following example retrieves the handle to the specified parameter and to use that handle to delete the parameter.

```
s1 = axlGetWindowSession()  
=> "session0"  
  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
  
axlGetParameters(x_mainSDB)  
("Two_Stage_Opamp/OpAmp/schematic/M10/l" "Two_Stage_Opamp/OpAmp/schematic/M10/m")  
  
axlGetParameter(x_mainSDB "Two_Stage_Opamp/OpAmp/schematic/M10/l")  
=> 2397  
  
axlRemoveElement( 2397)  
=> t
```

Virtuoso ADE SKILL Reference

Parameter Functions

Related Topics

[Working with Device Instance Parameters](#)

axlGetParameterValue

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

Description

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

Arguments

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

Value Returned

<i>t_parameterValue</i>	Value of the given parameter.
<i>nil</i>	Setup database handle to the parameter not found.

Examples

The following example returns the value of the given parameter.

```
s1 = (axlGetWindowSession)  
=> "session0"  
  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
  
axlGetParameters(x_mainSDB)  
=> ("Two_Stage_Opamp/OpAmp/schematic/M10/l"  
"Two_Stage_Opamp/OpAmp/schematic/M10/m" "Two_Stage_Opamp/OpAmp/schematic/M6/fw")  
  
axlGetParameterValue(x_mainSDB "Two_Stage_Opamp/OpAmp/schematic/M10/l")  
=> "500n"
```

Related Topics

Working with Device Instance Parameters

axlGetAllParametersDisabled

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

Description

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

Arguments

<i>x_mainSDB</i>	Setup database handle.

Value Returned

t	The <i>Parameters</i> check box in the Data View assistant pane is cleared.
nil	The <i>Parameters</i> check box in the Data View assistant pane is selected.

Examples

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

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

Related Topics

[Enabling and Disabling Device Parameters](#)

axlRegisterCustomDeviceFilter

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

Description

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

The default filters are *Default*, *CDF Parameters*, and *CDF Editable*. To add an additional filter, use the `axlRegisterCustomDeviceFilter` function.

Arguments

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

Value Returned

<code>t</code>	Successful registration.
<code>nil</code>	Unsuccessful registration.

Examples

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

First, define a custom filter. For example:

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

Virtuoso ADE SKILL Reference

Parameter Functions

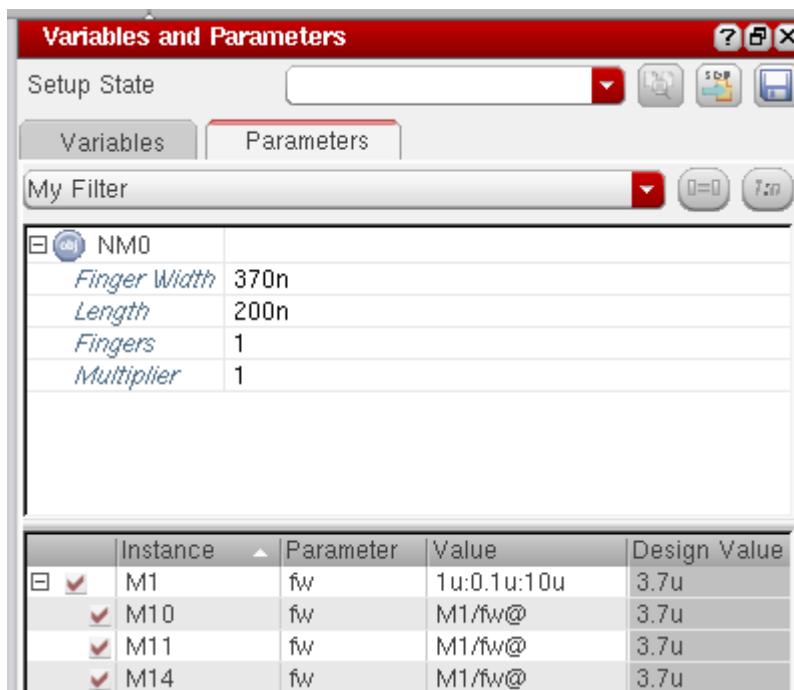
```
(list "m" (get inst "m"))
)
)
```

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

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

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

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



Related Topics

[Variables and Parameters assistant pane](#)

[Working with Device Instance Parameters](#)

axlSetParameter

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

Description

Sets a value for the specified device parameter.

Arguments

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

Value Returned

<i>t</i>	The value of the device parameter is successfully set.
<i>nil</i>	Unsuccessful status.

Examples

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

```
s1 = (axlGetWindowSession)  
=> "session0"  
  
x_mainSDB=axlGetMainSetupDB(s1 )  
=> 1001  
  
axlGetParameters(x_mainSDB)  
=> ("Two_Stage_Opamp/OpAmp/schematic/M10/l"  
"Two_Stage_Opamp/OpAmp/schematic/M10/m" "Two_Stage_Opamp/OpAmp/schematic/M6/fw" )  
  
axlSetParameter(x_mainSDB "Two_Stage_Opamp/OpAmp/schematic/M10/m" "2")  
=> t
```

Virtuoso ADE SKILL Reference

Parameter Functions

Related Topics

[Working with Device Instance Parameters](#)

axlSetAllParametersDisabled

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

Description

Sets the status of the *Parameters* check box in the Data View assistant pane in ADE Explorer and ADE Assembler.

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>g_enableStatus</i>	Option for setting the status of the <i>Parameters</i> check box. ■ 0: Selects the <i>Parameters</i> check box. ■ 1: Deselects the <i>Parameters</i> check box.

Value Returned

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

Examples

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

```
s1 = (axlGetWindowSession)  
=> "session1"  
  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
  
axlSetAllParametersDisabled(x_mainSDB 0)  
t
```

Virtuoso ADE SKILL Reference

Parameter Functions

Related Topics

[Working with Device Instance Parameters](#)

Model Functions

This topic lists the SKILL functions that let you work with the model files associated with the corners in the setup. You can use these functions to add a model file or model group to a corner, to modify their details, such as their section list, test name, or block name, or to get these details for the existing model files.

[axlAddModelPermissibleSectionLists](#)

[axlGetModel](#)

[axlGetModelBlock](#)

[axlGetModelFile](#)

[axlGetModelGroup](#)

[axlGetModelGroupName](#)

[axlGetModelGroups](#)

[axlGetModelPermissibleSectionLists](#)

[axlGetModelSection](#)

[axlGetModelSections](#)

[axlGetModelTest](#)

[axlGetModels](#)

[axlPutModel](#)

[axlPutModelGroup](#)

[axlSetModelBlock](#)

[axlSetModelFile](#)

Virtuoso ADE SKILL Reference

Model Functions

[axlSetModelGroupName](#)

[axlSetModelPermissibleSectionLists](#)

[axlSetModelSection](#)

[axlSetModelTest](#)

Related Topics

[Functions to Work with Model Files](#)

axlAddModelPermissibleSectionLists

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

Description

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

Arguments

<i>x_handleModel</i>	Handle to the model imported from a PCF file This should be a model file in the main setup database.
<i>l_sectionNames</i>	List of section names to be added to the list of permissible sections

Value Returned

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

Examples

Adds a new section name to the permissible section name list for the `gpd_k090.scs` model imported from a PCF file.

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

Virtuoso ADE SKILL Reference

Model Functions

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

```
mh1=axlGetModel(sdb "gpd090.scs")
=> 2323
```

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

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

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

; adding one more section name to the permissible sections list

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

Related Topics

[axlGetCorner](#)

[axlGetCorners](#)

[axlGetModel](#)

[axlSetModelPermissibleSectionLists](#)

[axlGetModelPermissibleSectionLists](#)

[LimitModelSections](#)

[Importing Corners from Customization File](#)

axlGetModel

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

Description

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

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

Arguments

<i>x_cornerHandle</i>	Handle to a corner in the main setup database.
<i>t_modelName</i>	Model file name

Value Returned

<i>x_modelFile</i>	Handle to the specified model file
0	Unsuccessful operation

Examples

Returns the handle to the `myModel.scs` model file associated with corner C0.

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

Related Topics

[axlGetCorner](#)

[axlPutModel](#)

axlGetModelBlock

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

Description

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

Arguments

<i>x_modelHandle</i>	Handle to a model.

Value Returned

<i>t_blockName</i>	The name of block associated with the specified model.
<i>nil</i>	Indicates an error.

Examples

Returns the block name for `myModel.scs`.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
cornerHandle=axlGetCorner(x_mainSDB "C0")  
=> 5497  
modelHandle=axlGetModel(cornerHandle "myModel.scs")  
=> 5505  
axlGetModelBlock(modelHandle)  
=> "Global"  
  
; the return value indicates that the model is associated with the entire test and  
not to a specific block
```

Related Topics

[axlGetModel](#)

[axlSetModelBlock](#)

Virtuoso ADE SKILL Reference

Model Functions

axlGetCorner

axlGetModelFile

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

Description

Returns the model file path for the specified model.

Arguments

<i>x_modelHandle</i>	Handle to a model

Value Returned

<i>t_modelFile</i>	The model file name is returned along with the full path. If the model file is imported from test, only the file name is returned.
nil	The model file name and path are not returned.

Examples

Returns the file details for the specified model file.

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

Related Topics

[axlGetModel](#)

[axlSetModelFile](#)

axlGetModelGroup

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

Description

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

Arguments

<i>x_mainSDB</i>	Handle to the setup database or a checkpoint history
<i>t_modelGroupName</i>	Model group name

Value Returned

<i>x_modelGroup</i>	The model group handle is returned.
<i>nil</i>	The model group handle is not returned.

Examples

Returns a handle to the model group mG1 in the current setup database.

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

The `axlGetModel` function works only with the corner handle whereas the `axlGetModelGroup` works with setup database or checkpoint handle.

Related Topics

[axlPutModelGroup](#)

axlGetModelGroupName

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

Description

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

Arguments

<i>x_cornerHandle</i>	Corner handle

Value Returned

<i>t_modelGroupName</i>	Names of model groups associated with the specified corner.

<i>nil</i>	Names of model groups associated with the specified corner are not returned.

Examples

Returns the names of model groups associated with the specified corner.

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

Virtuoso ADE SKILL Reference

Model Functions

Related Topics

[axlSetModelGroupName](#)

axlGetModelGroups

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

Description

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

Arguments

<i>x_mainSDB</i>	Handle to the setup database or a checkpoint history.

Value Returned

<i>l_modelGroups</i>	The list of model group names in the setup database is returned.
<i>nil</i>	The list of model group names is not returned.

Examples

Returns a list of model groups in the current setup database.

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

Related Topics

[axlPutModelGroup](#)

axlGetModelPermissibleSectionLists

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

Description

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

Arguments

<i>x_handleModel</i>	Handle to the model imported from a PCF file This should be a model file in the main setup database.

Value Returned

<i>l_sectionNames</i>	The list containing the setup handle and all the section names is returned.
<i>nil</i>	Indicates an error.

Examples

Returns the permissible section name list for the model file `gpdk090.scs`.

```
; get the handle to the main setup database  
  
sdb=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
; get the handle to the model imported from the PCF file.  
  
;Note: It is important to provide the handle to the main setup database in the  
;function call given below.  
  
mh1=axlGetModel(sdb "gpdk090.scs")  
=> 2323  
  
; View the existing permissible sections list for the given model.  
  
axlGetModelPermissibleSectionLists(mh1)  
=> (2326 ("TT_slv"))  
; the returned value shows that the permissible sections list for the model  
; includes only one section, TT_slv.  
  
axlAddModelPermissibleSectionLists(mh1 ( "FF_slv" ))
```

Virtuoso ADE SKILL Reference

Model Functions

```
=> (2603 2604)

; adding one more section name to the permissible sections list

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

Related Topics

[axlSetModelPermissibleSectionLists](#)

[axlAddModelPermissibleSectionLists](#)

axlGetModelSection

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

Description

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

Arguments

<i>x_modelHandle</i>	Handle to the model

Value Returned

<i>t_sectionName</i>	The name of the model section is returned.
<i>nil</i>	The name of the model section is not returned.

Examples

Returns the model section for `gpdk.scs` model file.

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

Related Topics

[axlGetModel](#)

[axlGetCorner](#)

[axlSetModelSection](#)

axlGetModelSections

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

Description

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

Arguments

<i>t_fileName</i>	Name of the model file

Value Returned

<i>l_sectionNames</i>	The list of the model sections is returned.
<i>nil</i>	The list of the model sections is not returned.

Examples

Returns the list of model sections for the model file `gpdk.scs`.

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

Related Topics

[axlGetModelaxlGetCorner](#)

[axlSetModelSection](#)

axlGetModelTest

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

Description

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

Arguments

<i>x_modelHandle</i>	Handle to a model

Value Returned

<i>t_testName</i>	Name of the test associated with the specified model is returned. In default mode, the function returns <code>All</code> . In MTS mode, it returns the name of the test of the block with which the model file is associated.
<i>nil</i>	Name of the test associated with the specified model is not returned.

Examples

Returns the name of the test associated with the model `model1.scs`.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
=> "C0"  
cornerHandle=axlGetCorner(x_mainSDB "C0")  
=> 1918  
model = axlGetModel(cornerHandle "model1.scs")  
=> 1311  
axlGetModelTest(model)  
"All"  
; the return value indicates that the model file is associated with all the tests  
in the adexl view.
```

Virtuoso ADE SKILL Reference

Model Functions

Related Topics

[axlGetModel](#)

[axlSetModelTest](#)

axlGetModels

```
axlGetModels(  
    x_cornerHandle  
)  
=> l_modelFiles / nil
```

Description

Returns a list of model files associated with the given corner.

Arguments

<i>x_cornerHandle</i>	Corner handle

Value Returned

<i>l_modelFiles</i>	The list of model file names is returned.
<i>nil</i>	The list of model file names is not returned.

Examples

Returns the model files associated with corner C0.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
sdb=axlGetMainSetupDB(axlGetWindowSession())  
cornerHandle=axlGetCorner(sdb "C0")  
=> 1918  
model = axlGetModels(cornerHandle)  
(1310  
    ("model11.scs" "model12.scs" "model13.scs")  
)
```

axlPutModel

```
axlPutModel(  
    x_cornerHandle  
    t_modelFileName  
    [ ?testName t_testName ]  
    [ ?blockName t_blockName ]  
)  
=> x_modelHandle / nil
```

Description

Adds a model file to the specified corner.

Virtuoso ADE SKILL Reference

Model Functions

Arguments

x_cornerHandle Handle to the corner for which the model file is to be added.

t_modelFileName Name of the model file

You must specify a unique name for the model file. For example, if `gpdK090.scs` is already present, you can specify `gpdK090.scs:1`.

?testName *t_testName*

Name of the test with which the model is to be associated.

Default value: All

By default, a model file is associated with all the tests. In the Multi-Technology Simulation (MTS) mode, use this argument to specify the test with which the model is to be associated.

?blockName *t_blockName*

Name of the design block.

Default value: Global

By default, a model file is used for the complete design. In the MTS mode, use this argument to specify the design block for which the model is to be used.

Value Returned

x_modelHandle The handle to the model file is returned.

nil The handle to the model file is not returned.

Examples

Adds two models from different model files for the `hightemp` corner. These models are applied to all the tests and blocks in the setup database.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
cCornerHandle=axlGetCorner(x_mainSDB "C0")
; adding model
modelHandle=axlPutModel(cCornerHandle "gpdK090.scs")

;adding file for the model
axlSetModelFile(modelHandle "../models/gpdK090.scs")

; specify a section to be used for the corner
```

Virtuoso ADE SKILL Reference

Model Functions

```
axlSetModelSection(modelHandle "ff")
```

```
; enable the model for the corner  
axlSetEnabled(modelHandle t)
```

Lets you use the `axlPutModel` function in MTS mode to add a model `gpdk045.scs` for the hightemp corner, and apply it to the given test and block.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
cornerHandle=axlGetCorner(x_mainSDB "hightemp")  
modelHandle=axlPutModel(cornerHandle "gpdk045" ?testName "MTS_test:testbench:1"  
?blockName "design_45 inv")
```

```
=>1319  
axlSetModelFile(modelHandle "../models/gpdk045.scs")  
; specifies path to the model file  
axlSetModelSection(modelHandle "ff")  
axlSetEnabled(modelHandle t)
```

Related Topics

[axlGetMainSetupDB](#)

[axlGetWindowSession](#)

[axlSetModelFile](#)

[axlSetModelSection](#)

axlPutModelGroup

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

Description

Adds a model group to the setup database or returns the handle to the model group if it already exists.

Arguments

<i>x_mainSDB</i>	Handle to the setup database or checkpoint history.

<i>t_modelGroupName</i>	Model group name

Value Returned

<i>x_modelGroup</i>	The handle to the model group is returned.
nil	The handle to the model group is not returned.

Examples

Adds a model group mG1 to the current setup database.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
axlPutModelGroup(x_mainSDB "FF")  
=> 1435
```

Related topics

[axlGetModelGroup](#)

axlSetModelBlock

```
axlSetModelBlock(  
    x_modelHandle  
    t_blockName  
)  
=> x_modelBlock / nil
```

Description

Sets the name of the block for the specified model. By default, a model is associated with all the design blocks and is set as `Global`. In the MTS mode, a model is associated with a specific MTS block, so you need to use this function to specify the name of that block.

Alternatively, you can use the [axlPutModel](#) function to set the block name while associating a model to a corner.

Arguments

<i>x_modelHandle</i>	Handle to a model
<i>t_blockName</i>	Block name

Value Returned

<i>x_modelBlock</i>	The handle to the model block element is returned.
<i>nil</i>	The handle to the model block element is not returned.

Examples

Sets the test name and a block name for a model to be associated with an MTS block.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
sdb_corner=axlGetCorner(x_mainSDB "C2")  
=> 1098  
modelHandle=axlPutModel(sdb_corner "fastModel")  
=> 1115  
axlSetModelFile(modelHandle "../gpdK045/models/spectre/gpdK045.scs")  
=> 1116  
axlSetModelSection(modelHandle "ff")  
=> 1119  
;  
;setting test name and block name for the design block, design_45 inv, for  
;the test MTS_test:testbench:1  
axlSetModelTest(modelHandle "MTS_test:testbench:1")
```


Virtuoso ADE SKILL Reference

Model Functions

```
=> 1117  
axlSetModelBlock(modelHandle "design_45 inv")  
=> 1118  
axlSetEnabled(modelHandle t)
```

Related Topics

[axlGetModel](#)

[axlGetModelBlock](#)

[axlSetModelTest](#)

[axlPutModel](#)

axlSetModelFile

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

Description

Sets the model file for the specified model.

Arguments

<i>x_modelHandle</i>	Handle to a model
<i>t_modelFile</i>	Model file path and name

Value Returned

<i>x_modelFile</i>	Handle to the model file element

<i>nil</i>	Unsuccessful operation

Examples

Sets the model file name for a model.

```
sdb=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
sdb_corner=axlGetCorner(sdb "C1")  
=> 1098  
modelHandle=axlPutModel(sdb_corner "fastModel")  
=> 1115  
axlSetModelFile(modelHandle  
"/../adex1/MTS_testcase/gpdk045/models/spectre/gpdk045.scs")
```

Virtuoso ADE SKILL Reference

Model Functions

```
=> 1116  
axlSetModelSection(modelHandle "fs")  
=> 1119
```

Related Topics

[axlGetModel](#)

[axlGetModelFile](#)

axlSetModelGroupName

```
axlSetModelGroupName(  
    x_corner  
    t_modelGroupName  
)  
=> x_modelGroup / nil
```

Description

Sets or associates the given model group with the specified corner.

Arguments

<i>x_corner</i>	Corner handle
<i>t_modelGroupName</i>	Model group name

Value Returned

<i>x_modelGroup</i>	The handle to the model group name element is returned.
nil	The handle to the model group name element is not returned.

Examples

Assigns a model group to the given corner.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
sdb_corner=axlGetCorner(x_mainSDB "C0")  
=> 7109  
axlSetModelGroupName(sdb_corner "mG2")  
=> 7192
```

Related Topics

[axlGetModelGroupName](#)

axlSetModelPermissibleSectionLists

```
axlSetModelPermissibleSectionLists(  
    x_modelHandle  
    l_sectionNames  
)  
=> l_sectionHandles / nil
```

Description

Sets a permissible section list for the given model extracted from a PCF file. If a model file includes many sections out of which only a limited number of sections are of relevance to your testbench, you can create a permissible section list for that model file. If the `LimitModelSections` environment variable is set to `LimitedList`, while displaying the list of section names in the Corners Setup UI, the tool checks the permissible section list for a model file and shows only the relevant names.

Arguments

<i>x_modelHandle</i>	Handle to the model imported from a PCF file This should be a model file in the main setup database.
<i>l_sectionNames</i>	Names of all the sections in the given model file that are allowed to be selected.

Value Returned

<i>l_sectionHandles</i>	The list containing handles to the permissible sections is returned.
<i>nil</i>	The list containing handles to the permissible sections is not returned.

Examples

Sets a permissible section name list for the `gpdK090.scs` model imported from a PCF file.

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

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

Virtuoso ADE SKILL Reference

Model Functions

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

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

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

; creating a new permissible sections list for this model file

axlSetModelPermissibleSectionLists(mh1 ( "FS_slv" "FF_slv" ))
=> (2606 2607)
; the return value shows that now the permissible sections list for the model
; includes two section names

axlGetModelPermissibleSectionLists(mh1)
=> (2326 ("FS_slv" "FF_slv"))
```

Related Topics

[axlAddModelPermissibleSectionLists](#)

[axlGetModelPermissibleSectionLists](#)

[LimitModelSections](#)

axlSetModelSection

```
axlSetModelSection(  
    x_modelHandle  
    t_sectionName  
)  
=> x_modelSection / nil
```

Description

Sets the section name for the specified model.

Arguments

<i>x_modelhandle</i>	Model handle
<i>t_sectionName</i>	Section name associated with the specified model

Value Returned

<i>x_modelSection</i>	The handle to the model section element is returned.
<i>nil</i>	The handle to the model section element is not returned.

Examples

Sets the model section for a model file.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
sdb_corner=axlGetCorner(x_mainSDB "C1")  
=> 1098  
modelHandle=axlPutModel(sdb_corner "fastModel")  
=> 1115  
axlSetModelFile(modelHandle "../gpdK045/models/spectre/gpdK045.scs")  
=> 1116  
axlSetModelSection(modelHandle "fs")  
=> 1119
```

Related Topics

[axlGetModel](#)

[axlGetModelSection](#)

axlSetModelTest

```
axlSetModelTest(  
    x_modelHandle  
    t_testName  
)  
=> x_modelTest / nil
```

Description

Sets the name of the test to be associated with the specified model. By default, a model is associated with all the tests. In case of MTS mode, you need to associate the model with a specific test. Alternatively, you can use the [axlPutModel](#) function to specify the test name while adding a model.

Arguments

<i>x_modelHandle</i>	Handle to a model.
<i>t_testName</i>	Name of the test associated with the specified model.

Value Returned

<i>x_modelTest</i>	The handle to the model test element is returned.
<i>nil</i>	The handle to the model test element is not returned.

Examples

Sets the test name and a block name for a model to be associated with an MTS block.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())  
=> 1001  
sdb_corner=axlGetCorner(x_mainSDB "C2")  
=> 1098  
modelHandle=axlPutModel(sdb_corner "fastModel")  
=> 1115  
axlSetModelFile(modelHandle "../gpdK045/models/spectre/gpdK045.scs")  
=> 1116  
axlSetModelSection(modelHandle "ff")  
=> 1119  
;  
;setting test name and block name for an MTS block, design_45 inv
```


Virtuoso ADE SKILL Reference

Model Functions

```
;
axlSetModelTest(modelHandle "MTS_test:testbench:1")
=> 1117
axlSetModelBlock(modelHandle "design_45 inv")
=> 1118
axlSetEnabled(modelHandle t)
```

Related Topics

[axlGetModel](#)

[axlGetModelTest](#)

[axlSetModelBlock](#)

[axlGetCorner](#)

[axlPutModel](#)

Functions to Work with Model Files

To work with the model files associated with a testbench, you can use the following SKILL functions:

- asiAddModelLibSelection
- asiGetModelLibSelectionList
- asiGetModelLibFile
- asiGetModelLibSection

A few examples that show how to add or view the model files associated with the tests are given below.

Example 1

Returns the model file name for the first model associated with a test:

```
session = (axlGetWindowSession (hiGetCurrentWindow))
=> "session0"
x_mainSDB = (axlGetMainSetupDB session)
=> 1001

axlGetTests(x_mainSDB)
=> (1004
   ("opamp090:full_diff_opamp_AC:1" "opamp090:full_diff_opamp_TRAN:1")
)

testSession = axlGetToolSession(axlsession, "opamp090:full_diff_opamp_AC:1")
=> sevSession1

oSession = sevEnvironment(testSession)
=> stdobj@0x1c030668

modellist = asiGetModelLibSelectionList( oSession )
=> (("gpd090.scs" "NN") ("fastmodel.scs" "FF"))

asiGetModelLibFile(car(modellist))
=> "gpd090.scs"
```

Example 2

Removes the existing model files for a test, if any, and set a model file for it:

```
testName="AC"
=> "AC"

; get the handle to the session
session=axlGetWindowSession()
=> "session0"
```

Virtuoso ADE SKILL Reference

Model Functions

```
; get the handle to the ADE L or test session
testSession=axlGetToolSession(session testName)
=> sevSession1
oSession=sevEnvironment(testSession)
=> stdobj@0x1e213620

; remove any existing model files for the test
asiSetEnvOptionVal(oSession 'modelFiles (list (list "" "")))
=> t
((" " ""))

; add a model file to the test

asiAddModelLibSelection(oSession "testModelFile1.scs" "ss")
=> t

; you can use the asiGetModelLibSelectionList function to get the list of all
; the model files attached to the test

modelList = asiGetModelLibSelectionList( oSession )
printf("\tModel list = %L\n" modelList )
=>Model list= ("../models/spectre/gpdk045.scs" "mc") ("testModelFile1.scs" "ss") )
=> t
```

Example 3

Returns the details of model files associated with all the tests:

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

x_mainSDB = (axlGetMainSetupDB session)
> 1001

(foreach testName (cadr axlGetTests(x_mainSDB) )
  printf( "Test %s\n" testName)
  testSession = axlGetToolSession(session testName)
  oSession = sevEnvironment(testSession)
  modelList = asiGetModelLibSelectionList( oSession )
  printf("\tModel list = %L\n" modelList )
)
>Test AC
    Model list =
(("../adegx1/VAD_workshop_616/gpdk045_v_3_5/gpdk045/../../models/spectre/gpdk045.scs
" "mc") ("testModelFile1.scs" "ss") )
Test TRAN
    Model list = (("gpdk045.scs" "mc") ("ind.scs" "TT") ("cap.scs" "TT") )
```

Virtuoso ADE SKILL Reference

Model Functions

Output Functions

This topic lists the functions that let you work with the outputs.

ALIAS

[axlAddOutputExpr](#)

[axlAddOutputs](#)

[axlAddOutputsColumn](#)

[axlAddOutputSignal](#)

[axlDeleteOutput](#)

[axlDeleteOutputsColumn](#)

[axlGetCustomOutputResultMenu](#)

[axlGetOutputNotation](#)

[axlGetOutputSignificantDigits](#)

[axlGetOutputSuffix](#)

[axlGetOutputUnits](#)

[axlGetOutputUserDefinedData](#)

[axlGetTemperatureForCurrentPointInRun](#)

[axlGetUserDefinedOutputsColumns](#)

[axlOutputResult](#)

[axlOutputsExportToFile](#)

[axlOutputsImportFromFile](#)

Virtuoso ADE SKILL Reference

Output Functions

[axlPutOutputNotation](#)

[axlPutOutputSignificantDigits](#)

[axlPutOutputSuffix](#)

[axlPutOutputUnits](#)

[axlRenameOutputsColumn](#)

[axlSetOutputUserDefinedData](#)

[calcVal](#)

[measureAcross](#)

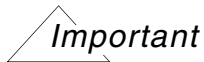
ALIAS

```
ALIAS (
    t_globalVar
)
=> t_globalVarValue / nil
```

Description

Can be used in two scenarios- to give alias or alternate names to long net names or instance name paths, and to return the first value of a waveform result, for non-swept variables.

For the second purpose, ALIAS provides a wrapper around the VAR function to output a scalar value by returning the first Y value from the VAR waveform. Therefore, you can replace VAR with ALIAS where VAR returns a waveform but you need a scalar value for an output or expression. ALIAS returns the first Y value of the waveform returned by VAR.



You cannot use ALIAS function in the following cases:

- ☐ In pre-processing. It can be used only for the processed simulation results.
- ☐ To access sweep variables. It returns an incorrect value for sweep variables.

Arguments

<i>t_globalVar</i>	Name of the global variable.

Value Returned

<i>t_globalVarValue</i>	Value of the global variable.
<i>nil</i>	Unsuccessful operation.

Examples

The following example demonstrates how to use a net/terminal name with an alias name:

```
mynet ="/I0/I1/M1/M2/M3/net2"
db(vh('hbac ALIAS("mynet") '((0 -1))))
```

Virtuoso ADE SKILL Reference

Output Functions

Functions such as VT, VF, or VH expect a net/terminal name, and perform unsuccessfully with VAR. Instead, it is recommended to use ALIAS with such functions.

The following example demonstrates when to use ALIAS within a calculator function to provide a single scalar value for outputs that return a waveform:

```
X_Freq = 1M
value(db(vh('hba "/X4" ' ((0 -1)))) ALIAS("X_Freq"))
```

Note: The value function expects a scalar value in the second argument. If we use VAR("X_Freq"), it returns a scalar value when evaluating at the leaf level, but a waveform at the root level, where all the values will be 1M. Therefore, it is recommended to use ALIAS because it returns a scalar at the leaf level, and the first value from the waveform, when evaluating at the root level.

When to avoid using ALIAS?

In example2 shown above, if the X_Freq variable is swept with three different values, as shown below, ALIAS returns the value, 1M:

```
X_Freq = 1M 2M 3M
value(db(vh('hba "/X4" ' ((0 -1)))) ALIAS("X_Freq"))
```

Here, using ALIAS("X_Freq") would only provide the first value from the swept variable, which would be incorrect.

Therefore, it is strongly recommended not to use ALIAS with sweep variables, as it returns incorrect result values.

axlAddOutputExpr

```
axlAddOutputExpr(
    t_sessionName
    t_testName
    t_outputName
    [ ?expr t_expr ]
    [ ?evalType t_evalType ]
    [ ?exprDPLs l_exprDPLs ]
    [ ?plot g_plot ]
    [ ?save g_save ]
)
=> t / t_error
```

Description

Adds an output expression to a test setup.

Arguments

<i>t_sessionName</i>	Name of session
<i>t_testName</i>	Name of test
<i>t_outputName</i>	Name to be assigned to the expression output
<i>?expr t_expr</i>	Expression to be used to calculate the output Default value: " "
<i>?exprDPLs l_exprDPLs</i>	A disembodied property list that provides details of expressions to be added in batch mode. Each output structure can be specified in the following format: 'outputName value 'expr value 'evalType value 'plot value 'save value) Default value: nil

Virtuoso ADE SKILL Reference

Output Functions

<code>?evalType t_evalType</code>	Evaluation type of the expression Possible values: ■ <code>"point"</code>: Calculates the expression for every design point ■ <code>"corners"</code>: Calculates the expression across all the corners ■ <code>"sweeps"</code>: Calculates the expression across all the sweep points ■ <code>"maa"</code>: Calculates the expression across all the corners and sweep points Default value: <code>"point"</code> <code>"sweeps"</code> and <code>"maa"</code> are applicable only for the maestro cellviews created using ADE Assembler.
<code>?plot g_plot</code>	Specifies whether to plot the output. Default value: <code>t</code>
<code>?save g_save</code>	Specifies whether to save the output. Default value: <code>t</code>

Value Returned

<code>t</code>	Successful addition of output to the test
<code>t_error</code>	Returns an error message if unsuccessful.

Examples

The following example code shows how to use the `axlAddOutputExpr` function to add outputs for a test:

```
session = axlGetWindowSession()
=>"session0"
; returns handle to the current session
axlAddOutputExpr(session "AC" "output1" )
=> t
; the above statement adds an expression output, but the expression to be evaluated
; is not specified. The evaluation type of this output is 'point'
```

Virtuoso ADE SKILL Reference

Output Functions

```
axlAddOutputExpr(session, "AC" "SRp" ?expr "ymax(deriv(VT(\"/OUT\")))" ?evalType  
"corners" ?plot t ?save t)
```

; the above statement adds an expression output that evaluates the given expression
; across corners

The following example code shows how to use the `axlAddOutputExpr` function to add outputs in batch mode:

```
session = axlGetWindowSession()  
; gets a handle to the session
```

```
;; define a DPL for the first output to be added  
dpl='(nil)  
putprop(dpl "BW" 'outputName)  
putprop(dpl "bandwidth(VT(\"/out\") 3 \"low\")" 'expr)  
putprop(dpl t 'plot)  
; append it to exprs  
exprs=append(nil list(dpl))
```

```
; define a DPL for the second output to be added
```

```
dp11='(nil)  
putprop(dp11 "Gain" 'outputName)  
putprop(dp11 "value(dB20(mag(VF(\"/OUT\"))))" 'expr)  
putprop(dp11 t 'plot)  
; append it to exprs
```

```
exprs=append(exprs list(dp11))
```

```
; add the outputs
```

```
axlAddOutputExpr(session "opamp090:full_diff_opamp_AC:1" "" ?exprDPLs exprs)
```

```
;; this script adds two outputs, BW and Gain, to the setup
```

axlAddOutputs

```
axlAddOutputs(  
    l_outputNames  
)  
=> t / nil
```

Description

Defines one or more output measures in an OCEAN script.

It is optional to specify the `axlAddOutputs` command. The tool parses the script for `axlOutputResult` commands to extract derived measure names. You can specify this command anywhere in the script.

Arguments

<i>l_outputNames</i>	List of output names

Value Returned

<i>t</i>	Successful operation
<i>nil</i>	Unsuccessful operation

Examples

The following command defines two outputs, `maxOfOut` and `minOfOut`. You can set the values of these outputs using `axlOutputResult`.

```
axlAddOutputs( ' ("maxOfOut" "minOfOut") )  
=> t
```

axlAddOutputsColumn

```
axlAddOutputsColumn(  
    x_mainSDB  
    t_columnName  
)  
=> t / nil
```

Description

Adds a new user-defined column to the outputs table.

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.
<i>t_columnName</i>	Name of the user-defined column to be added to the outputs table

Value Returned

<i>t</i>	Successful addition of the specified column.
<i>nil</i>	Unsuccessful operation

Examples

The following example demonstrates how to add a user-defined column to the Outputs table.

```
session = axlGetWindowSession()  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
axlAddOutputsColumn( x_mainSDB "Spec Description")  
=>t
```

axlAddOutputSignal

```
axlAddOutputSignal(
    t_sessionName
    t_testName
    t_signalName
    [ ?type t_outputType ]
    [ ?outputName t_outputName ]
    [ ?plot g_plot ]
    [ ?save g_save ]
)
=> t / t_error
```

Description

Adds a signal to a test setup.

Arguments

<i>t_sessionName</i>	Name of session
<i>t_testName</i>	Name of test
<i>t_signalName</i>	Name of the signal to be added to the outputs of the given test.
<i>?type t_outputType</i>	Type of the signal. Possible values: <code>terminal</code> and <code>net</code> . Default value: <code>"net"</code> " <code>sweeps</code> " and " <code>maa</code> " are applicable only for the maestro cellviews created using ADE Assembler.
<i>?outputName t_outputName</i>	Name to be assigned to the signal output. Default value: <code>" "</code>
<i>?plot g_plot</i>	Specifies whether to plot the output. Default value: <code>t</code>
<i>?save g_save</i>	Specifies whether to save the output. Default value: <code>t</code>

Virtuoso ADE SKILL Reference

Output Functions

Value Returned

<code>t</code>	Successful addition of output to the test
<code>t_error</code>	Returns an error message if unsuccessful.

Examples

The following example code shows how to use the `axlAddOutputExpr` function to add outputs in batch mode:

```
session = axlGetWindowSession()
testname = "voltage_divider:voltage_divider:1"
axlAddOutputSignal(session testname "/net1")
=> t
axlAddOutputSignal(session testname "V0/PLUS" ?type "terminal" ?plot t )
=> t
axlAddOutputSignal(session testname "/net2" ?outputName "Out1" ?type "net")
=> t
```

axlDeleteOutput

```
axlDeleteOutput(  
    t_sessionName  
    t_testName  
    [ ?outputName t_outputName ]  
    [ ?type t_outputType ]  
)  
=> t / t_error
```

Description

Deletes an output from a test setup.

Arguments

<i>t_sessionName</i>	Name of session
<i>t_testName</i>	Name of test
<i>?outputName t_outputName</i>	Name of the output to be deleted. Default value: " "
<i>?type t_outputType</i>	Type of the output. Possible values: <code>signal</code> and <code>expr</code> . Default value: " "

Value Returned

<i>t</i>	Successful deletion of an output from the test
<i>t_error</i>	Returns an error message if unsuccessful.

Examples

The following examples delete the outputs from the specified test.

```
session = axlGetWindowSession()  
testname = "simLib1:sim_top1:1"  
axlDeleteOutput(session testname "/net6")  
axlDeleteOutput(session testname "V0/I/1" ?type "expr")
```


Virtuoso ADE SKILL Reference

Output Functions

```
axlDeleteOutput(session testname "/net5" ?type "signal")  
=> t
```

axlDeleteOutputsColumn

```
axlDeleteOutputsColumn(  
    x_mainSDB  
    t_columnName  
)  
=> t / nil
```

Description

Deletes a user-defined column from the Outputs table.

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.
<i>t_columnName</i>	Name of the user-defined column to be deleted from the Outputs table

Value Returned

<i>t</i>	Successful deletion of the specified column.
<i>nil</i>	Unsuccessful operation.

Examples

The following example deletes a user-defined column, `Comments`.

```
session = axlGetWindowSession()  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
axlGetUserDefinedOutputsColumns(x_mainSDB)  
=> ("Comments" "Spec Description")  
axlDeleteOutputsColumn( x_mainSDB "Comments")  
=>t
```

axlGetCustomOutputResultMenu

```
axlGetCustomOutputResultMenu(  
    [ g_adeSession ]  
)  
=> l_commandDetails
```

Description

A callback function that you can use to add custom commands in the shortcut menu of the Results tab of Virtuoso ADE Explorer and Virtuoso ADE Assembler. These products call this function to get the custom command names and the associated callback functions to be added to the menu.

Arguments

<i>g_adeSession</i>	Handle to the session associated with the test for which you want to add the custom commands.

Value Returned

<i>l_commandDetails</i>	A list containing the command name, callback name, and the value of disable flag. These return values are used by the tool to add the given commands to the shortcut menu and process them when clicked by a user.

Examples

The following example shows how to add a custom command using this function:

```
adeSession=asiGetCurrentSession()  
=> stdobj@0x38a3d128  
  
procedure( axlGetCustomOutputResultMenu( @key adeSession )  
    let((menuList nil) disableFlag)  
    cmd = sprintf(nil "cmenu('%L %L)' adeSession  
asiGetPsfDir(sevEnvironment(adeSession)))  
    menuList = '("testMenu" ,cmd ,disableFlag)  
    menuList  
)  
)
```

Virtuoso ADE SKILL Reference

Output Functions

```
function axlGetCustomOutputResultMenu redefined  
axlGetCustomOutputResultMenu
```

axlGetOutputNotation

```
axlGetOutputNotation(  
    x_sdb  
    t_testName  
    t_outputID  
)  
=> t_notation / " "
```

Description

Returns the notation style set for the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.

Value Returned

<i>t_notation</i>	Notation style of the specified output.
" "	Notation style is not set for the specified output.

Examples

The following example shows that suffix is the notation style for the UGF output of the opamp:OpAmp_lab1AC_top:1 test.

```
axlGetOutputNotation(1001 "opamps:OpAmp_lab1_AC_top:1" "UGF")  
=>"suffix"
```

axlGetOutputSignificantDigits

```
axlGetOutputSignificantDigits(  
    x_sdb  
    t_testName  
    t_outputID  
)  
=> t_digits / 0
```

Description

Returns the number of significant digits set for the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.

Value Returned

<i>t_digits</i>	Number of the significant digits of the specified output.
0	Significant digits are not set for the specified output.

Examples

The following example shows that 6 is set as the significant digits for the output `delay_read1` of the test `tran`.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlGetOutputSignificantDigits(sdb "tran" "delay_read1")  
=> 6
```

axlGetOutputSuffix

```
axlGetOutputSuffix(  
    x_sdb  
    t_testName  
    t_outputID  
)  
=> t_suffix / " "
```

Description

Returns the suffix set to the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.

Value Returned

<i>t_suffix</i>	Suffix of the specified output.
" "	Suffix is not set for the specified output.

Examples

The following example shows that *p* is set as the suffix for the output *delay_read1* of the test *tran*.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlGetOutputSuffix(sdb "tran" "delay_read1")  
=> "p"
```

axlGetOutputUnits

```
axlGetOutputUnits(  
    x_sdb  
    t_testName  
    t_outputName  
)  
=> t_unit / " "
```

Description

Returns the suffix set to the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputName</i>	Name of the output.

Value Returned

<i>t_unit</i>	Unit of the specified output.
" "	Unit is not set for the specified output.

Examples

The following example shows that Hz is set as the unit for the output *freq* of the test *tran*.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlGetOutputUnits(sdb "tran" "freq")  
=> "Hz"
```


axlGetOutputUserDefinedData

```
axlGetOutputUserDefinedData(  
    x_sdb  
    t_testName  
    t_outputName  
    t_columnName  
)  
=> t_unit / " "
```

Description

Returns the value saved in a user-defined column for the given output and test name combination.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputName</i>	Name of the output.
<i>t_columnName</i>	Name of the user-defined column.

Value Returned

<i>t_unit</i>	Value saved in the user-defined column for the specified output.
" "	No value is saved in the user-defined column.

Examples

The following example shows that `delayRead1 < 78p` is set in the user-defined column `myColumn` for the output `delay_read1` of the test `tran`.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlGetOutputUserDefinedData(sdb "tran" "delay_read1" "myColumn")  
=> "delayRead1 < 78p"
```

axlGetTemperatureForCurrentPointInRun

```
axlGetTemperatureForCurrentPointInRun(  
    )  
=> t_temperature / nil
```

Description

Within the OCEAN measurement script, this function allows to access the temperature of the current point in the run.

Arguments

None

Value Returned

<i>t_temperature</i>	Returns the temperature of the current point in the run.
nil	Returns nil otherwise.

Examples

The following example returns the temperature of the current point in run.

```
axlGetTemperatureForCurrentPointInRun()  
=> "27"
```

axlGetUserDefinedOutputsColumns

```
axlGetUserDefinedOutputsColumns(  
    x_sdb  
)  
=> l_columnNames / nil
```

Description

Returns a list of names of the user-defined columns in the specified setup database.

Arguments

<i>x_sdb</i>	Handle to the main setup database.

Value Returned

<i>l_columnNames</i>	A list containing names of the user-defined columns.
<i>nil</i>	No user-defined columns found.

Examples

The following example returns a list containing names of the user-defined columns.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlGetUserDefinedOutputsColumns(sdb)  
=> ("myComments" "specDescription")
```

axlOutputResult

```
axlOutputResult(  
    g_value  
    [ t_outputName ]  
)  
=> t / nil
```

Description

Specifies the value of an output in an OCEAN script file. This function only assigns a value to a measure, but does not add it to the script. You need to use the `axlAddOutputs` function to add or define output measures in an OCEAN script.

It is recommended to initialize the output values before calling this function.

Arguments

<i>g_value</i>	Output value
<i>t_outputName</i>	Name of the output

Value Returned

<i>t</i>	Successful operation
<i>nil</i>	Unsuccessful operation

Examples

The following OCEAN script sets the value of the `maxOfOut` output to 110 and `minOfOut` to 0.

```
$ cat myMeas.ocn  
aVar = 55  
axlOutputResult( aVar*2 "maxOfOut" )  
axlOutputResult( 0 "minOfOut" )
```

For details on how to load an OCEAN script file for a measurement, refer to [Loading an OCEAN or a MATLAB Measurement](#).

Virtuoso ADE SKILL Reference

Output Functions

Related Topics

[axlAddOutputs](#)

axlOutputsExportToFile

```
axlOutputsExportToFile(  
    t_sessionName  
    t_fileName  
    [ ?omitTestCol g_omitTestCol ]  
)  
=> t / nil
```

Description

Exports outputs from the currently active setup to the specified CSV file.

Arguments

<i>t_sessionName</i>	Name of the currently active session.
<i>t_fileName</i>	Name of the CSV file to which outputs are to be exported.
<i>?omitTestCol g_omitTestCol</i>	Specifies whether to include the <i>Test</i> column in the exported output details. If the <i>Test</i> column is not included, the outputs can be reused for a different test. In the latter case, you need to explicitly specify the test name while importing outputs.

Value Returned

<i>t</i>	Successful operation
<i>nil</i>	Unsuccessful operation

axlOutputsImportFromFile

```
axlOutputsImportFromFile(  
    t_sessionName  
    t_fileName  
    [ ?operation t_operationType ]  
    [ ?test t_test ]  
)  
=> t / nil
```

Description

Imports outputs from the specified CSV file. Outputs can be exported to a CSV file by using `axlOutputsExportToFile`.

Arguments

<i>t_sessionName</i>	Name of the currently active session.
<i>t_fileName</i>	Name of the CSV file from which outputs are to be imported.
<i>?operation t_operationType</i>	Specifies how to use the imported outputs. Valid values are: ■ overwrite: Overwrites the existing set of outputs that are already defined with the same name as that of an output being imported. ■ retain: Retains the existing set of outputs when the name of an existing output matches with that of an output being imported. ■ merge: Keeps the existing set of outputs and merges them with the outputs imported from the specified CSV file. Default value: <code>overwrite</code>

Virtuoso ADE SKILL Reference

Output Functions

`?test t_test`

Name of the test for which the imported outputs are to be used.

Specify this argument only when the test name is not saved in the CSV file being imported. If the CSV file contains the *Test* column, each output is imported for the specified test.

If you wish to use the imported outputs for all the tests, set the value for this argument to `Import to All Tests`.

Default value: " "

Value Returned

`t`

The outputs are imported from the specified CSV file.

`nil`

Indicates an error.

Examples

The following example imports all the outputs from `outputs_Two_Stage_Opamp_OpAmp_AC_top_maestro16.csv` and merges them with the existing set of outputs.

```
axlOutputsImportFromFile("fnxSession0"
"outputs_Two_Stage_Opamp_OpAmp_AC_top_maestro16.csv")
=> t
```

The following example imports all the outputs from `outputs_Two_Stage_Opamp_OpAmp_AC_top_maestro16.csv` for the `test1_1` test and overwrites the existing set of outputs.

```
axlOutputsImportFromFile("fnxSession0"
"outputs_Two_Stage_Opamp_OpAmp_AC_top_maestro16.csv" ?operation "overwrite" ?test
"test1_1")
=> t
```

The following example imports the outputs from `outputs_Two_Stage_Opamp_OpAmp_AC_top_maestro16.csv` to all the tests and overwrites the existing set of outputs.

```
axlOutputsImportFromFile("fnxSession0"
"outputs_Two_Stage_Opamp_OpAmp_AC_top_maestro16.csv" ?operation "overwrite" ?test
"Import to All Tests")
=> t
```


axlPutOutputNotation

```
axlPutOutputNotation(  
    x_sdb  
    t_testName  
    t_outputID  
    t_value  
)  
=> x_sdb / nil
```

Description

Sets the notation style for the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.
<i>t_value</i>	Notation style to be set for the specified output Valid values are: ■ default: Displays results using the default notation style specified in the Default Formatting Options form. ■ eng: Displays results in the engineering notation. ■ sci: Displays results in the scientific notation. ■ suffix: Displays results in the suffix notation.

Virtuoso ADE SKILL Reference

Output Functions

Value Returned

<code>x_sdb</code>	Returns the handle to the output database when the notation style is successfully set.
<code>nil</code>	Notation style is not set for the specified output.

Examples

The following example sets the notation of the output `Current` of test AC to `eng` (engineering).

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlPutOutputNotation(sdb "AC" "Current" "eng")  
=> 2529
```

axlPutOutputSignificantDigits

```
axlPutOutputSignificantDigits(  
    x_sdb  
    t_testName  
    t_outputID  
    x_value  
)  
=> x_sdb / nil
```

Description

Sets the number of significant digits corresponding to the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.
<i>x_value</i>	The digits to be set for the specified output. Valid value: Any integer between 2–15.

Value Returned

<i>x_sdb</i>	Returns the handle to the output database when the number of significant digits is successfully set
<i>nil</i>	The significant digits are not set for the specified output.

Examples

The following example sets the number significant digits for the output `Current` of test `AC` to 5.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlPutOutputNotation(sdb "AC" "Current" 5)  
=> 2529
```

axlPutOutputSuffix

```
axlPutOutputSuffix(  
    x_sdb  
    t_testName  
    t_outputID  
    x_value  
)  
=> x_sdb / nil
```

Description

Sets the suffix corresponding to the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.
<i>x_value</i>	Suffix to be set for the specified output. Valid value: Any integer between 2–15.

Value Returned

<i>x_sdb</i>	Returns the handle to the output database when the suffix is successfully set
<i>nil</i>	The suffix is not set for the specified output.

Examples

The following example sets the suffix for the output `Current` of test `AC` to `m`.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlPutOutputNotation(sdb "AC" "Current" "m")  
=> 2529
```

axlPutOutputUnits

```
axlPutOutputUnits(  
    x_sdb  
    t_testName  
    t_outputID  
    t_value  
)  
=> x_sdb / nil
```

Description

Sets the unit to the specified output of the test.

Arguments

<i>x_sdb</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test in the setup database.
<i>t_outputID</i>	Name of the output.
<i>t_value</i>	Unit to be set for the specified output.

Value Returned

<i>x_sdb</i>	Returns the handle to the output database when the unit of the output is successfully set
<i>nil</i>	The suffix is not set for the specified output.

Examples

The following example sets the unit for the output UGF of test AC to Hz.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlPutOutputNotation(sdb "AC" "UGF" "Hz")  
=> 2533
```

axlRenameOutputsColumn

```
axlRenameOutputsColumn(  
    x_mainSDB  
    t_columnName  
    t_newColumnName  
)  
=> t / nil
```

Description

Changes the name of a user-defined column in the outputs table.

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.
<i>t_outputName</i>	Name of the user-defined column to be renamed.
<i>t_newColumnName</i>	New name to be assigned to the user-defined column.

Value Returned

<i>t</i>	The user-defined column is renamed.
<i>nil</i>	The user-defined column is not renamed.

Examples

The following example renames the user-defined column `myColumn` to `myRemarks`.

```
session = axlGetWindowSession()  
=> "session0"  
sdb= axlGetMainSetupDB(session)  
=> 1001  
axlRenameOutputsColumn(sdb "myRemarks" "myColumn")  
=> t
```

axlSetOutputUserDefinedData

```
axlSetOutputUserDefinedData(  
    x_mainSDB  
    t_testName  
    t_outputName  
    t_columnName  
    t_columnValue  
)  
=> t / nil
```

Description

Sets the value in the user-defined column for the specified test name and output.

Arguments

<i>x_mainSDB</i>	Handle to the main setup database.
<i>t_testName</i>	Name of the test.
<i>t_outputName</i>	Name of the output in the test setup.
<i>t_columnName</i>	Name of the user-defined column.
<i>t_columnValue</i>	Value to be set in the user-defined column for the specified output.

Value Returned

<i>t</i>	The value is set for the specified output in the user-defined column.
<i>nil</i>	Unsuccessful operation

Examples

Sets the value `Current < 1.5m` for the output `Current` of test `AC` in the user-defined column `myRemarks`.

```
session=axlGetWindowSession()  
=> "fnxSession0"  
sdb=axlGetMainSetupDB(session)  
=> 1001  
axlSetOutputUserDefinedData(sdb "AC" "Current" "myRemarks" "Current < 1.5m")
```

Virtuoso ADE SKILL Reference

Output Functions

=> t

calcVal

```
calcVal(  
    t_outputName  
    [ t_testName ]  
    [ ?cornerName t_cornerName ]  
    [ ?historyName t_historyName ]  
    [ ?lib t_libName ]  
    [ ?cell t_cellName ]  
    [ ?view t_viewName ]  
    [ ?run t_runName ]  
    [ ?result t_resultName ]  
    [ ?getFirstSweepPoint t_getFirstSweepPoint ]  
    [ ?ignoreParams g_ignoreParams ]  
    [ ?matchParams g_matchParams ]  
    [ ?defaultVal t_defaultVal ]  
    [ ?relxName t_relxName ]  
    [ ?ageValue t_ageValue ]  
    [ ?ageUnit t_ageUnit ]  
    [ ?she g_she ]  
)  
=> g_output / nil
```

Description

Returns the value of an output from the same or another test. You can use the value returned by this function in another output expression.

Arguments

<i>t_outputName</i>	Name of the output or expression. The output can return a scalar value or waveform. To retrieve the value of an output of type signal, the simulation waveform results must be saved.

Limitation:

calcVal supports waveform outputs in expressions, but it does not support waveforms outputs used in subexpressions.

For example, the following calcVal expression returns a good result :

```
a=calcVal("b" "test1")  
;          where, b=(VT("/out1") + 1)
```

The following calcVal expression returns an error:

```
c=calcVal("d" "test1")  
;          where, d=(e + 1) and e=VT("/out1")
```

Virtuoso ADE SKILL Reference

Output Functions

t_testName

Name of test name to which the given output belongs. When not specified, the name of the current test is used.

You do not need to provide this argument when using the `calcVal` expression for global variables.

For `calcVal` expressions that use the result of an output from another test, it is recommended to use outputs that return scalar values for better performance.

?cornerName t_cornerName

Name of the corner. When specified, the value of the given output for this corner is used. Name of a corner group cannot be used in this argument.

Note: Do not use a SKILL expression to specify a corner name unless you are also using the `?historyName` argument in the `calcVal` expression.

?historyName t_historyName

Virtuoso ADE SKILL Reference

Output Functions

Name of the history from which the results are to be retrieved. When specified, the scalar or waveform result of the output from the given history is used. Otherwise, the value of the output from the current simulation run is used.

You can use the following wildcards to find a history saved for the given cellview:

- `<highest>`: To match a history name that has the highest numerical value amongst all histories for which the name begins with the given prefix. For example, `Interactive.<highest>`.
- `<lowest>`: To match a history name that has the lowest numerical value amongst all histories for which the name begins with the given prefix. For example, `Interactive.<lowest>`.
- `<latest>`: To match the history name so that it is the latest amongst all histories for which the name matches the given pattern. The `<latest>` wildcard matches any substring. For example, `Inter<latest>` or `<latest>ive.4`.
- `<oldest>`: To match the history name so that it is the oldest amongst all histories for which the name matches the given pattern. The `<oldest>` wildcard matches any substring. For example, `Inter<oldest>` or `<oldest>ive.4`.
- `<any>`: To match a history name that begins with the given prefix followed by any substring. For example, `Hist<any>3` would match a history name that begins with `Hist` and ends with `3`.
- `<ro>`: To match a history name irrespective of the `.RO` suffix. The editable or read-only status is ignored. For example, `Interactive.4<ro>` would match with `Interactive.4` or `Interactive.4.RO`.

You can combine two or more wildcards, for example, `In<any>.<highest><ro>`. Ensure that the combination does not produce conflicting results. For example, avoid using `<oldest>` and `<latest>` together.

```
?lib t_libName
```

Virtuoso ADE SKILL Reference

Output Functions

The library associated with the maestro cellview from where the results are to be retrieved.

This argument can be used only when the `?historyName` argument is specified along with it. This ensures `calcVal` knows which history to retrieve the results from.

`?cell t_cellName`

The cell associated with the maestro cellview from where the results are to be retrieved.

This argument can be used only when the `?historyName` argument is specified along with it. This ensures that `calcVal` knows which history to retrieve the results from.

`?view t_viewName`

The view associated the maestro cellview from where the results are to be retrieved.

This argument can be used only when the `?historyName` argument is specified along with it. This ensures that `calcVal` knows which history to retrieve the results from.

`?run t_runName`

Name of the run plan from which the value of the given output is to be returned. This argument is useful only when the setup has a run plan.

Use this argument to get the value of an output from the results of one run (in the run plan) and use it in a variable, expression, or run condition for another run in the same run plan or history. It cannot be used to refer to the results of any other history.

When both `?run` and `?historyName` are specified, `?historyName` is ignored by `calcVal`.

Note: The `?run` argument is used only for a cellview created using ADE Assembler.

Also see: [Using Results of One Run in Another](#)

`?result t_resultName`

Virtuoso ADE SKILL Reference

Output Functions

Name of an analysis from which the `signal` type output is to be retrieved. When specified, returns a waveform for the corresponding output of that analysis. Use this argument only for the outputs of type `signal`.

The specified analysis must exist in the test along with the simulation waveform results in order for `calcVal` to return the expected result.

`?getFirstSweepPoint` *t_getFirstSweepPoint*

Specifies that when sweep values are used, the result of only the first swept point is to be used for calculation.

If `?cornerName` is specified with this argument, the function retrieves the output value of the first swept point for the given corner. Otherwise, `calcVal` retrieves the results for all the corners set up for the test.

Default value: `nil`

`?matchParams` *t_matchParams*

Virtuoso ADE SKILL Reference

Output Functions

Specifies the requirements to look for points that have matched variables and parameters in the current run and the source run. The source run can be of any other test in the current history, or the same or another test in another history specified by `?historyName`.

Possible values:

- `"none"`: (Default value) Does not look for the matched variables and parameters. Instead, it uses the result value for the current data point if all data points in the source and target tests are same. Otherwise, `calcVal` returns `nil`.
- `"all"`: Matches each variable or parameter value in the current test with the variables and parameters of the source test. If an identical point is found, returns the required output value for that point. If no point with the same number and values of variables and parameters is found, returns `nil`.
- A list containing a list of name-value pairs of the parameters to be matched: Compares the given variables and parameters. The list is given in the `list(list(<param1> <val1>) list(<param2> <val2>) ...)` format, where `<valN>` can be a constant or a value returned by a user-defined function. If the same variables and parameters are found with the given values in both source and target points, returns the value of the given output.

Important Points to Note

- This argument considers the corners and sweep parameters in the same way.
- If the `?getFirstSweepPoint` argument is set to `t`, only the first parameter is compared.
- Avoid setting `?matchParams` as `"none"` when the tests contain local sweeps.
- Escape the string values. Example, `(list "trim" "\"b\"")`.
- If no matching point is found, the function returns `nil`. If multiple matching points are found and each point belongs to a different corner, `calcVal` returns the value from the corner that has the same name as that of the current corner for which the function is being evaluated—this is called corner affinity. In all other cases, `calcVal` returns `nil`.

Virtuoso ADE SKILL Reference

Output Functions

`?ignoreParams` *g_ignoreParams*

Ignores the specified parameters. For example, a local parameter value that is different in the source and the target test or a parameter whose value is a `calcVal` expression.

If a list of name-value pairs are specified for `?matchParams`, the given parameters must match with the given value. Values of all other parameters should match with the values of corresponding parameters in the current point. If you want to consider only `?matchParams` and ignore the rest of the variables and parameters that are not specified by `?matchParams`, set `?ignoreParams` to `rest`.

If the model files in the setup of the source and target tests are different and you want to ignore those while comparing setups, add `modelFiles` to the list specified for `?ignoreParams`.

If `?ignoreParams` is specified without `?matchParams`, the tool assumes `?matchParams` to be set to `all`.

`?defaultVal` *t_defaultVal*

The default value to be returned if `calcVal` fails for any reason.

This is helpful when `calcVal` is used in an expression for a dependent output measurement, or in the input value for a test variable.

`?relxName` *t_relxName*

Name of the reliability setup to be used.

`?ageValue` *t_ageValue*

Specifies one of the following values:

- `Fresh`: For the fresh stage
- `Stress`: For stress stage
- `age_number`: For aged stage (Aging simulation)

"`age_number`" specifies the age points at which the device degradation is calculated.

`?ageUnit` *t_ageUnit*

Virtuoso ADE SKILL Reference

Output Functions

Specifies the unit for the age points.

Valid Values: `yr`, `year`, `years`, `d`, `day`, `days`, `h`, `hour`, `hours`, `min`, `minute`, `minutes`, `sec`, `second`, `seconds`.

Default Value: `year`

`?she g_she`

Boolean value to specify whether the self-heating effect is enabled.

Valid Values:

- `t`: Self-heating effect is enabled
- `nil`: Self-heating effect is disabled

Default Value: `nil`

Value Returned

`g_output`

Returns the value of the output.

`nil`

Returns `nil` if the function fails and the default value is not specified.

The function fails when the values specified for its arguments are not correct. For example, when the specified test, output expression, or corner is not found in the given history or when there is no history that matches the criteria set by the `?historyName` argument.

Examples

In the following example, `avg_vt` represents the expression `average(VT("/out"))` and `myTest` is the test name:

```
calcVal("avg_vt" "myTest")
```

You can compute the value of an expression with reference to another expression from a different test, as shown below.

```
calcVal("output1" "test1")/calcVal("output2" "test3")
```

You can also use the output value for a corner, as shown below.

```
calcVal("Gain" "AC" ?cornerName "C2")
```

You can also use the output value from a history that has the highest numerical value suffixed with the history name `Interactive`, as shown below.

Virtuoso ADE SKILL Reference

Output Functions

```
calcVal("Gain" "AC" ?historyName "Interactive.<highest>")
```

You can use the waveform from the result, as shown below.

```
calcVal("/OUT" "AC" ?result "ac")
=> srrWave:0x40421090
```

If you want to retrieve the value from only the first sweep point, you can use the `?getFirstSweepPoint` argument. The following example will always return the first sweep point value of `myCalib` in the test `calib_tb`. This is useful if you run a sweep on the test where the `calcVal` expression is used but the source of the value `myCalib` is not swept.

```
calcVal("myCalib" "calib_tb" ?getFirstSweepPoint t)
```

You can specify a default value that can be used when the result for the specified output is not found.

```
calcVal("Gain" "AC" ?cornerName "C2" ?defaultVal "1000")
```

The default value can also be used to show an error message when the result for the corner is not available.

```
calcVal("Gain" "AC" ?cornerName "C2" ?defaultVal "ERROR: Gain for C2 is not available")
```

You can retrieve the results from a nominal run. The following example would give an output, which is the ratio of the measured frequency of the specified corner to the frequency in the nominal run:

```
calcVal("Freq")/calcVal("Freq" ?cornerName "Nominal")
```

This example shows how `?matchParams` works when multiple matching points are found and they belong to different corners.

The setup contains two tests, AC and TRAN, with multiple points and corners. The corners setup appears as shown below.

Corners	☒ Nominal	☐ 2Sigma_UGF	☒ ...haseMargin	☒ C0
Design Variables		6 stat vars	4 stat vars	
__scale		1.0	1.0	
Click to add				
Parameters				
Click to add				
Model Files				
Click to add				
Model Group(s)		<modelgroup>	<modelgroup>	<modelgroup>
Click to add				
Tests				
☒ AC	☒	☒	☒	☐
☒ TRAN	☒	☒	☒	☒

Virtuoso ADE SKILL Reference

Output Functions

The test TRAN has an output GainVal with the calcVal expression shown below.

```
calcVal("Gain" "AC" ?matchParams "all")
```

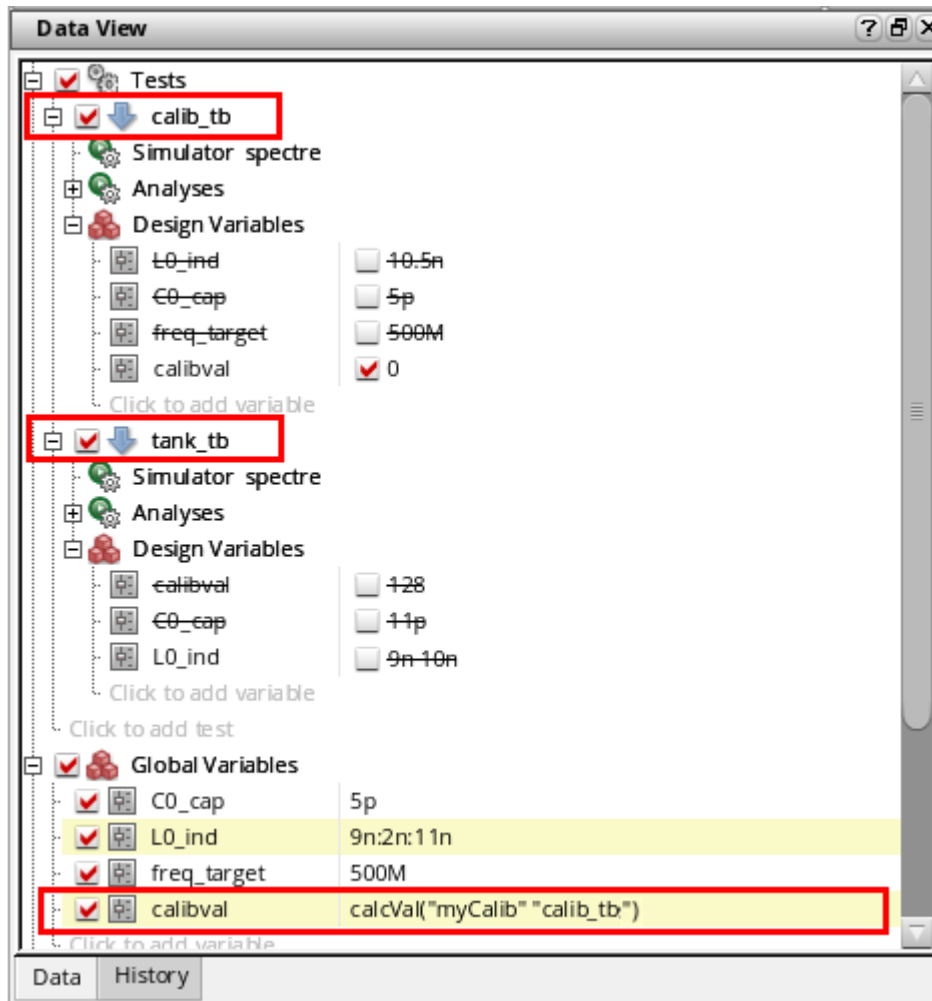
The corners Nominal and C0 are identical, so calcVal evaluation for each point finds two corners in which all variables and parameters match. However, the calcVal expression is successfully evaluated only for Nominal.

For C0, it returns eval err because when ?matchParams is used, even though calcVal finds the matching point in multiple corners, it can use the value from a corner that has the same name as that of the corner for which the expression is being evaluated, which is C0 in this case. C0 is disabled for AC, so no value is returned for Gain. Therefore, the output returns an error. To return a value result for GainVal, you must enable C0 for AC.

Point	Test	Output	Nominal	Spec	Weight	Pass/Fail	Min	Max	Sigma_PhaseMargi	C0
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
3	AC	/inm								disabled
3	AC	/inp								disabled
3	AC	/cmin								disabled
3	AC	VAR("nomGain")	0				0	0	0	disabled
Parameters: vdd=1.3										
4	TRAN	/OUT								
4	TRAN	Swing	1.008	> 0.98		pass	1.006	1.008	1.006	1.008
4	TRAN	SettlingTime	7.763 ns	< 8n		pass	7.763 ns	7.795 ns	7.795 ns	7.763 ns
4	TRAN	RelativeSwingP...	77.51 %	> 75		pass	77.42 %	77.51 %	77.42 %	77.51 %
4	TRAN	PhaseMargin	20.71 degree	> 19.4		pass	19.58 deg...	20.71 deg...	19.58 degree	20.71 degree
4	TRAN	GainVal	46.74 dB				46.69 dB	46.74 dB	46.69 dB	eval err
Parameters: vdd=1.4										

More Examples Using ?matchParams

The following examples show how you can use the `?matchParams` argument of `calcVal`. The base setup used in these examples is shown below.



This setup contains two tests, `calib_tb` and `tank_tb`.

Virtuoso ADE SKILL Reference

Output Functions

If not overriding, these tests use four global variables: `C0_cap`, `L0_ind`, `freq_target`, and `calibval`. `calibval` contains an expression that uses the value of the `myCalib` output defined for the test `calib_tb`.

Test	Name	Type	Details	EvalType	Plot
calib_tb	calib_dummy_output	expr	1	point	✓
calib_tb	myCalib	expr	$((\text{xmin}(\text{mag}(\text{IF}("/\text{IO}/\text{tankp}")) 1) / 1000000) - 500) * 4)$	point	✓
calib_tb		signal	/IO/tankp	point	□
calib_tb		expr	$((\text{xmin}(\text{mag}(\text{IF}("/\text{IO}/\text{tankp}")) 1) / 1000000) - 200) * 4)$	point	✓
tank_tb		signal	/IO/tankp	point	✓
tank_tb	freq_res	expr	$\text{xmin}(\text{mag}(\text{IF}("/\text{IO}/\text{tankp}")) 1)$	point	✓
tank_tb		expr	<code>VAR("/calibval")</code>	point	✓

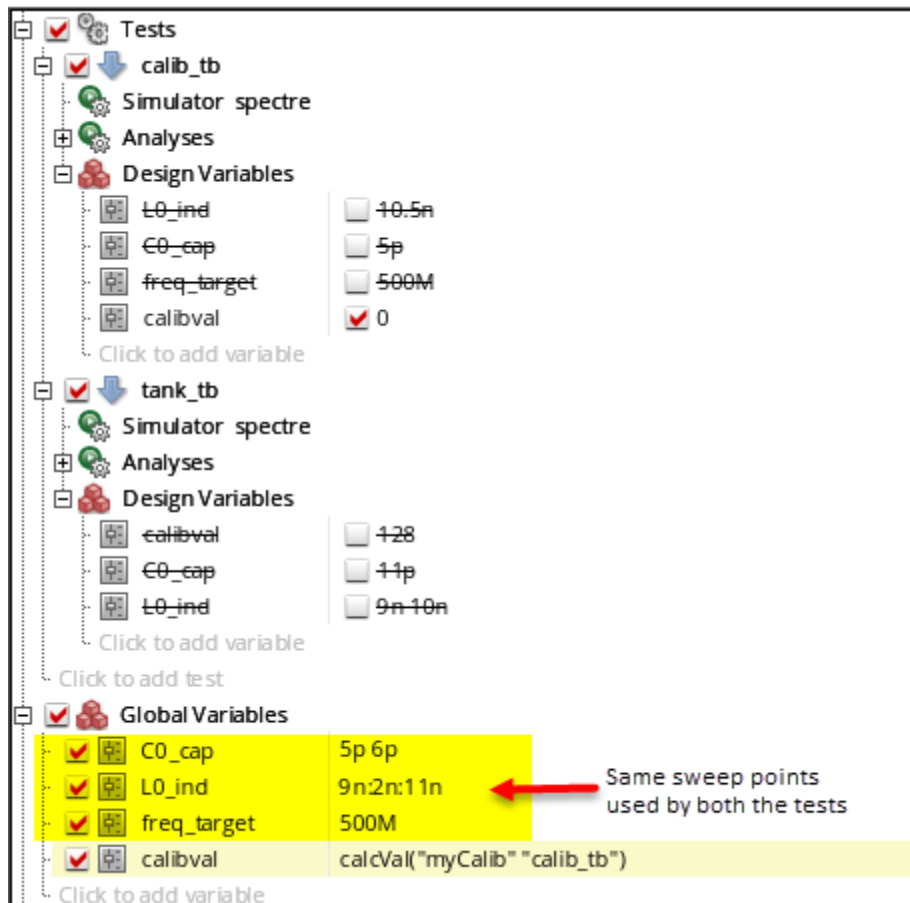
The `VAR("/calibval")` output of test `tank_tb` returns the value of the `calibval` global variable, which saves the result of the `calcVal` function.

In the following example cases, you can observe the difference in the value of `VAR("/calibval")` for different values of `?matchParam`, and varying variables and parameters:

Virtuoso ADE SKILL Reference

Output Functions

Case 1: ?matchParams is set to the default value ("none"), the number of data points in the calibrated and target tests is same, and the swept variables and corners are identical



For each data point of tank_tb, there will be a valid value of the calibval global variable. Thus, calcVal returns a valid, non-nil value.

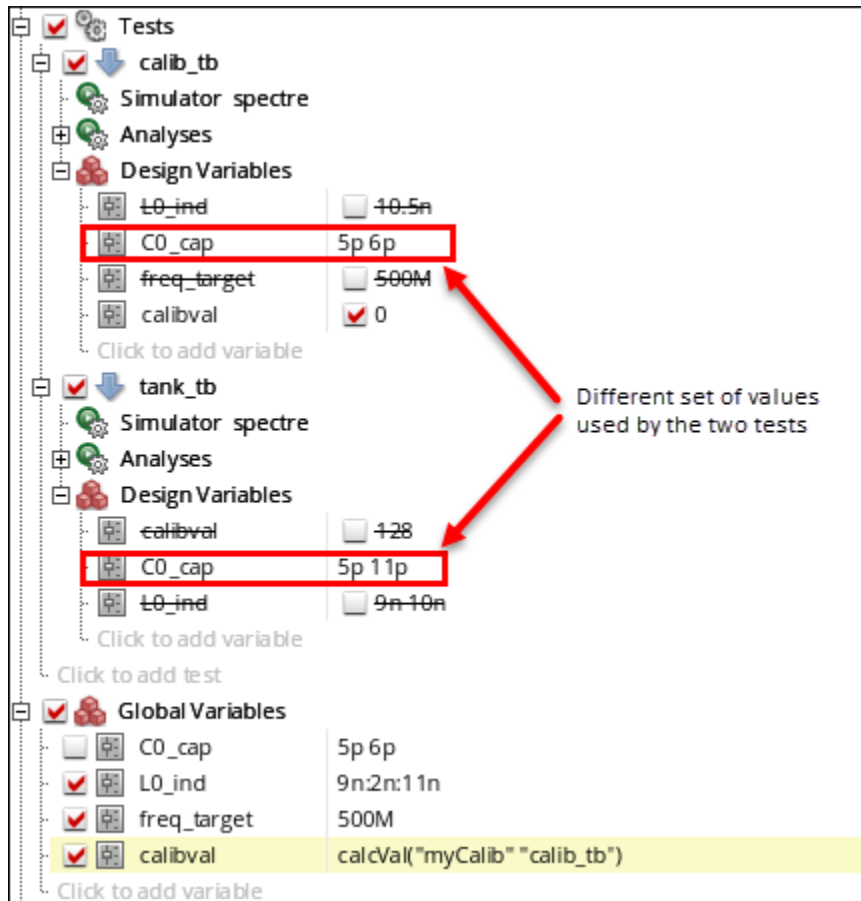
calib_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
myCalib	198	4	99	-85
tank_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
calibval	198	4	99	-85

Note: In the case described above, the return value would be same if ?matchParams is set to "all" because the count and values of all variables and parameters match.

Virtuoso ADE SKILL Reference

Output Functions

Case II: ?matchParams is set to the default value ("none"), the number of data points in the calibrated and target tests is same, but the values are different



In this example, ?matchParams is set to "none", but calcVal does not look for any matching variable because the setup contains local sweeps. As a result, for each point of tank_tb, calcVal returns nil.

calib_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
myCalib	198	4	99	-85
tank_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=11p L0_ind=9n	C0_cap=11p L0_ind=11n
calibval	nil	nil	nil	nil

Virtuoso ADE SKILL Reference

Output Functions

Important

In the case shown above, `calcVal` does not return valid results. Therefore, it is recommended to use `?matchParams` set to "none" only when the tests do not have local sweeps, the count of the enabled sweep points and corners in the source and target test is same and they are identical.

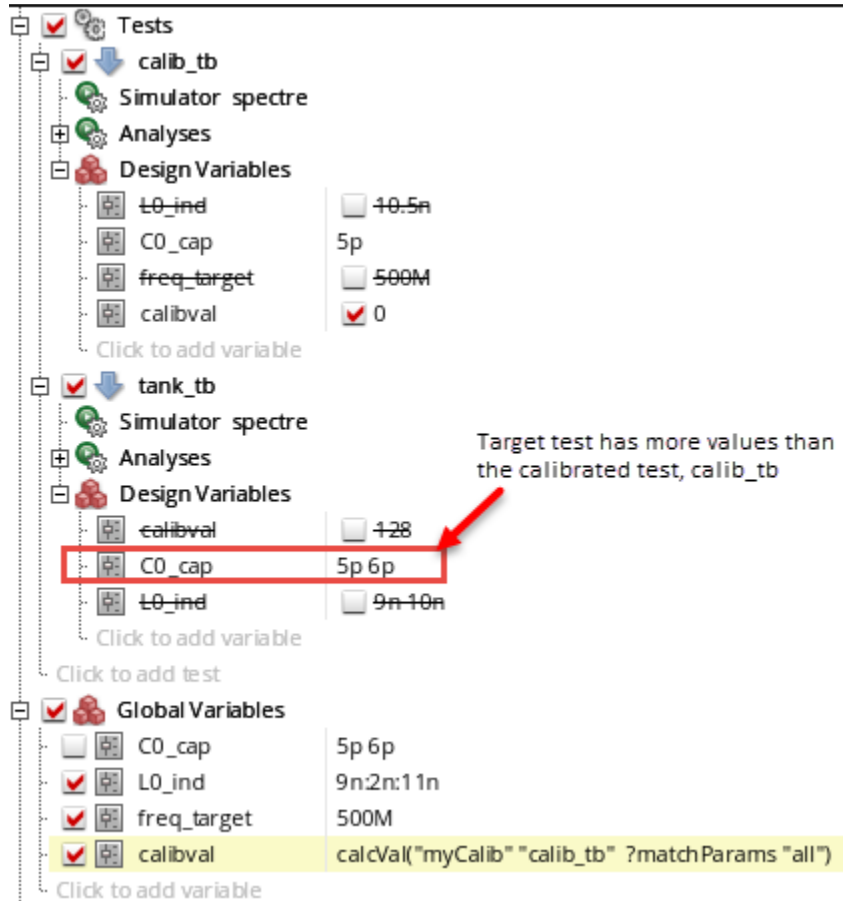
In case II described above, if `?matchParams` is set to "all" instead of "none", `calcVal` looks for matching variables and parameters in other points, and the results are different, as shown below.

calib_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
myCalib	198	4	99	-85
tank_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=11p L0_ind=9n	C0_cap=11p L0_ind=11n
calibval	198	4	nil	nil

Virtuoso ADE SKILL Reference

Output Functions

Case III: ?matchParams is set to "all" and the target test has more data points than the calibrated test



calcVal does not find matching data points for some parameters. For those points, it returns nil, as shown below.

calib_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n		
myCalib	198	4		
tank_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
calibval	198	4	nil	nil

Virtuoso ADE SKILL Reference

Output Functions

Case IV: ?matchParams is set to list(list("L0_ind" "11n") list("C0_cap" "8p"))

The screenshot shows the Virtuoso ADE SKILL interface. The 'Global Variables' section is expanded, showing the configuration for 'calibval'. The expression for 'calibval' is highlighted in a red box: `calcVal("myCalib" "calib_tb" ?matchParams list(list("C0_cap" "8p") list("L0_ind" "11n")))`. A red arrow points to this expression with the text: `?matchParams uses a list to define the variables to match`.

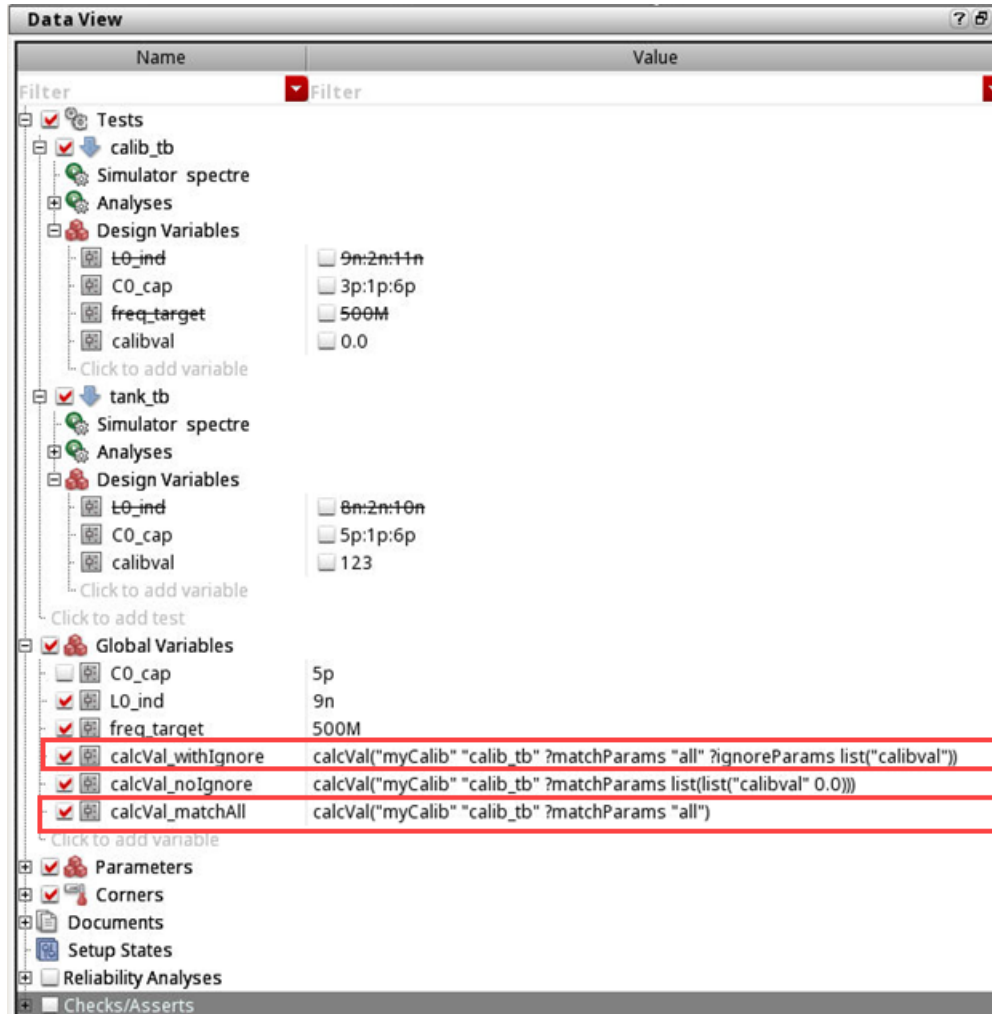
In this case, `calcVal` looks for data points in the calibrated test that have parameters `L0_ind=11n` and `C0_cap=8p`. If found, the target test uses the value of the `myCalib` expression to get the value for `calibval` for all the points in the target test, as shown below.

calib_tb	C0_cap=5p L0_ind=9n	C0_cap=5p L0_ind=11n	C0_cap=8p L0_ind=9n	C0_cap=8p L0_ind=11n
myCalib	198	4	99	-85
tank_tb	C0_cap=8p L0_ind=9n	C0_cap=8p L0_ind=11n	C0_cap=6p L0_ind=9n	C0_cap=6p L0_ind=11n
calibval	-85	-85	-85	-85

Virtuoso ADE SKILL Reference

Output Functions

Case V: ?matchParams is used with ?ignoreParams where ?matchParams is set to "all" and ?ignoreParams is set to list ("calibVal")



On using ?matchParams with ?ignoreParams, calcVal looks for matching data points but ignores the local sweep variable calibval that has different values in both the tests. The target test then uses the value of the myCalib expression to get the value for calcVal_withIgnore for all the points in the target test.

Virtuoso ADE SKILL Reference

Output Functions

However, on setting `?matchParams` to "all" without using `?ignoreParams`, you get an evaluation error, as `calcVal` tries to match all data points and encounters `calibval` that has a different value in both the tests.

Point	Test	Output	Nominal	Spec	Weight	Pass/Fail
Parameters: C0_cap=3p, calcVal_withIgnore=calcVal("myCalib" "calib_tb" ?matchParams "all" ?ignoreParams list("calibval")), calcVal_noIg...						
1	calib_tb	calib_dummy_output	1			
1	calib_tb	myCalib	523.8			
1	calib_tb	/I0/tankp				
1	calib_tb	((xmin(mag(IF("/I0/tankp")) 1) / 1...	1.724K			
Parameters: C0_cap=4p, calcVal_withIgnore=calcVal("myCalib" "calib_tb" ?matchParams "all" ?ignoreParams list("calibval")), calcVal_noIg...						
2	calib_tb	calib_dummy_output	1			
2	calib_tb	myCalib	410.2			
2	calib_tb	/I0/tankp				
2	calib_tb	((xmin(mag(IF("/I0/tankp")) 1) / 1...	1.61K			
Parameters: C0_cap=5p, calcVal_withIgnore=calcVal("myCalib" "calib_tb" ?matchParams "all" ?ignoreParams list("calibval")), calcVal_noIg...						
3	calib_tb	calib_dummy_output	1			
3	calib_tb	myCalib	198.2			
3	calib_tb	/I0/tankp				
3	calib_tb	((xmin(mag(IF("/I0/tankp")) 1) / 1...	1.398K			
Parameters: C0_cap=6p, calcVal_withIgnore=calcVal("myCalib" "calib_tb" ?matchParams "all" ?ignoreParams list("calibval")), calcVal_noIg...						
4	calib_tb	calib_dummy_output	1			
4	calib_tb	myCalib	99.23			
4	calib_tb	/I0/tankp				
4	calib_tb	((xmin(mag(IF("/I0/tankp")) 1) / 1...	1.299K			
Parameters: C0_cap=5p, calcVal_withIgnore=calcVal("myCalib" "calib_tb" ?matchParams "all" ?ignoreParams list("calibval")), calcVal_noIg...						
5	tank_tb	/I0/tankp				
5	tank_tb	freq_res	531.3M	range 495e6 5...		near
5	tank_tb	VAR("calibval")	123			
5	tank_tb	myCalib_withIgnoreParams	198.2			
5	tank_tb	myCalib_noIgnoreParams	198.2			
5	tank_tb	myCalib_matchAll	eval err			
Parameters: C0_cap=6p, calcVal_withIgnore=calcVal("myCalib" "calib_tb" ?matchParams "all" ?ignoreParams list("calibval")), calcVal_noIg...						
6	tank_tb	/I0/tankp				
6	tank_tb	freq_res	506.6M	range 495e6 5...		near
6	tank_tb	VAR("calibval")	123			
6	tank_tb	myCalib_withIgnoreParams	99.23			
6	tank_tb	myCalib_noIgnoreParams	99.23			
6	tank_tb	myCalib_matchAll	eval err			

Recommendations for ?output in calcVal

Consider the following recommendations while using `calcVal` expressions in outputs:

- To optimize `calcVal` performance, it is recommended to point `calcVal` to scalar output expressions. Though `calcVal` can also be used on signals and expression resulting into waveforms, it is costly to evaluate those.

For example, to get `ymax` of signal `\OUT` from `test1` in `test2`, you can use two options:

- Use `calcVal` to get the signal `\OUT` into `test2` and then calculate `ymax` in `test2`

Virtuoso ADE SKILL Reference

Output Functions

- ❑ Calculate `ymax` of `\OUT` in `test1` itself and fetch that scalar value by `calcVal`

The second option is more efficient. However, there can certainly be cases when it cannot be avoided to use `calcVal` on waveform expression and signals.

- You can use `calcVal` inside complex expressions, however, it is more efficient to use a separate output expression for each `calcVal`. This helps in debugging because you can see which part of the expression might cause an issue.

For example, the expression `myValue = max(calcVal(...) calcVal(...))`, can be represented as shown below.

```
value1 = calcVal(...)
value2=calcVal(...)
myValue=max(value1 value2)
```

Recommendations for Using `?matchParams` in `calcVal`

Depending on the variable and corner settings in the target and source (calibrated) test, you can specify an appropriate value for the `?matchParams` argument of `calcVal` to obtain correct results. You may find the following recommendations useful:

- Use the default value, `"none"`, when the following conditions are met:
 - ❑ The source and target tests do not use local sweep variables
 - ❑ The count of sweep values of variables and corners in the source and target tests is same
 - ❑ The values of all enabled corners and swept variables are identical

In this case, `calcVal` considers only the count, and setup of variables and corners. It does not look for matching parameters.

- Set `?matchParams` to `"all"` when the tests have local sweeps or corners with varying values. In this case, `calcVal` finds points with matching sweeps and corners in the current or given history or run of the source test.
- Set `?matchParams` to `<a-list-of-specific-design-points>` or a user-defined function that returns a list containing name-value pairs of variables when the tests have local sweeps or corners with varying values, and you want `calcVal` to explicitly look for the given matching design points in the source and target tests. See the example in case IV above. You can also use the `?ignoreParams` argument in addition to the `?matchParams` argument to ignore specific parameters in case you have local parameters with varying values or a parameter that has a `calcVal` expression as a value.

Virtuoso ADE SKILL Reference

Output Functions

- To use the results of a non-identical corner from the source test, use the default value of `?matchParams` and specify the corner name of the source test by using the `?cornerName` argument.
- To use the results from another history or run by using the `?history` or `?run` arguments of `calcVal`, ensure that the design points are identical. These arguments can be used with any value of `?matchParams`.
- Do not specify `?matchParams` when `?getFirstSweepPoint` is set to `t`, which indicates that the result of the first sweep point in the source test is to be used for all design points in the target test. `?getFirstSweepPoint` works with the default value of `?matchParams` and ignores any other value set for it.

Virtuoso ADE SKILL Reference

Output Functions

Troubleshooting calcVal

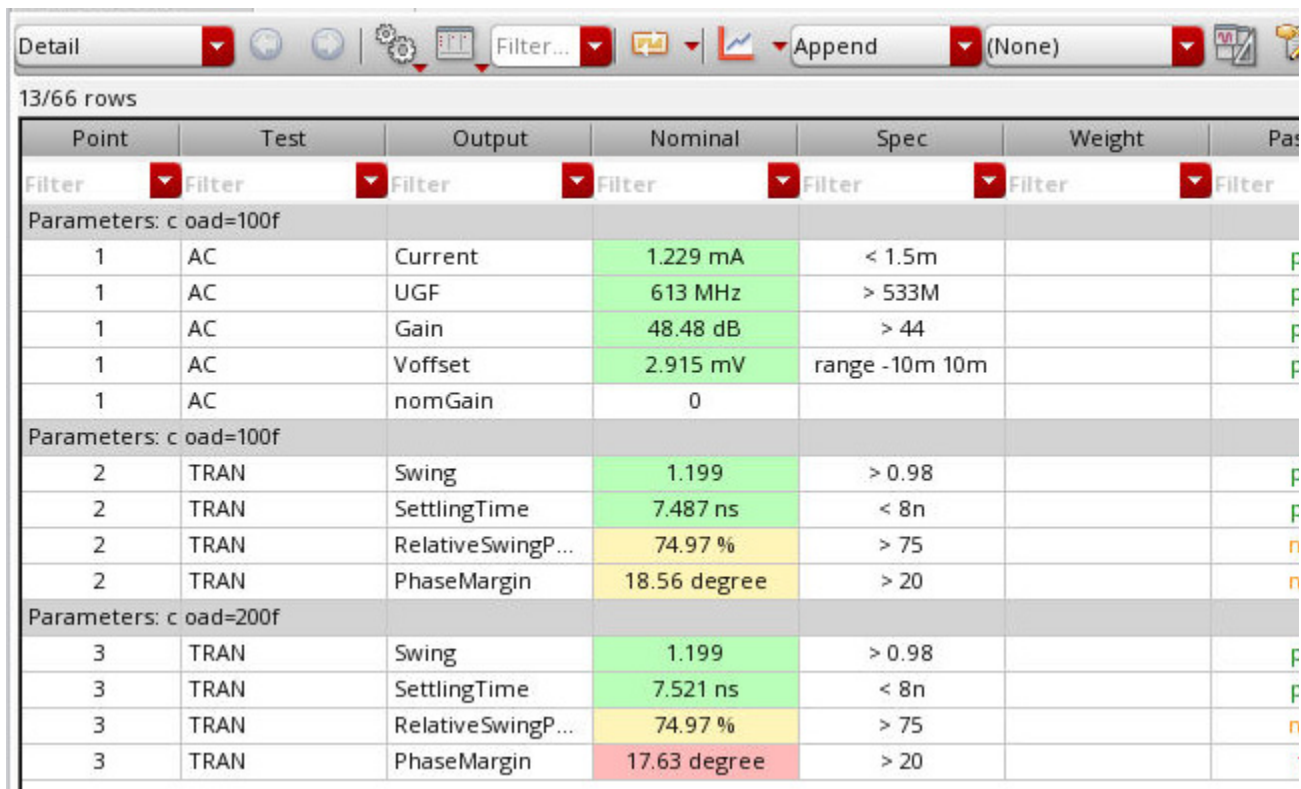
Before running a simulation, ADE Explorer or ADE Assembler check the `calcVal` expressions in outputs or variables for any syntax issues and report errors that you can fix before running the simulation. If you proceed without fixing the syntax issues or if the values assigned to the arguments during runtime are invalid, the expressions return `eval err`. You can set the `debug` environment variable to specify the level of details to be reported in the log file to help you resolve issues.

The examples cases described in this section show how you can resolve issues for `calcVal` expressions.

Case 1: calcVal with Local Sweeps

With local sweep values, the *Results* tab displays independent sweep points for each test. In such cases, on using `calcVal` with just the arguments `t_outputName` and `t_testName`, the function returns an evaluation error or `nil`. This is because when `calcVal` tries to fetch the target value from the specified test, it looks for the same sweep point as that of the test where the function is specified.

For example, let us consider the following setup.



Point	Test	Output	Nominal	Spec	Weight	Pass/Fail
Parameters: c oad=100f						
1	AC	Current	1.229 mA	< 1.5m		P
1	AC	UGF	613 MHz	> 533M		P
1	AC	Gain	48.48 dB	> 44		P
1	AC	Voffset	2.915 mV	range -10m 10m		P
1	AC	nomGain	0			
Parameters: c oad=100f						
2	TRAN	Swing	1.199	> 0.98		P
2	TRAN	SettlingTime	7.487 ns	< 8n		P
2	TRAN	RelativeSwingP...	74.97 %	> 75		N
2	TRAN	PhaseMargin	18.56 degree	> 20		N
Parameters: c oad=200f						
3	TRAN	Swing	1.199	> 0.98		P
3	TRAN	SettlingTime	7.521 ns	< 8n		P
3	TRAN	RelativeSwingP...	74.97 %	> 75		N
3	TRAN	PhaseMargin	17.63 degree	> 20		N

Virtuoso ADE SKILL Reference

Output Functions

Here, on using the function `calcVal("UGF" "AC")` for the second sweep point of the test `TRAN`, `calcVal` will try to find the value of `UGF` from the second sweep point for the test `AC`, which does not exist. This will result in an evaluation error.

You can fix this by using additional arguments like `?getFirstSweepPoint`, `?matchParams`, and `?ignoreParam`.

To avoid such errors, you can also use the *calcVal lint check* button in the *Run Preview* tab to run pre-simulation checks on `calcVal` expressions. You can then view the resulting output messages in the CIW.

Case 2: Using `?matchParams` with multiple matching points in the target test

On using the arguments `?ignoreParams` and `?matchParams`, `calcVal` looks for matching points in the target test. But on encountering multiple matching points, it would not know which value to pick and will return `nil`.

For example, on setting `?matchParams` and `?ignoreParams` to `list(list("a" 1) list("b" 3))` and `list("c")` respectively, with a local sweep `c=5 6 7` in the target test, `calcVal` would find three matching values and would not know which one to use.

Case 3: Syntax errors

The most common issues with `calcVal` expressions are syntax errors, such as missing double quotation marks or a missing parenthesis.

When the `calcVal` expressions have syntax errors, the tool displays warning messages. It is recommended to fix the reported warnings before continuing with simulation. However, if you still continue, the incorrect expressions either result in a `nil` value in the netlist or an evaluation error in the results of output expressions.

To avoid such errors, you can use the *calcVal lint check* button in the *Run Preview* tab to run pre-simulation checks on `calcVal` expressions. You can then view the resulting output messages in the CIW.

Case 4: Using the argument `?getFirstSweepPoint`

In general, the `?getFirstSweepPoint` argument is useful to tell `calcVal` to fetch the value from the first sweep point only, especially while working with local sweeps. But in specific cases, you might get an evaluation error.

Virtuoso ADE SKILL Reference

Output Functions

In case, you have a local sweep in the target test and you run only one point in the source test, `calcVal` would not know which value to pick even though there is just one point.

To avoid such issues, you can specify the option `?getFirstSweepPoint t` in the expression to tell `calcVal` to fetch the value from the first sweep point, irrespective of the number of sweeps in the source test.

Case 5: `calcVal` accuracy

The `calcVal` return value is a copy of the target value this function is pointing to, but a comparison with the original value shows some extremely small differences.

`calcVal` needs to do some processing of the values and expressions. Usually, a 64-bit double floating point value can be precisely processed upto fifteen digits. Differences in the thirteen, fourteen, and, fifteen digits range indicate a difference in the numerical noise level. The simulator can accurately process values in the range of five to six digits. Therefore, comparing values from the numerical noise level should be avoided. Comparison should always be done with a certain range to avoid any false alarm.

Case 6: The usage of `?matchParams` and `?ignoreParams`

The `?matchParams` and `?ignoreParams` arguments let you specify which value to fetch in case of local sweeps. But in some cases, problems may arise.

For example, consider two tests, `test1` and `test2` containing local sweep points `vdd=1 2 3 4` and `vdd=2 4` respectively. Here, if a `calcVal` expression in the second point of `test2`, `vdd=4` tries to get a value from `test1`, it would not know which point of `test1` to fetch the value from.

In such scenarios, using `?matchParams "all"` would solve the issue because `calcVal` will then look at all the points in the `test1` to find a point with `vdd=4` and will get the value from point four of `test1`.

But if you sweep `vdd=2 4` in `test1` and `vdd=1 2 3 4` in `test2`, the above argument would not solve the problem since `calcVal` will try to match all values and will not find a match for `vdd=1` and `vdd=3`. In this case, you can specify which value to match by using the argument `?matchParams list(list("vdd" 4))`

Let us add another variable `rload` to this setup with `rload=2` for `test1` and `rload=3` for `test2`. Now considering the local sweep point `vdd=1 2 3 4`, the above mentioned solution will not work since all the variables need to match and the value for `rload` is different for both the tests. In this case, you can use any of the following arguments as a solution.

Virtuoso ADE SKILL Reference

Output Functions

- `?matchParams list(list("vdd" 4) list("rload" 2))`
- `?matchParams list(list("vdd" 4)) ?ignoreParams list("rload")`

Case 7: Circular dependencies

An error message is displayed in case you have a circular dependency in `calcVal` expressions.

For example, consider a circular dependency between the `calcVal` expressions in four tests, `test1`, `test2`, `test3`, and `test4`. The `calcVal` expression in `test2` is dependent on `test1`. Accordingly, `test3` is dependent on `test2` and `test3` on `test4`. If `test1` has an additional dependency on `test4`, the setup contains a circular dependency. This would mean that `test1` is dependent on `test2`, `test3`, `test4` and finally on itself. This indicates that there is an issue with your setup and the dependency loop needs to be broken at one point.

Note: In case of a longer loop, the error message does not provide complete information. In the above situation, `calcVal` might report only the dependency between `test2` and `test3`. However, it could be that the wrong dependency is actually between `test3` and `test1`.

measureAcross

```
measureAcross(  
    t_expression  
    [ ?parameters l_parameters ]  
)  
=> x_scalar / o_waveform / nil
```

Description

Computes the given calculator expression across the parameters provided in the parameter list. Use this function only when you are creating expressions for outputs that can be evaluated across any dimension. These expressions have their eval type as any. If you specify this expression for other eval types, the parameter list is ignored.

The measureAcross function does not work in OCEAN standalone mode or in SKILL scripts for any purpose other than the creation of outputs.

Arguments

t_expression Calculator expression for an output.

?parameters l_parameters

A list of corner variables, swept variables, device parameters, or aliases that define the dimension for the measurements across any dimension.

The following aliases are supported:

- **Sweeps**: to specify that the calculation is to be done across all swept variables
- **Corners**: to specify that the calculation is to be done across all corner variables
- **mc_iteration**: to specify that the calculation is to be done across the Monte Carlo parameter `monteCarlo::param::sequence`. This alias is valid for the Monte Carlo run mode only.

Virtuoso ADE SKILL Reference

Output Functions

Value Returned

<i>x_scalar</i>	A scalar result of the expression.
<i>o_waveform</i>	A waveform result of the expression.
<i>nil</i>	An error in the expression

Examples

The following example shows how to compute the average current across two parameters, **vdd** and **temperature**:

```
measureAcross(average(Current ?overall t) ?parameters list("vdd" "temperature"))
```

The following expression calculates the average phase margin across two dimensions, the **vdd** design variable and Monte Carlo iterations:

```
measureAcross(average(PhaseMargin ?overall t) ?parameters list("vdd" "mc_iteration"))
```

The following expression calculates the average phase margin across two design variables, the **vdd** and **temperature** swept variables:

```
measureAcross(average(PhaseMargin ?overall t) ?parameters list("vdd" "temperature"))
```

The following expression calculates the average phase margin across the **vdd** design variable and all corners:

```
measureAcross(average(PhaseMargin ?overall t) ?parameters list("vdd" "Corners"))
```

The following expression calculates the average phase margin across all Monte Carlo iterations:

```
measureAcross(average(PhaseMargin ?overall t) ?parameters list("mc_iteration"))
```

The following expression calculates the current across the varying values of a parameter **M3 / fw** that is swept as **5.1u:0.1u:5.5u**:

```
measureAcross(Current ?parameters list("M3/fw"))
```

; In this example, Current is another expression, Current=abs(IDC("/V1/PLUS"))

Virtuoso ADE SKILL Reference

Output Functions

Test Functions

This topic lists the functions that let you manage tests in maestro cellviews.

[axlGetCornersForATest](#)

[axlGetEnabledGlobalVarPerTest](#)

[axlGetEnabledTests](#)

[axlGetOrigTestToolArgs](#)

[axlGetTest](#)

[axlGetTests](#)

[axlGetTestToolArgs](#)

[axlSaveResValue](#)

[axlSetTestToolArgs](#)

[axlToolSetSetupOptions](#)

[axlWriteOceanScriptLCV](#)

Related Topics

[Working with Tests](#)

axlGetCornersForATest

```
axlGetCornersForATest(  
    x_session  
    t_test  
)  
=> l_corners / nil
```

Description

Returns a list of corners enabled for the given test.

Arguments

<i>x_session</i>	Handle to the session.
<i>t_test</i>	Test name.

Value Returned

<i>l_corners</i>	List containing the names of corners associated with the given test and a list of corner variables and their value pairs.
<i>nil</i>	Unsuccessful operation.

Examples

The following example returns a list of all the enabled corners associated with test `Test1`:

```
data_session = axlGetWindowSession(hiGetCurrentWindow())  
axlGetCornersForATest(data_session "Test1")  
  
(("C1_VDD_2.2_Temp_1"  
  ("VDD" "2.2")  
    ("temperature" "75")  
    ("corModelSpec" "File=All#Global#gpdK045.scs Section=\"tt\";")  
  )  
)  
("C1_VDD_2.2_Temp_0"  
  ("VDD" "2.2")  
    ("temperature" "-25")  
    ("corModelSpec" "File=All#Global#gpdK045.scs Section=\"tt\";")  
)
```

Virtuoso ADE SKILL Reference

Test Functions

```
)  
)  
("" nil)  
)
```

Related Topics

[Working with Tests](#)

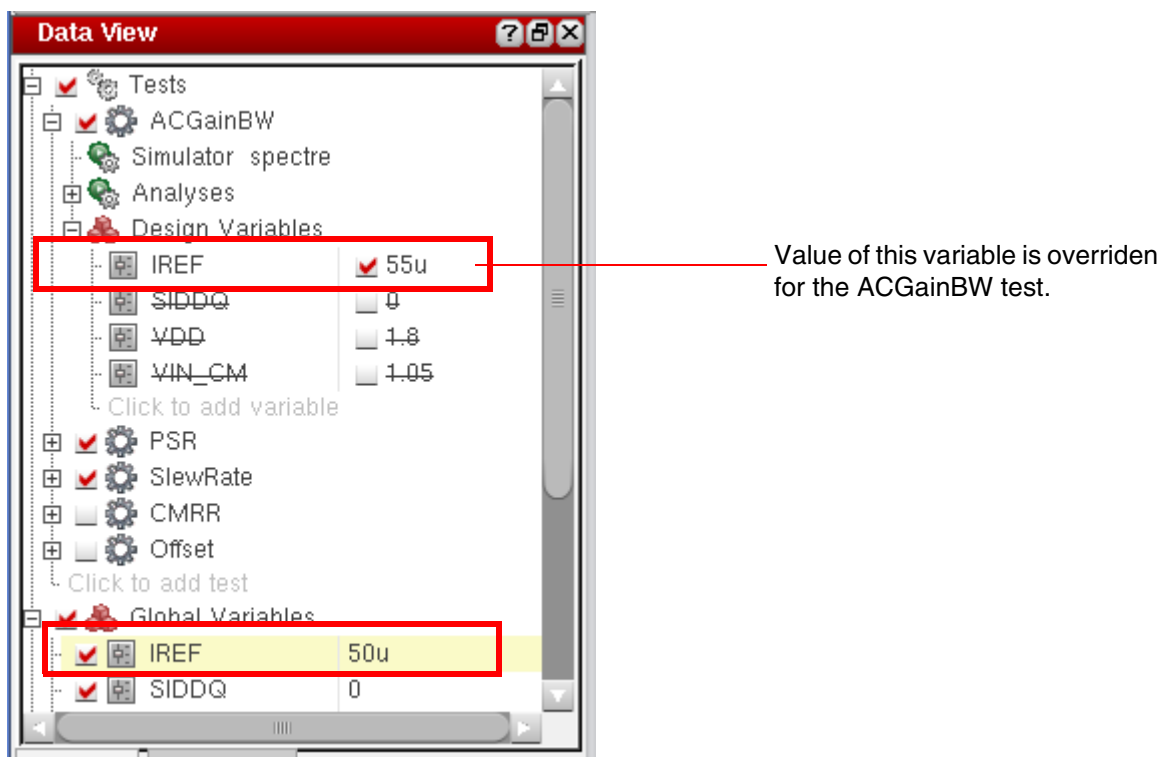
axlGetEnabledGlobalVarPerTest

```
axlGetEnabledGlobalVarPerTest(  
    x_hbdb  
    t_varName  
    t_test  
)  
=> t / nil
```

Description

Returns the status of a global variable for the given test. When a global variable is overridden for a test, this function returns `nil`, which implies that the value of the global variable will not be considered for the test. Instead, the test will use a local value set for that variable.

For example, in the figure shown below, the function will return `nil` for `IREF` because the global variable is not enabled or used for the test.



The function does not determine whether point sweeps or variables are enabled for simulation runs. Use the following functions for such cases:

- Use [axlGetAllVarsDisabled](#) to find if the variables are enabled for the simulation run. This function returns the status of the *Global Variables* check box in the *Data View* pane.

Virtuoso ADE SKILL Reference

Test Functions

- Use [axlGetAllSweepsEnabled](#) to find if the sweep points are enabled for the simulation run. This function returns the status of the *Point Sweep* check box in the *Run Summary* pane.
- Use [axlGetEnabled](#) to find if a specific variable is enabled.

Arguments

<code>x_hsdb</code>	Handle to the main setup database.
<code>t_varName</code>	Name of the global variable.
<code>t_test</code>	Name of the test.

Value Returned

<code>t</code>	The global variable is enabled for the given test.
<code>nil</code>	Unsuccessful operation.

Examples

The following example returns the status of the global variables `VDD` and `IREF` for the test `ACGainBW` when the setup is as shown in the example figure shown above:

```
s1 = axlGetWindowSession(hiGetCurrentWindow())
x_mainSDB=axlGetMainSetupDB(s1)
=> 1001
axlGetEnabledGlobalVarPerTest(x_mainSDB "VDD" "ACGainBW")
=>t
axlGetEnabledGlobalVarPerTest(x_mainSDB "IREF" "ACGainBW")
=> nil
```

Related Topics

[axlSetDesignVariablePerTest](#)

[Working with Tests](#)

axlGetEnabledTests

```
axlGetEnabledTests(  
    x_mainSDB  
)  
=> l_tests / nil
```

Description

Returns a list of tests enabled in the given setup database.

Arguments

<i>x_mainSDB</i>	Handle to the setup database.

Value Returned

<i>l_tests</i>	A list containing the handles and names of the tests enabled in the given setup.
<i>nil</i>	Unsuccessful operation.

Examples

The following example shows how to get the list of enabled tests:

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=>1001  
axlGetEnabledTests(x_mainSDB)  
=>((1005 "AC") (1013 "TEST"))
```

Related Topics

[Working with Tests](#)

axlGetOrigTestToolArgs

```
axlGetOrigTestToolArgs(  
    x_hsdb  
)  
=> l_toolOptions / nil
```

Description

Returns an associative list of original tool options set for the test before you ran the simulation after modifying the test setup.

Argument

<i>x_hsdb</i>	Handle to the test.

Value Returned

<i>l_toolOptions</i>	Associative list of original tool options set for the test before you ran a simulation after modifying the test setup.
<i>nil</i>	Unsuccessful operation.

Examples

The following example shows how to get an associative list of original tool options set for a test:

```
data_session = axlGetWindowSession(hiGetCurrentWindow())  
data_sdb=axlGetMainSetupDB(data_session)  
  
;; Get test args  
testh= axlGetTest( data_sdb "OpAmp1" )  
axlGetOrigTestToolArgs(_testh )  
'(("lib"    "opamplib")  
  ("cell"   "ampTest")  
  ("view"   "schematic")  
  ("sim"    "spectre")  
  ("path"   "./aState")  
  ("state"  "tran_state")  
)
```

Related Topics

[axlGetTestToolArgs](#)

axlGetTest

```
axlGetTest(  
    x_hsdb  
    t_test  
)  
=> x_test / nil
```

Description

Finds a test in the setup database and returns its handle.

Arguments

<i>x_hsdb</i>	Setup database handle.
<i>t_test</i>	Test name.

Value Returned

<i>x_test</i>	Test handle.
<i>nil</i>	Unsuccessful operation.

Examples

The following example finds a test in the `data_sdb` setup database.

```
data_session = axlGetWindowSession(hiGetCurrentWindow())  
data_sdb=axlGetMainSetupDB(data_session)  
;; Enable a test  
data_dead_band = axlGetTest( data_sdb "data_dead_band" )  
axlSetEnabled( data_dead_band t )  
  
;; Disable tests  
foreach( test cadr( axlGetTests( data_sdb ) )  
        axlSetEnabled( axlGetTest( data_sdb test ) nil )  
3
```

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

[axlGetTests](#)

Virtuoso ADE SKILL Reference

Test Functions

axlSetEnabled

axlGetTests

```
axlGetTests(  
    x_hsdb  
)  
=> l_tests / nil
```

Description

Returns a list containing a handle to all tests in the setup database and a list of all test names.

Arguments

<code>x_hsdb</code>	Setup database handle.

Value Returned

<code>l_tests</code>	List containing a handle to all tests in the setup database and a list of all test names.
<code>nil</code>	Unsuccessful operation.

Examples

The following example retrieves a list containing a handle to all tests in the `data_sdb` setup database and a list of all test names.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
;; Disable tests  
foreach( test cadr( axlGetTests( data_sdb ) )  
        axlSetEnabled( axlGetTest( data_sdb test ) nil )  
'((1011 "Trans_12u")  
  (1021 "dc_acI0G_2")  
)
```

Related Topics

[axlSetMainSetupDB](#)

[axlSetEnabled](#)

[axlGetTest](#)

axlGetTestToolArgs

```
axlGetTestToolArgs(  
    x_hsdb  
)  
=> l_toolOptions / nil
```

Description

Returns an associative list of tool option names and values for a test.

Arguments

<i>x_hsdb</i>	Handle to the test.

Value Returned

<i>l_toolOptions</i>	Associative list of tool option names and values for the test. Valid values when the tool is ADE are: ■ Library name: <code>lib t_libName</code> ■ Cell name: <code>cell t_cellName</code> ■ View name: <code>view t_viewName</code> ■ Simulator name: <code>sim t_simulator</code> ■ State name: <code>state t_stateName</code> ■ Path to ADE state: <code>path t_path</code>
<i>nil</i>	Unsuccessful operation.

Examples

The following example retrieves an associative list of tool option names and values for test with handle `testh`.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
  
;; Get test args  
testh= axlGetTest( data_sdb "OpAmp1" )  
axlGetTestToolArgs( testh )  
'(("lib"      "opamplib")  
  ("cell"     "ampTest")
```

Virtuoso ADE SKILL Reference

Test Functions

```
("view" "schematic")
("sim" "spectre")
("path" "./artist_state")
("state" "tran_state")
)
```

Related Topics

[axlGetOrigTestToolArgs](#)

[axlToolSetSetupOptions](#)

axlSaveResValue

```
axlSaveResValue(  
    t_resultName  
    g_resultValue  
)  
=> t / nil
```

Description

Adds a name and value to the results database for the current point. You can use this function to specify an OCEAN measurement that you want to appear on the Outputs assistant pane.

Arguments

<i>t_resultName</i>	Result name.
<i>g_resultValue</i>	Result value (number, list, or waveform).

Value Returned

<i>t</i>	A name and value were added to the results database.
<i>nil</i>	Unsuccessful operation.

Examples

The following example adds the specified name and value to `result`.

```
axlSaveResValue( "PhaseMargin" result )  
=> t
```

Related Topics

[Working with Tests](#)

axlSetTestToolArgs

```
axlSetTestToolArgs(  
    x_hsdb  
    l_toolOptions  
)  
=> x_hsdb / nil
```

Description

Sets the tool options for the test.

Arguments

<i>x_hsdb</i>	Handle to the test for which you need to set options.
<i>l_toolOptions</i>	Associative list of tool option names and values for the test. Valid Values when the tool is ADE: ■ Library name: <code>lib t_libName</code> ■ Cell name: <code>cell t_cellName</code> ■ View name: <code>view t_viewName</code> ■ Simulator name: <code>sim t_simulator</code> ■ State name: <code>state t_stateName</code> ■ Path to ADE state: <code>path t_path</code>

Value Returned

<i>x_hsdb</i>	Setup database handle.
<i>nil</i>	Unsuccessful operation.

Examples

The following example shows how to load a state saved from ADE L for a test into ADE XL:

```
session = axlGetWindowSession()  
sdb = axlGetMainSetupDB(session)  
libName = axlGetSessionLibName(session)  
cellName = axlGetSessionCellName(session)
```

Virtuoso ADE SKILL Reference

Test Functions

```
handleToTest = axlPutTest(sdb "state_test" "ADE")

axlSetTestToolArgs(handleToTest list( list("lib" libName) list("cell" cellName)
list("view" "schematic") list("sim" "spectre")))

testSession=asiGetSession(handleToTest)

asiLoadState(testSession ?name "spectre_state1" ?lib libName ?cell cellName
?simulator "spectre")
```

Related Topics

Working with Tests

axlToolSetOriginalSetupOptions

```
axlToolSetOriginalSetupOptions(  
    t_session  
    t_test  
    l_toolOptions  
    [ ?history x_history ]  
)  
=> t / nil
```

Description

Sets options to their original values for the tool instance associated with the specified session and test.

Arguments

<i>t_session</i>	Session name.
<i>t_test</i>	Test name.
<i>l_toolOptions</i>	Associative list of original option names and values for the tool instance. Valid Values for tool instance ADE: ■ Library name: lib <i>t_libName</i> ■ Cell name: cell <i>t_cellName</i> ■ View name: view <i>t_viewName</i> ■ Simulator name: sim <i>t_simulator</i> ■ State name: state <i>t_stateName</i> ■ Path to ADE state: path <i>t_path</i>
<i>?history x_history</i>	An integer representing the history entry.

Virtuoso ADE SKILL Reference

Test Functions

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Examples

The following example sets options to their original values for tools open in `session0`.

```
axlToolSetOriginalSetupOptions( "session0" "delayTest" axlGetOrigTestToolArgs (
1031 ) )
=> t
```

Related Topics

[axlGetOrigTestToolArgs](#)

[axlToolSetSetupOptions](#)

axlToolSetSetupOptions

```
axlToolSetSetupOptions(  
    t_session  
    t_test  
    l_toolOptions  
)  
=> t / nil
```

Description

Sets the option values for the tool instance associated with the specified session and test.

Arguments

<i>t_session</i>	Session name.
<i>t_test</i>	Test name.
<i>l_toolOptions</i>	Associative list of original option values for the tool instance. Valid Values for tool instance ADE: ■ Library name: <code>lib t_libName</code> ■ Cell name: <code>cell t_cellName</code> ■ View name: <code>view t_viewName</code> ■ Simulator name: <code>sim t_simulator</code> ■ State name: <code>state t_stateName</code> ■ Path to ADE state: <code>path t_path</code>

Value Returned

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

Examples

The following example sets the option values for the tool instance associated with session `session0` and test `delayTest`.

Virtuoso ADE SKILL Reference

Test Functions

```
axlToolSetSetupOptions( "session0" "delayTest" axlGetTestToolArgs( 1031 ) )
=> t
```

The following example shows how to modify analyses for multiple tests together:

```
LJNaxlUpdateAnalysisFieldValAllTests (analysis field value)
;   Update all tests that contain the specified analysis type.
;   e.g. (LJNaxlUpdateAnalysisFieldValAllTests 'tran 'stop 100n)

LJNaxlUpdateTranStopTimeAllTests (stopTime)

;   Shortcut function to update tran stop time.
;   e.g. LJNaxlUpdateTranStopTimeAllTests 100n
;   A cellview must be opened. Make sure the view is in focus before
;   entering the command in the CIW. All tests that contain the specified analysis
;   type will be updated.

;load this file in the .cdsinit:
(when !(axlIsICRPPProcess)
  load("<this filename>")
)

;Example: Add $HOME/skill directory to SKILL path:
setSkillPath(cons("~/skill" getSkillPath()))
```

Note: The GUI does not update automatically with the new analysis value although the value has been updated. If you close the view and open it again the value is updated or if you open the 'Choosing Analyses' form and select OK the value (e.g. tran stop time) is updated.

```
(defun LJNaxlUpdateAnalysisFieldValAllTests (analysisType field value)
  (let (axl tests testSession oSession analysisName analysisList)
    axl = (LJNaxlGetCurrentWindowSessionAndSetupDB)
    tests = (cadr (axlGetTests axl->sdbh) )
    ; Special case to convert tran stop time to string if necessary (handle 1e-9 or
    ; ln without double-quotes)

    (when (and (analysisType == 'tran) (field == 'stop) (not (stringp value)))
      value = (aelSuffixNotation value)
    )

    (foreach testName tests
      (printf "Test %s\n" testName)
      testSession = (axlGetToolSession axl->sess testName) ;sev session
      oSession = (sevEnvironment testSession) ;oasis session
      analysisName = (asiGetAnalysisName (asiGetAnalysis oSession
analysisType))
      analysisList = (asiGetEnabledAnalysisList oSession)
      (foreach analysis analysisList
        (printf "\tAnalysis name = %L\n" analysis->name)
        (when analysis->name == analysisName
          (printf "\t\tSetting %A = %A\n" field value)
          (asiSetAnalysisFieldVal analysis field value)
          (printf "\t\tNew val = %A\n" (asiGetAnalysisFieldVal analysis field
)))
        )
      )
    )
  )
```

Virtuoso ADE SKILL Reference

Test Functions

```
(defun LjNaxlUpdateTranStopTimeAllTests (stopTime)
  (let ()
    (LjNaxlUpdateAnalysisFieldValAllTests 'tran 'stop stopTime)
  )
)

;Utility functions

(defun LjNaxlGetCurrentWindowSessionAndSetupDB ()
  (let (sess sdbh ret)
    sess = (LjNaxlGetCurrentWindowSession)
    (unless sdbh
      (error "Cannot find setupDB handle.\n")
    )
    ret=(ncons nil)
    ret->sess=sess
    ret->sdbh=sdbh
    ret
  )
)

(defun LjNaxlGetCurrentWindowSetupDB ()
  (let (sess sdbh)
    sess = (LjNaxlGetCurrentWindowSession)
    sdbh = (axlGetMainSetupDB sess)
    (unless sdbh
      (error "Cannot find setupDB handle.\n")
    )
    sdbh
  )
)

(defun LjNaxlGetCurrentWindowSession ()
  (
    let (sess)
      sess = (axlGetWindowSession (hiGetCurrentWindow))
      (unless sess
        (error "Cannot find the session from current window. Select the ADE
Explorer window to set the current window.\n")
      )
      sess
    )
  )

;To get a list of available analysis fields for an analysis:
;e.g:
;(LjNdisplayAnalysisField 'tran)
;(LjNdisplayAnalysisField 'dc)

(defun LjNdisplayAnalysisField (analysisType)
  (let (session analysis)
    session = asiGetCurrentSession()
    analysis = asiGetAnalysis(session analysisType)
    asiDisplayAnalysisField(analysis)
  )
)
```


Related Topics

[axlGetTestToolArgs](#)

[axlToolSetOriginalSetupOptions](#)

axlCustomADETestName

```
axlCustomADETestName (  
    t_libName  
    t_cellName  
    t_viewName  
    t_stateName  
)  
=> t_testName / nil
```

Description

A function that can be defined in the .cdsinit file or in CIW to customize the test name in ADE XL. The default format of test name is

`<library-name>:<cell-name>:<sequence-num>`. You can use this function to customize the format to be used for test name. While creating a new test, ADE XL checks for the existence of a definition of `axlCustomADETestName`. If found, ADE XL calls this function and uses the returned value as the test name.

Note: The inclusion of `<sequence-num>` in the test name is controlled by the `initiallyAddNameUniqifier` environment variable. The `axlCustomADETestName` function defines only the initial part of the test name.

Arguments

<code>t_libName</code>	Name of the library being used to create a test.
<code>t_cellName</code>	Name of the cell being used to create a test.
<code>t_viewName</code>	Name of the view being used to create a test.
<code>t_stateName</code>	Name of the state being used to create a test.

Value Returned

<code>t_testName</code>	A customized name for the test being created.
<code>nil</code>	The test name resulting from the given logic is invalid. In this case, ADE XL uses the default format for test name.

Examples

The following example shows how to customize the test names to include the project name.

```
(define (axlCustomADETestName lib cell view state)
```

Virtuoso ADE SKILL Reference

Test Functions

```
(strcat lib ":" cell "myProj"))
```

With this definition, a new test will be named as:

library-name:cell-name:myProj:sequence-num

axlWriteOceanScriptLCV

```
axlWriteOceanScriptLCV(  
    t_fileName  
    t_libraryName  
    t_cellName  
    t_viewName  
=> t / nil
```

Description

Writes an OCEAN script for the given cellview in the specified file. If `axlWriteOceanScriptLCV()` is used in an existing session, it will write the current in-memory values and not the values saved on the disk.

If a file already exists with the same name, it is overwritten with the new one.

Arguments

<i>t_fileName</i>	Name of the OCEAN file in which you need to save the OCEAN script.
<i>t_libraryName</i>	Library name in the cellview
<i>t_cellName</i>	Cell name in the cellview
<i>t_viewName</i>	Name of the cellview

Value Returned

<i>t</i>	An OCEAN script is written for the given view.
<i>nil</i>	The function cannot write an OCEAN script.

Examples

The following example saves an OCEAN script for the `opamplib:ampTest:adex1` cellview in the `oceanScript.ocn` file:

```
axlWriteOceanScriptLCV("oceanScript.ocn" "opamplib" "ampTest" "adex1")  
=> t
```

Virtuoso ADE SKILL Reference

Test Functions

Related Topics

[Working with OCEAN-Based Tests](#)

Virtuoso ADE SKILL Reference

Test Functions

Specification Functions

This topic lists the SKILL functions that let you manage output specifications in ADE sessions.

[axlAddSpecToOutput](#)

[axlGetSpecs](#)

[axlGetSpec](#)

[axlGetSpecData](#)

[axlGetSpecWeight](#)



Earlier, `axlPutSpec` was used to add a specification for an output and the `axlSetSpec*` functions, such as `axlSetSpecMax` or `axlSetSpecRange`, were used to add specification values and other details. It is now recommended to use the `axlAddSpecToOutput` instead of `axlPutSpec` or `axlSetSpec*` to add a specification for an output.

axlAddSpecToOutput

```
axlAddSpecToOutput (
    x_hsdb
    t_testName
    t_resultName
    [ ?min g_minValue ]
    [ ?max g_maxValue ]
    [ ?gt g_greaterThanValue ]
    [ ?lt g_lessThanValue ]
    [ ?range g_rangeValues ]
    [ ?tol g_toleranceValue ]
    [ ?info g_info ]
    [ ?weight g_weightingFactor ]
    [ ?corner g_cornerName ]
)
=> t / t_error
```

Description

Adds a specification to an output defined for a test. You can also use this function to modify an existing specification for an output.

Arguments

<i>x_hsdb</i>	Setup database handle
<i>t_testName</i>	Name of the test.
<i>t_resultName</i>	Name of the result.
?min <i>g_minValue</i>	Value for the minimum specification.
?max <i>g_maxValue</i>	Value for the maximum specification.
?gt <i>g_greaterThanValue</i>	Value for the greater than specification.
?lt <i>g_lessThanValue</i>	Value for the less than specification.
?range <i>g_rangeValues</i>	A range of values for the range specification.
?tol <i>g_toleranceValue</i>	Value for the tolerance specification.

Virtuoso ADE SKILL Reference

Specification Functions

`?info g_info` Any information string for information specification.

`?weight g_weightingFactor`
A weighting factor for this specification.

`?corner g_cornerName`
Name of the corner for which the specification is to be enabled. This argument helps to override a specification for a particular corner.

By default, a specification defined for a measurement applies to all the corners enabled for the test. To change the specification for a particular corner, specify the name of that corner in this argument. In the ADE XL GUI, you can view the overridden corner name in the Override Specifications form.

Value Returned

`t` A specification is added to the output.

`t_error` The command is unsuccessful. An error message is displayed.

Examples

The following example shows how to add different types of specification to an existing output for a test:

```
session = (axlGetWindowSession)
sdb = (axlGetMainSetupDB session)

axlAddSpecToOutput(sdb "TRAN" "" ?gt "1")
=> t

axlAddSpecToOutput(sdb "TRAN" "" ?lt "1" ?weight "1" ?corner "CC_1")
=> t

axlAddSpecToOutput(sdb "TRAN" "" ?range 1:3 )
=> t
; The above statement removes the existing lt spec from the output and
; assigns the range spec.
axlAddSpecToOutput(sdb "TRAN" "" ?info "info spec")
=> t
axlAddSpecToOutput(sdb "TRAN" "gain" ?tol 1:2 ?corner "CC_2")
=> t
```

Virtuoso ADE SKILL Reference

Specification Functions

Related Topics

[axlGetSpecs](#)

[axlGetSpec](#)

[axlGetSpecData](#)

[axlGetSpecWeight](#)

axlGetSpecs

```
axlGetSpecs(  
    x_hsdb  
)  
=> l_list / nil
```

Description

Returns a list containing a handle to all specifications in the setup database and a list of all specification names.

Arguments

<i>x_hsdb</i>	Setup database handle

Value Returned

<i>l_list</i>	List containing a handle to all specifications in the setup database and a list of all specification names.
<i>nil</i>	The command is unsuccessful.

Examples

The following example shows how to get all the existing specifications from the given setup database:

```
session = axlGetWindowSession()  
x_hsdb = axlGetMainSetupDB(session)  
axlGetSpecs( x_hsdb )  
=> (1002 ( "opamplib:ampTest:1.gain" "opamplib:ampTest:1.bandwidth" ) )
```

Related Topics

[axlAddSpecToOutput](#)

[axlGetSpec](#)

[axlGetSpecData](#)

[axlGetSpecWeight](#)

axlGetSpec

```
axlGetSpec (  
    x_hsdb  
    t_specName  
)  
=> x_spec / 0
```

Description

Finds the named specification in the setup database and returns its handle.

Arguments

<i>x_hsdb</i>	Setup database handle
<i>t_specName</i>	Specification name

Value Returned

<i>x_spec</i>	Specification handle
0	The command is unsuccessful.

Examples

The following example shows how to get the handle to an existing specification from the given setup database:

```
session = (axlGetWindowSession)  
sdb = (axlGetMainSetupDB session)  
axlGetSpec(sdb "opamplib:ampTest:1.gain")  
=> 1002
```

For more examples, see [axlGetSpecData](#).

Related Topics

[axlAddSpecToOutput](#)

[axlGetSpecs](#)

[axlGetSpecData](#)

Virtuoso ADE SKILL Reference

Specification Functions

axlGetSpecWeight

axlGetSpecData

```
axlGetSpecData(  
    x_hsdb  
    t_resultName  
    t_testName  
    [ t_cornerName ]  
)  
=> l_specDetails / nil
```

Description

Returns the specification for the given result, test, and corner combination in the given setup database.

Arguments

<i>x_hsdb</i>	Handle to the setup database.
<i>t_resultName</i>	Name of the measure.
<i>t_testName</i>	Name of the test.
<i>t_cornerName</i>	Name of the corner.

Value Returned

<i>l_specDetails</i>	Details of the specification.
<i>nil</i>	No specification exists for the given result.

Examples

The following examples show how you can get the specification details for a given result:

```
session = (axlGetWindowSession)  
x_mainSDB = (axlGetMainSetupDB session)  
axlGetSpecData(x_mainSDB "pw" "opamplib:ampTest:1" "C0")  
=>("gt" "10")  
  
axlGetSpecData(x_mainSDB "pw" "opamplib:ampTest:2")  
=> ("range" "10" "20")  
  
axlGetSpecData(x_mainSDB "pw" "opamplib:ampTest:3")
```

Virtuoso ADE SKILL Reference

Specification Functions

```
=> nil
```

The following example displays specification details for all the results in the active setup database.

```
session = (axlGetWindowSession)
x_mainSDB = (axlGetMainSetupDB session)
(
foreach spec (cadr (axlGetSpecs x_mainSDB))
  specname=parseString(spec ".")
  test=car(specname)
  result=cadr(specname)
  specval=axlGetSpecData(1001 result test)
  printf("test=%s, result=%s, specValue=%L \n" test result specval)
)
```

The above code displays all the specification details, as shown below.

```
=> test=Two_Stage_Opamp:OpAmp_AC_top:1, result=Current, specValue=("lt" "1m")
test=Two_Stage_Opamp:OpAmp_AC_top:1, result=Gain, specValue=("max" "45")
test=Two_Stage_Opamp:OpAmp_AC_top:1, result=UGF, specValue=("gt" "250M")
test=Two_Stage_Opamp:OpAmp_TRAN_top:1, result=SettlingTime, specValue=("lt"
"9n")
test=Two_Stage_Opamp:OpAmp_TRAN_top:1, result=Swing, specValue=("gt" "1")
test=Two_Stage_Opamp:OpAmp_AC_top:1:1, result=Current, specValue=("lt"
"1e-3")
test=Two_Stage_Opamp:OpAmp_AC_top:1:1, result=Gain, specValue=("max" "45.0")
test=Two_Stage_Opamp:OpAmp_AC_top:1:1, result=UGF, specValue=("gt" "250e6")
```

Related Topics

[axlAddSpecToOutput](#)

[axlGetSpecs](#)

[axlGetSpec](#)

[axlGetSpecWeight](#)

axlGetSpecWeight

```
axlGetSpecWeight (
    x_spec
)
=> t_weight / nil
```

Description

Returns the weight value for a specification. This value represents the weighting factor for the specification during optimization.

Arguments

<i>x_spec</i>	Specification handle

Value Returned

<i>t_weight</i>	Weight value
<i>nil</i>	The command is unsuccessful.

Examples

The following example retrieves the weight value for specification *spec*.

```
session = (axlGetWindowSession)
x_mainSDB = (axlGetMainSetupDB session)
spec = axlGetSpec(x_mainSDB "gain" )
axlGetSpecWeight( spec )
=> 1
```

Related Topics

[axlAddSpecToOutput](#)

[axlGetSpecs](#)

[axlGetSpec](#)

[axlGetSpecData](#)

Corners Functions

The following corner SKILL functions can be used to work with corners in a maestro cellview.

[axlGetAllCornersEnabled](#)

[axlCorners](#)

[axlGetCorner](#)

[axlGetCorners](#)

[axlGetCornerDisabledTests](#)

[axlGetCornerCountForName](#)

[axlGetCornerNameForCurrentPointInRun](#)

[axlGetNominalCornerEnabled](#)

[axlGetNominalCornerTestEnabled](#)

[axlGetStatVars](#)

[axlLoadCorners](#)

[axlLoadCornersFromPcfToSetupDB](#)

[axlPlotAcrossDesignPoints](#)

[axlPutCorner](#)

[axlPutDisabledCorner](#)

[axlSetDefaultCornerEnabled](#)

[axlSetAllCornersEnabled](#)

[axlSetCornerName](#)

Virtuoso ADE SKILL Reference

Corners Functions

axlSetCornerTestEnabled

axlSetNominalCornerEnabled

axlSetNominalCornerTestEnabled

axlSetWCCTime

axlGetWCCCorner

axlGetWCCHistory

axlGetWCCResult

axlGetWCCSpec

axlGetWCCSpecs

axlGetWCCTest

axlGetWCCTime

axlGetWCCRangeBound

axlGetWCCVarMonotonicity

axlGetWCCVars

axlGetAllCornersEnabled

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

Description

Returns the selection status of the *Corners* check box in the Run Summary assistant in ADE Assembler or the Explorer Run Summary assistant in ADE Explorer.

Arguments

<i>x_mainSDB</i>	Setup database handle.

Value Returned

t	The <i>Corners</i> check box is selected.
nil	The <i>Corners</i> check box is deselected.

Examples

The following example displays the status of the *Corners* check box in the Run Summary assistant in ADE Assembler or the Explorer Run Summary assistant in ADE Explorer:

```
axlsession=axlGetWindowSession(hiGetCurrentWindow())  
=> "session0"  
x_mainSDB=axlGetMainSetupDB(axlsession)  
=> 1001  
axlGetAllCornersEnabled(x_mainSDB)  
=> t
```

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

axlCorners

```
axlCorners(  
    t_session  
    [ g_refresh ]  
)  
=> t / nil
```

Description

Opens the Corners Setup form.

If the Corners Setup form is not open, ADE Explorer or ADE Assembler opens the form and brings it to focus. If the form is already open, it is brought to focus. In the latter case, the argument *g_refresh* specifies whether any change related to corners and tests are to be reflected in the Corners Setup form.

Arguments

t_session

String representing the ADE session name.

g_refresh

Specifies if the changes related to corners and tests in the setup database are automatically reflected in the Corners Setup form while it is already open.

The default value is `nil`, which indicates that you need to close and re-open the form to view the updates.

When this variable is set to `t`, updates are automatically reflected in the form.

Note: This argument is helpful only when the Corners Setup form is already open.

Value Returned

`t`

The Corners Setup form was opened.

`nil`

The Corners Setup form could not be opened.

Examples

The following example opens the Corners Setup form for the current session.

Virtuoso ADE SKILL Reference

Corners Functions

```
session_name = axlGetWindowSession()  
=>"session0"  
  
;; Load Corner Setup User Interface Form  
axlCorners(session_name t)  
=> t
```

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

axlGetCorner

```
axlGetCorner(  
    x_mainSDB  
    t_cornerName  
)  
=> x_corner / nil
```

Description

Finds a corner by its name and returns a handle to that corner.

Arguments

<i>x_mainSDB</i>	Specifies the setup database handle.
<i>t_cornerName</i>	Specifies the corner name.

Value Returned

<i>x_corner</i>	Handle to a corner.
<i>nil</i>	The command was unsuccessful.

Examples

The following example shows how to find a corner with name VDD_C0.

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
cHandle = axlGetCorner(x_mainSDB "VDD_C0")  
=> 1340  
  
; you can further use this corner handle to modify or remove the corner  
axlRemoveElement(cHandle)
```

The following example disables all the corners in the current session.

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
;; Disable corners  
foreach(corners cadr( axlGetCorners(x_mainSDB) )  
    axlSetEnabled(axlGetCorner(x_mainSDB corner) nil))  
=> ("C0" "C1" "C2")
```

Virtuoso ADE SKILL Reference

Corners Functions

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

[axlSetEnabled](#)

[axlGetCorners](#)

axlGetCorners

```
axlGetCorners(  
    x_mainSDB  
)  
=> l_corners / nil
```

Description

Returns a list containing a handle to all corners and a list of names of corners and corner groups in the setup database.

Arguments

<i>x_mainSDB</i>	Setup database handle.

Value Returned

<i>l_corners</i>	List containing a handle to the corners and a list of names of the corners and corner groups in the setup database.
<i>nil</i>	The command was unsuccessful.

Examples

The following example gets a list of all the corners and corner groups in the current session:

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( session )  
=>1001  
axlGetCorners(x_mainSDB)  
=> (1003 ("C0" "C1" "C2_0_0" "C2_0_1" "C2_0_2")  
)
```

The following example shows how to remove all the corners from the setup database of the current session:

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( session )  
=>1001  
axlGetCorners(x_mainSDB)  
=> (1003  
("C0" "C1" "C2_0_0" "C2_0_1" "C2_0_2")  
)
```


Virtuoso ADE SKILL Reference

Corners Functions

```
axlRemoveElement(1003)
=> t
; this code removes all the existing corners from the setup database
)
```

Related Topics

[axlGetWindowSession](#)

[axlGetMainSetupDB](#)

[axlRemoveElement](#)

[axlGetCorner](#)

axlGetCornerDisabledTests

```
axlGetCornerDisabledTests(  
    x_cornerHandle  
)  
=> l_testNames / nil
```

Description

Returns a list containing a handle to the disabled tests and the names of tests that are disabled for the given corner.

Arguments

<i>x_cornerHandle</i>	Handle to the corner for which you need to find the disabled tests.

Value Returned

<i>l_testNames</i>	List containing a handle to the disabled tests and the names of tests that are disabled for the given corner.
<i>nil</i>	The command was unsuccessful.

Examples

The following example gets a list of all the test names disabled for corner C1.

```
session = axlGetWindowSession()  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
x_cHandle= axlGetCorner(x_mainSDB "C1")  
=> 1065  
axlGetCornerDisabledTests(x_cHandle)  
=> ( (3978 ("AC" "TRAN")) )
```

Related Topics

[Working with Corners](#)

axlGetCornerCountForName

```
axlGetCornerCountForName(  
    x_mainSDB  
    t_cornerGroup  
)  
=> x_cornerCount / -1
```

Description

Returns the count of individual corners contained in the specified corner group.

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>t_cornerGroup</i>	Name of the corner group.

Value Returned

<i>x_cornerCount</i>	Number of corners found in the corner group.
-1	Values for <i>x_mainSDB</i> or <i>t_cornerGroup</i> were not found.

Examples

The following example shows how to find the number of corners present in a corner group:

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( session )  
=>1001  
axlGetCorners(x_mainSDB)  
=> (1003  
    ("C0" "C1" "C2_0_0" "C2_0_1" "C2_0_2"))  
axlGetCornerCountForName 1003 "C1"  
=> 1  
axlGetCornerCountForName 1003 "C2_0_0"  
=> 6
```

Related Topics

axlGetCornerNameForCurrentPointInRun

```
axlGetCornerNameForCurrentPointInRun(  
    )  
=> t_cornerName
```

Description

Returns the name of the corner for the current point being simulated. This function is useful for customized processing or debugging in the OCEAN script-based measures.

Arguments

None

Value Returned

<i>t_cornerName</i>	Name of the corner for the current point being simulated. For nominal corner, the text " " is returned.

Examples

Returns the name of the corner for the current point being simulated.

```
axlGetCornerNameForCurrentPointInRun()  
=> "cor_77"
```

Here, `cor_77` is the current corner being run.

Related Topics

[Loading an OCEAN or a MATLAB Measurement](#)

axlGetNominalCornerEnabled

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

Description

Checks whether the nominal corner is enabled in the specified setup database. This is same as the status of the *Nominal* check box in the Run Summary assistant.

Arguments

<i>x_mainSDB</i>	Setup database handle.

Value Returned

t	Nominal corner is enabled.
nil	Nominal corner is disabled.

Examples

The following example returns the status of the nominal corner:

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( session )  
=>1001  
axlGetNominalCornerEnabled(x_mainSDB)  
=>t
```

Related Topics

axlGetNominalCornerTestEnabled

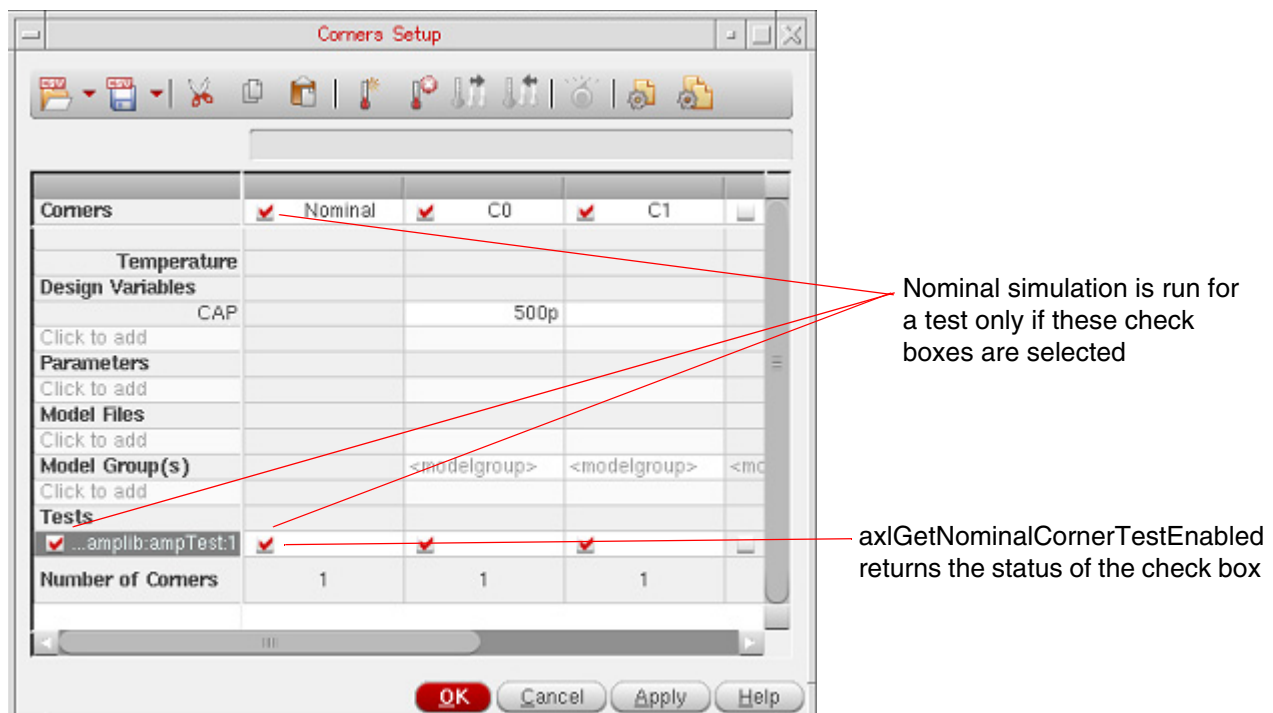
```
axlGetNominalCornerTestEnabled(  
    t_testHandle  
)  
=> t / nil
```

Description

Returns the status of the check box corresponding to the given test in the *Nominal* column on the Corners Setup form.

Nominal simulation for a test is run only if the following conditions are met:

- The check box before the column heading in the *Nominal* column on the Corners Setup form is selected.
- The check box before the test name in the row header for the tests is selected.
- The check box in the cell corresponding to the given test in the *Nominal* corner is enabled.



Virtuoso ADE SKILL Reference

Corners Functions

Arguments

<code>t_testHandle</code>	Handle to the test for which you need to check whether nominal simulation can be run.

Value Returned

<code>t</code>	The command was successful.
<code>nil</code>	The command could not be run.

Examples

The following example returns the status of the nominal corner:

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
x_testHandle = axlGetTest(x_mainSDB "AC")
axlGetNominalCornerTestEnabled(x_testHandle)
=> t
```

Related Topics

[axlGetNominalCornerEnabled](#)

[axlGetEnabled](#)

axlGetStatVars

```
axlGetStatVars(  
    x_mainSDB  
    x_cornerId  
)  
=> l_vars / nil
```

Description

Returns a list of the names of statistical variables that use the `statistical::mismatch` or `statistical::global` prefix for the given statistical corner.

Arguments

<code>x_mainSDB</code>	Handle to the active setup database.
<code>x_cornerID</code>	ID of the statistical corner.

Value Returned

<code>l_vars</code>	A list of statistical variables.
<code>nil</code>	The command was unsuccessful.

Examples

The following example returns a list of statistical variables that use the given prefixes.

```
x_mainSDB=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))  
=>1002  
  
cornerName=axlGetCorner(x_mainSDB "Stat_seq_1_params")  
=>44567  
  
statvarsList=axlGetStatVars(cornerName)  
=> (("statistical:mismatch:I0.M7:parl2" "1.546454185")  
    ("statistical:mismatch:I0.M7:plo_dxw" "1.412371304")  
    ("statistical:mismatch:I0.M5:plo_dx1" "-0.8702216304")  
    ("statistical:mismatch:I0.M2:plo_dxw" "-1.253658311")  
    ("statistical:mismatch:I0.M9:parl1" "-0.878480435")  
    ("statistical:mismatch:I0.M1:plo_tox" "0.8711784603")  
    ("statistical:mismatch:I0.M1:parl1" "0.689194273")  
    ("statistical:mismatch:I0.M6:plo_dxw" "-1.139652443")  
    ("statistical:mismatch:I0.M5:plo_dxw" "0.3336112529")  
    ("statistical:mismatch:I0.M3:plo_dx1" "-1.544827915")  
    ("statistical:mismatch:I0.M2:plo_tox" "0.01353515771"))
```


Virtuoso ADE SKILL Reference

Corners Functions

Related Topics

[axlGetNominalCornerEnabled](#)

axlLoadCorners

```
axlLoadCorners(  
    x_mainSDB  
    t_SDBfileName  
)  
=> x_cornersHandle / 0
```

Description

Loads a set of corners from the specified XML file in which the corners were saved earlier.

This function removes all the existing corners from the setup before creating the corners loaded from the specified setup database (XML) file.

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>t_SDBfileName</i>	Path to the setup database file from which corner details are to be loaded.

Value Returned

<i>x_cornersHandle</i>	A set of corners were loaded from the specified XML file.
0	The command was unsuccessful.

Examples

The following example loads the corners from the `co.sdb` file into the current session.

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( session  
=>1001  
axlLoadCorners(x_mainSDB "/home/user1/co.sdb")  
1003
```

Virtuoso ADE SKILL Reference

Corners Functions

Related Topics

axlLoadCornersFromPcfToSetupDB

```
axlLoadCornersFromPcfToSetupDB (  
    t_session  
    t_fileName  
    t_testNameList  
    g_overwriteExistingCorners  
)  
=> t / nil
```

Description

Imports a set of predefined corners from a process customization file into the corners setup for the given session.

Arguments

<i>t_session</i>	Name of session in which you want to import corners.
<i>t_fileName</i>	Path to the process customization file (PCF).
<i>t_testNameList</i>	List of tests for which the imported corners should be enabled. Separate the test names in the list using a blank space.
<i>g_overwriteExistingCorners</i>	Flag to specify how corners should be imported when an existing corner and a corner defined in the PCF file have the same name. Valid values are: ■ <i>t</i>: If an existing corner and a corner defined in the PCF file have the same name, overwrite the existing corner with the corner defined in the PCF file. ■ <i>nil</i>: If an existing corner and a corner defined in the PCF file have the same name, import the corner defined in the PCF file with a different name. For example, if there is an existing corner named <i>C1</i> and the PCF file has a corner named <i>C1</i>, the corner in the PCF file are imported as <i>C1_0</i>.

Virtuoso ADE SKILL Reference

Corners Functions

Value Returned

t	Corners were imported from the specified PCF file.
nil	The command could not be run.

Examples

The following example loads corners from a PCF file to the setup database:

```
session = axlGetWindowSession()
=> "session0"
axlLoadCornersFromPcfToSetupDB("session0" "./myCorners.pcf" "\"test1\" \"test2\""
"t")
=> t
```

Related Topics

axlPlotAcrossDesignPoints

```
axlPlotAcrossDesignPoints(  
    t_session  
    t_testName  
    t_historyName  
    t_outputName  
    t_cornerName  
)  
=> t / nil
```

Description

Plots an output across all the design points for a particular corner.

Arguments

<i>t_session</i>	Name of the session.
<i>t_testName</i>	Name of the test.
<i>t_historyName</i>	Name of the history from which results are to be used.
<i>t_outputName</i>	Name of the output to be plotted across corners.
<i>t_cornerName</i>	Name of the corner for which the results are to be plotted.

Value Returned

<i>t</i>	The results are successfully plotted.
<i>nil</i>	The command could not be run.

Examples

The following example plots the output `OPT_V` across all the design points for corner `C0_0`:

```
session = axlGetWindowSession()  
=>"session0"  
axlPlotAcrossDesignPoints("session" "Tran_sim" "Interactive.0" "OPT_v" "C0_0")  
=>t
```

Related Topics

axlPutCorner

```
axlPutCorner(  
    x_mainSDB  
    t_cornerName  
)  
=> x_corner / nil
```

Description

Adds a new corner by the given name and returns a handle to that corner. If a corner already exists with the same name, the function returns the handle to that corner.

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>t_cornerName</i>	Corner name.

Value Returned

<i>x_corner</i>	Handle to a corner.
<i>nil</i>	The command could not be run.

Example

The following example adds a new corner, `testC`, to the existing database and also assigns value to the VDD variable for that corner:

```
session = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( session )  
=>1001  
axlPutCorner(x_mainSDB "testC")  
=>2080  
axlPutVar(2080 "VDD" "2.0")  
=>2082
```

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

axlPutDisabledCorner

```
axlPutDisabledCorner(  
    x_testHandle  
    t_cornerName  
)  
=> x_disabledcorner / nil
```

Description

Adds a new corner by the given name and returns a handle to that corner. If a corner already exists with the same name, the function returns the handle to that corner. In addition, the corner is disabled for the specified test name, but enabled for other tests in the session.

You can also use this function to disable a specific corner for a particular test.

Arguments

<i>x_testHandle</i>	Test handle.
<i>t_cornerName</i>	Corner name.

Value Returned

<i>x_disabledcorner</i>	Handle to a disabled corner.
<i>nil</i>	The command could not be run.

Examples

The following example gets the list of all the corners in the setup database. It then disables corner C2 for the opamplib:ampTest:2 test:

```
s1 = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( s1 )  
=>1001  
axlGetCorners(x_mainSDB)  
=> (1003  
  ("C0" "C1" "C2" "C2_0_1" "C2_0_2")  
)  
x_testHandle2 = axlGetTest( x_mainSDB "opamplib:ampTest:2")  
=>2028  
axlPutDisabledCorner(x_testHandle2 "C2")  
=>2186
```


Virtuoso ADE SKILL Reference

Corners Functions

The following example adds a new corner, testC2, and disables it for the opamplib:ampTest:1 test:

```
s1 = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( s1 )  
=>1001  
x_testHandle = axlGetTest( x_mainSDB "opamplib:ampTest:1")  
=>1015  
axlPutDisabledCorner(1015 "testC2")  
=>2186
```

Related Topics

axlSetDefaultCornerEnabled

```
axlSetDefaultCornerEnabled(  
    x_testHandle  
    g_enable  
)  
=> 1 / 0
```

Description

Enables or disables the default (nominal) corner for the specified test. The program creates a nominal corner when you create a test. This corner represents the absence of corner-specific information.

Arguments

<i>x_testHandle</i>	Test handle.
<i>g_enable</i>	Enable flag. Valid Values are: ■ <code>nil</code>: Disables the default corner for the test. ■ any other value: Enables the default corner for the test.

Value Returned

1	Successful operation.
0	The command was unsuccessful.

Examples

The following example shows how to disable the nominal corner for a specific test:

```
s1 = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( s1 )  
=>1001  
testHandle = axlGetTest( x_mainSDB "test1" )  
=>1005  
axlSetDefaultCornerEnabled(1005 nil)  
=>1
```

You can use the `axlSetDefaultCornerEnabled` function to enable or disable the nominal corner for a test in a SKILL trigger code.

Virtuoso ADE SKILL Reference

Corners Functions

Related Topics

[axlCreateSession](#)

[axlGetTest](#)

[axlSetMainSetupDB](#)

axlSetAllCornersEnabled

```
axlSetAllCornersEnabled(  
    x_mainSDB  
    g_enable  
)  
=> t / nil
```

Description

Enables or disables all the corners for simulation. This changes the selection status of the *Corners* check box in the Run Summary assistant.

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>g_enable</i>	Specifies whether to enable or disable the corners. 1: Enables all corners. 0: Disables all corners.

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Examples

The following example disables all the corners in the current session.

```
session = axlGetWindowSession()  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
axlSetAllCornersEnabled(x_mainSDB nil)  
=> t
```

axlSetCornerName

```
axlSetCornerName(  
    x_cornerHandle  
    t_cornerName  
)  
=> t / nil
```

Description

Sets or updates the name of the given corner.

Arguments

<i>x_cornerHandle</i>	Handle to the corner for which the name is to be changed.
<i>t_cornerName</i>	New name to be set for the corner.

Value Returned

t	The corner name was set or updated.
nil	The command could not be run.

Examples

The following example shows how to update the name for a corner.

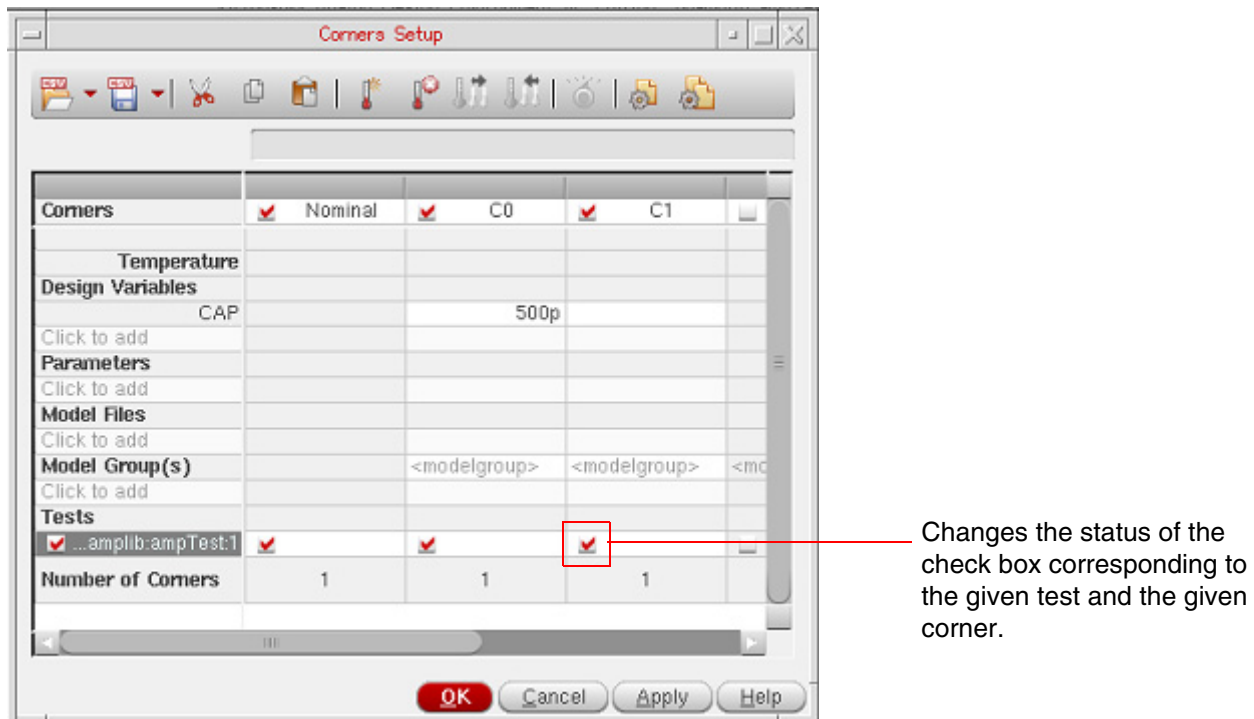
```
Ssession = axlGetWindowSession()  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
  
;; list pre-existing corners  
cadr(axlGetCorners(x_mainSDB))  
=> ("C0" "C1")  
  
;; retrieve the SDB handle for a specific corner C0  
cornerHandle = axlGetCorner(x_mainSDB "C0")  
=> 1234    ;; a non-zero integral handle  
  
;; update the corner name  
axlSetCornerName(cornerHandle "newName")  
=> t    ;; successful modification  
;; retrieve corners again to validate the name change  
cadr(axlGetCorners(x_mainSDB))  
=> ("newName" "C1")
```

axlSetCornerTestEnabled

```
axlSetCornerTestEnabled(  
    x_cornerHandle  
    t_testName  
    g_enableFlag  
)  
=> x_num / nil
```

Description

Enables or disables simulation of a test for the given corner. This function changes the status of the check box corresponding to a test in the column for the given corner.



Virtuoso ADE SKILL Reference

Corners Functions

Arguments

<i>x_cornerHandle</i>	Handle to the corner
<i>t_testName</i>	Name of the test
<i>g_enableFlag</i>	Enable or disable status

Value Returned

<i>x_num</i>	A handle to the corner.
<i>nil</i>	The command could not be run.

Examples

The following example shows how to enable the test AC for corner C0.

```
session = axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
x_cornerHandle = axlGetCorner(x_mainSDB "C0")
=> 3573
axlSetCornerTestEnabled(x_cornerHandle "AC" t)
=> 3979
```

axlSetNominalCornerEnabled

```
axlSetNominalCornerEnabled(  
    x_mainSDB  
    g_enable  
)  
=> t / nil
```

Description

Enables or disables the nominal corner in the specified setup database.

Arguments

<i>x_mainSDB</i>	Setup database handle.
<i>g_enable</i>	Specifies if the nominal corner is to be enabled or disabled. Set the value as 1 to enable the nominal corner, otherwise set it as 0.

Value Returned

t	The nominal corner value is set as specified.
nil	The command is not successful.

Examples

The following example enables the nominal corner in the current session.

```
session = axlGetWindowSession()  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
axlSetNominalCornerEnabled(x_mainSDB 1)  
=>t
```


axlSetNominalCornerTestEnabled

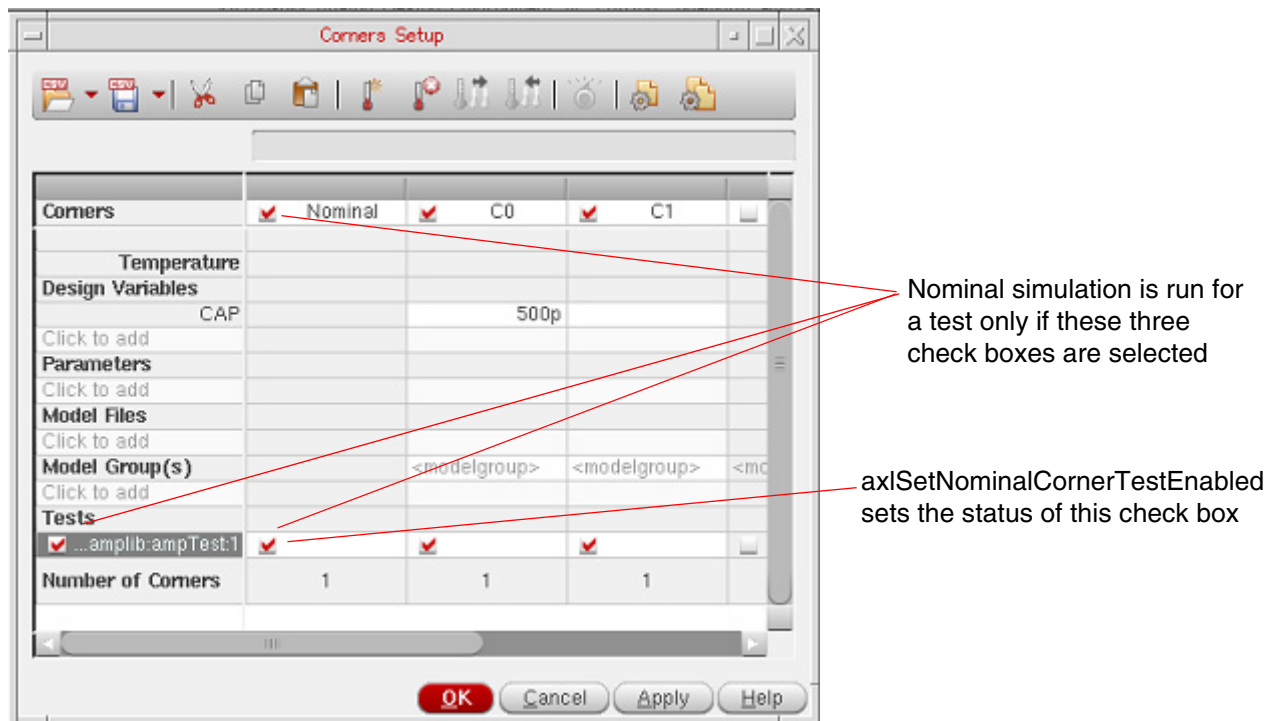
```
axlSetNominalCornerTestEnabled(  
    x_testHandle  
    g_enableFlag  
)  
=> t / nil
```

Description

Sets the status of the check box corresponding to the given test in the *Nominal* column on the Corners Setup form.

Nominal simulation for a test is run only if the following three conditions are met:

- The check box before the column heading in the *Nominal* column on the Corners Setup form is selected
- The check box before the test name in the row header for the tests is selected
- The check box in the cell corresponding to the given test in the *Nominal* corner is enabled



Virtuoso ADE SKILL Reference

Corners Functions

Argument

<i>t_testHandle</i>	Handle to the test for which you need to specify if the nominal simulation would be run
<i>g_enableFlag</i>	t or nil status to be set for the check box corresponding to the given test

Value Returned

t	Status of the check box is set.
nil	The command is unsuccessful.

Examples

The following example selects the status of the nominal corner.

```
session = axlGetWindowSession()
=>"session0"
x_mainSDB=axlGetMainSetupDB( session )
=>1001
x_testHandle = axlGetTest(x_mainSDB "AC")
axlSetNominalCornerTestEnabled( x_testHandle t)
=> t
```

Related Topics

[axlSetNominalCornerEnabled](#)

[axlSetEnabled](#)

axlSetWCCTime

```
axlSetWCCTime(  
    x_specID  
    t_time  
)  
=> t / nil
```

Description

Sets the time information for the given specification handle of a worst-case corner.

This function does not require any particular format for specifying the input time and also no validity checks are required for the time string.

Arguments

<i>x_specID</i>	Handle to the specification of the worst-case corner.
<i>t_time</i>	Time information to be set.

Value Returned

<i>t</i>	The time information is set.
<i>nil</i>	The command is unsuccessful.

Examples

Sets the time information of the spec (specification handle) to "Dec 3 20:52:47 2022".

Here, 1054 is the worst-case corner sdb handle.

```
speech = axlGetWCCSpec(1054, "Gain")  
axlSetWCCTime( speech, "Dec 3 20:52:47 2022")  
=> t
```

Related Topics

Worst Case Corners

axlGetWCCCorner

```
axlGetWCCCorner(  
    x_specHandle  
)  
=> t_value / nil
```

Description

Gets the corner name for the given specification handle of a worst-case corner.

Arguments

<i>x_specHandle</i>	Setup database handle to the specification of a worst-case corner.

Value Returned

<i>t_value</i>	Name of the corner.
<i>nil</i>	No worst-case corner is found.

Examples

Returns the corner name for the given handle.

```
corner1 = axlGetWCCCorner(1005)  
=> "_default"
```

Related Topics

[Worst Case Corners](#)

axlGetWCCHistory

```
axlGetWCCHistory(  
    x_specHandle  
)  
=> t_historyName / nil
```

Description

Returns name of the history for a specification of a worst-case corner.

Arguments

<i>x_specHandle</i>	Database handle to the specification of a worst-case corner.

Value Returned

<i>t_historyName</i>	Name of the history.
<i>nil</i>	The command was not successful.

Examples

Returns the name of the history for corner 1005.

```
t_history = axlGetWCCHistory(1005)  
=> "History.1"
```

Related Topics

[Worst Case Corners](#)

axlGetWCCResult

```
axlGetWCCResult(  
    x_specHandle  
)  
=> t_result / nil
```

Description

Returns result of a specification of a worst-case corner.

Arguments

<i>x_specHandle</i>	Database handle to the specification of a worst-case corner.

Value Returned

<i>t_result</i>	Result of the specification.
<i>nil</i>	The command was not successful.

Examples

Returns handle to specification 1005.

```
axlGetWCCResult(1005)  
=> "avg_vt"
```

Related Topics

[Worst Case Corners](#)

axlGetWCCSpec

```
axlGetWCCSpec(  
    x_cornerHandle  
    t_specName  
)  
=> x_spec / nil
```

Description

Returns handle to a specification for the specified worst-case corner.

Arguments

<i>x_cornerHandle</i>	Handle to a worst-case corner for which you want to get specification.
<i>t_specName</i>	Name of the spec for which you want to get the handle.

Value Returned

<i>x_spec</i>	Handle to a specification.
<i>nil</i>	The command was unsuccessful.

Examples

Returns the handle to the specification `test1.result1`.

```
axlGetWCCSpec(1005 "test1.result1")  
=> 1987
```

Related Topics

[Worst Case Corners](#)

axlGetWCCSpecs

```
axlGetWCCSpecs(  
    x_wccHandle  
)  
=> l_specs / nil
```

Description

Returns a list of specifications for the given worst-case corner.

Arguments

<i>x_wccHandle</i>	Handle to the worst-case corner.

Value Returned

<i>l_specs</i>	A list of specifications for the given worst-case corner.
<i>nil</i>	The command was not successful.

Examples

Returns a list of specifications for the given worst-case corner.

```
l_specs = axlGetWCCSpecs(1005)  
=> (1940 ("solutions:ampTest:1.avg_vt" "solutions:ampTest:2.avg_vt1"))
```

Related Topics

[Worst Case Corners](#)

axlGetWCCTest

```
axlGetWCCTest(  
    x_wccHandle  
)  
=> t_testName / nil
```

Description

Returns the name of the test of the specification associated with a worst-case corner.

Arguments

<i>x_wccHandle</i>	Handle to the spec of a worst-case corner.

Value Returned

<i>t_testName</i>	Name of the test of the given specification.
<i>nil</i>	The command was not successful.

Examples

The following example prints the test and spec names associated with the worst-case corners in the setup.

```
corners = (cadr (axlGetCorners sdb))  
  
(foreach corner corners  
    cornerh = (axlGetCorner sdb corner)  
    specs = (axlGetWCCSpecs cornerh)  
    (if (cadr specs) then  
        (printf "Corner: %s\n" corner)  
    else  
        (if (cadr specs) then  
            (printf "Corner: %s\n" corner)  
            (printf "\tno specs\n")  
        )  
    )  
  
(foreach spec (cadr specs)  
    spec = (axlGetWCCSpec cornerh spec)  
    testname = (axlGetWCCTest spec)  
    specname = (axlGetWCCResult spec)  
    wcccorner = (axlGetWCCCorner spec)  
    (printf "\tTest = %s\n" testname)  
    (printf "\tSpec name = %L\n" specname)  
    (printf "\tWCC corner = %L\n\n" wcccorner)  
    )
```

Virtuoso ADE SKILL Reference

Corners Functions

)

Related Topics

Worst Case Corners

axlGetWCCTime

```
axlGetWCCTime(  
    x_specId  
)  
=> t_time / nil
```

Description

Returns the generated time information for the given specification handle of a worst-case corner.

Arguments

<i>x_specId</i>	Handle to the specification of the worst-case corner.

Value Returned

<i>t_time</i>	The time string value.
<i>nil</i>	The command was unsuccessful.

Examples

The following example prints the test name, spec name, and the time stamp associated with the worst-case corners in the setup.

```
corners = (cadr (axlGetCorners sdb))  
  
(foreach corner corners  
  cornerh = (axlGetCorner sdb corner)  
  specs = (axlGetWCCSpecs cornerh)  
  (if (cadr specs) then  
    (printf "Corner: %s\n" corner)  
  else  
    (if (cadr specs) then  
      (printf "Corner: %s\n" corner)  
      (printf "\tno specs\n")  
    )  
)  
  
(foreach spec (cadr specs)  
  spec = (axlGetWCCSpec cornerh spec)  
  testname = (axlGetWCCTest spec)  
  specname = (axlGetWCCResult spec)  
  wcctime = (axlGetWCCTime spec)  
  wcccorner = (axlGetWCCCorner spec)  
  (printf "\tTest = %s\n" testname)  
  (printf "\tSpec name = %L\n" specname)
```

Virtuoso ADE SKILL Reference

Corners Functions

```
        (printf "\tWCC corner = %L\n\n" wcccorner)
        (printf "\tWCC time= %L\n\n" wcctime)
    )
)
```

Related Topics

Worst Case Corners

axlGetWCCRangeBound

```
axlGetWCCRangeBound(  
    x_hbdb  
)  
=> t_rangeBound
```

Description

Returns an integer value that specifies whether the worst-case corner corresponds to the minimum or the maximum value of the spec.

Arguments

<i>x_hbdb</i>	Handle to the worst-case corner.

Value Returned

0	A corner is created for the lower boundary of the spec range.
1	A corner is created for the upper boundary of the spec range.

Examples

Returns an integer value that signifies the boundary along which the corner is created.

```
corner = axlGetCorner(1001 "WCC_C2")  
=> 1934  
axlGetWCCRangeBound(1934)  
=> 1
```

Related Topics

[Worst Case Corners](#)

axlGetWCCVar

```
axlGetWCCVar(  
    x_hscr  
    t_name  
)  
=> x_handle / nil
```

Description

Finds the specified variable by name and returns a handle to it.

Arguments

<i>x_hscr</i>	Handle to the worst-case corner.
<i>t_name</i>	Name of the variable for which you want to get the handle.

Value Returned

<i>x_handle</i>	The handle to the variable.
<i>nil</i>	The command was not successful.

Examples

Returns a handle to the variable CAP.

```
x_handleToVar = axlGetWCCVar(1005 "CAP")  
=> 1005
```

Related Topics

[Worst Case Corners](#)

axlGetWCCVarMonotonicity

```
axlGetWCCVarMonotonicity(  
    x_hsdb  
)  
=> t_monotonicity / nil
```

Description

Gets the monotonicity of a specific variable or parameter of the worst-case corner.

Arguments

<i>x_hsdb</i>	Setup database handle to the variable or parameter for which you want to get the monotonicity.

Value Returned

<i>t_monotonicity</i>	A monotonicity value.
<i>nil</i>	The command was not successful.

Examples

Returns the monotonicity value of the specified variable or parameter.

```
m_var1 = axlGetWCCVarMonotonicity(1005)  
=> +1
```

Related Topics

[Worst Case Corners](#)

axlGetWCCVars

```
axlGetWCCVars (  
    x_hsdb  
)  
=> l_vars / nil
```

Description

Returns a list containing a handle to all variables and a list of all variable names in a worst-case corner.

Arguments

<i>x_hsdb</i>	Setup database handle.

Value Returned

<i>l_vars</i>	A list of variables for the given worst-case corner handle.
<i>nil</i>	The command was not successful.

Examples

The following example prints the test, spec names, and variables associated with the worst-case corners in the setup.

```
corners = (cadr (axlGetCorners sdb))  
  
(foreach corner corners  
    cornerh = (axlGetCorner sdb corner)  
    specs = (axlGetWCCSpecs cornerh)  
    (if (cadr specs) then  
        (printf "Corner: %s\n" corner)  
    else  
        (if (cadr specs) then  
            (printf "Corner: %s\n" corner)  
            (printf "\tno specs\n")  
        )  
    )  
  
(foreach spec (cadr specs)  
    spec = (axlGetWCCSpec cornerh spec)  
    testname = (axlGetWCCTest spec)  
    specname = (axlGetWCCResult spec)  
    wcctime = (axlGetWCCTime spec)  
    wcccorner = (axlGetWCCCorner spec)  
    (printf "\tTest = %s\n" testname)  
    (printf "\tSpec name = %L\n" specname)
```


Virtuoso ADE SKILL Reference

Corners Functions

```
(printf "\tWCC corner = %L\n\n" wcccorner)
(printf "\tWCC time= %L\n\n" wcctime)
(printf "\t WCC variables = %L \n\n" axlGetWCCVars(specname)
)
)
```

Related Topics

[Worst Case Corners](#)

Virtuoso ADE SKILL Reference

Corners Functions

Optimization Functions

This topic lists the SKILL functions that let you optimize the Improve Yield flow in Virtuoso Variation Option.

[axlGetWYCSigmaTargetLimit](#)

[axlSetWYCSigmaTargetLimit](#)

axlGetWYCSigmaTargetLimit

```
axlGetWYCSigmaTargetLimit(  
    )  
=> n_sigma_limit
```

Description

Gets the sigma-to-target limit for the Improve Yield flow using worst yield corners. If this value is not set, the flow internally sets it to 100.

Arguments

None

Value Returned

<i>n_sigma_limit</i>	The sigma-to-target limit for the Improved Yield flow.
<i>nil</i>	The command was unsuccessful.

Examples

```
axlGetWYCSigmaTargetLimit()  
=> 54
```

Related Topics

[axlSetWYCSigmaTargetLimit](#)

[Yield Improvement](#)

axlSetWYCSigmaTargetLimit

```
axlSetWYCSigmaTargetLimit(  
    n_sigma_limit  
)  
=> t / nil
```

Description

Sets the sigma-to-target limit for Improve Yield flow using worst yield corners.

Arguments

<i>n_sigma_limit</i>	The sigma-to-target limit.

Value Returned

t	The sigma-to-target limit for Improve Yield flow is set.
nil	The sigma-to-target limit for Improve Yield flow cannot be set.

Examples

```
axlSetWYCSigmaTargetLimit(120)  
=> t
```

Related Topics

[axlGetWYCSigmaTargetLimit](#)

[Yield Improvement](#)

Run-Related Functions

This topic lists the SKILL functions that let you manage simulations runs in ADE Explorer and ADE Assembler.

[maeGetSimulationMessages](#)

[maeGetMappingForJobAndPoint](#)

[maeOpenLogViewer](#)

[maeResumeSimulation](#)

[maeRunSimulation](#)

[maeStopSimulation](#)

[maeSuspendSimulation](#)

[maeSetPreRunScript](#)

[maeSetRunOption](#)

[maeWaitUntilDone](#)

[maeWriteScript](#)

[axlExportOutputView](#)

[axlGetAllSweepsEnabled](#)

[axlGetCurrentRunMode](#)

[axlGetParasiticRunMode](#)

[axlGetParasiticParaLCV](#)

[axlGetParasiticSchLCV](#)

Virtuoso ADE SKILL Reference

Run-Related Functions

[axlGetPreRunScript](#)

[axlGetRunDistributeOptions](#)

[axlGetRunData](#)

[axlGetRunMode](#)

[axlGetRunModes](#)

[axlGetRunOption](#)

[axlGetRunOptionName](#)

[axlGetRunOptions](#)

[axlGetRunOptionValue](#)

[axlGetRunStatus](#)

[axlPutRunOption](#)

[axlIsSimUsingStatParams](#)

[axlRunAllTests](#)

[axlRunAllTestsWithCallback](#)

[axlRunSimulation](#)

[axlSetCurrentRunMode](#)

[axlImportPreRunScript](#)

[axlSetParasiticRunMode](#)

[axlSetPreRunScript](#)

[axlSetPreRunScriptEnabled](#)

[axlSetRunDistributeOptions](#)

[axlSetRunOptionName](#)

[axlStop](#)

[axlStopAll](#)

Virtuoso ADE SKILL Reference

Run-Related Functions

axlReadHistoryResDB

axlReadResDB

axlSetRunOptionValue

maeGetSimulationMessages

```
maeGetSimulationMessages(  
    [ ?session t_session ]  
    [ ?msgType t_messageType ]  
)  
=> t_messages / nil
```

Description

Displays an error message of the given type during the simulation run.

Arguments

?session t_session

Name of the session.

?msgType t_messageType

Type of messages to be displayed.

The valid values are:

- ERROR. This is the default value.
- WARNING
- INFO
- ALL

Value Returned

t_messages

Returns a string of errors for the given simulation.

nil

The command was unsuccessful.

Examples

The following example shows how you can print all warning messages from a simulation run:

```
mySession=maeOpenSetup("solutions" "amptest" "maestro")  
=> t  
>  
maeRunSimulation()  
=> "Interactive.1"
```

Virtuoso ADE SKILL Reference

Run-Related Functions

```
>
maeWaitUntilDone('all')
>
maeGetSimulationMessages(?session mySession ?msgType "ERROR")
=>"ERROR (ADEXL-1611): Sweep and Corners are using the same variables:vddEither
disable variables in sweeps, or disable corners that use these variables.\nERROR
(ADEXL-1742): Cannot run 'Monte Carlo Sampling' with swept variable/parameters.You
are trying to run Monte Carlo with sweeps enabled. This is currently not supported.
Disable all sweeps or enable 'Use Reference Point' in the Monte Carlo options to
continue.\n"
>
```

maeGetMappingForJobAndPoint

```
maeGetMappingForJobAndPoint(  
    [ ?session t_session ]  
)  
=> l_jobPointMapping / nil
```

Description

Returns a list containing the mapping of job IDs to the point IDs allocated to them for all the simulations run in the current ADE Explorer or ADE Assembler session. After starting a simulation run, you can use this information to debug incomplete simulations.

Arguments

<code>?session t_session</code>	Name of the session.

Value Returned

<code>l_jobPointMapping</code>	Returns a list of job ID and point ID mapping.
<code>nil</code>	The command was unsuccessful.

Examples

The following example shows the result of a simulation run on an LSF farm. The maximum jobs value in the job policy is 4.

```
maeGetMappingForJobAndPoint()  
=> ((1 "1 ") (2 "2 3 ") (4 "2 ") )
```

maeOpenLogViewer

```
maeOpenLogViewer(  
    [ t_dbFilePath ]  
)  
=> t / nil
```

Description

Opens the Log Viewer window, where you can view the messages loaded from a database.

Arguments

<i>t_dbFilePath</i>	Path to the database file in which messages are saved.

Value Returned

<i>t</i>	The Log Viewer window was opened.
<i>nil</i>	The command was unsuccessful.

Examples

The following example code opens the Log Viewer window and loads the given database:

```
maeOpenLogViewer("/home/example/Interactive.0.msg.db")  
=> t
```

maeResumeSimulation

```
maeResumeSimulation(  
    [ t_session ]  
)  
=> t / nil
```

Description

Resumes the simulations that were earlier suspended automatically by ADE Explorer or ADE Assembler according to the settings specified on the *Resources* tab of the Job Policy Setup form.

Arguments

<i>t_session</i>	Name of the ADE Explorer or ADE Assembler session. If not specified, the current session is used.

Value Returned

<i>t</i>	The simulation was resumed.
<i>nil</i>	The command was not successful.

Examples

The following example shows how you can print all warning messages from a simulation run:

```
mySession=maeOpenSetup("solutions" "amptest" "maestro")  
=> t  
>  
maeRunSimulation()  
=> "Interactive.1"  
...  
;; assumption that the simulation got suspended due to unavailability of disk space  
maeResumeSimulation()  
=> t
```

Related Topics

[Automatically Suspend Simulations When the Disk Space is Low](#)

maeRunSimulation

```
maeRunSimulation(  
    [ ?session t_sessionName ]  
    [ ?runMode t_runMode ]  
    [ ?callback t_callback ]  
    [ ?run t_runPlan ]  
    [ ?waitUntilDone g_waitUntilDone ]  
    [ ?returnRunId g_returnRunId ]  
)  
=> t_histname / x_runID / nil
```

Description

Sets the given run mode for the given session and runs simulation.

Arguments

?session t_sessionName

Name of the ADE Explorer or ADE Assembler session.

Default: Current session

?runMode t_runMode

Virtuoso ADE SKILL Reference

Run-Related Functions

Name of the run mode to be run.

The valid values are:

- "Single Run, Sweeps and Corners"
- "Monte Carlo Sampling"
- "Global Optimization"
- "Local Optimization"
- "Improve Yield"
- "Sensitivity Analysis"
- "Conjugate Gradient Optimization"
- "Feasibility Analysis"
- "Create Worst Case Corners"
- "Manual Optimization"
- "Manual Tuning"
- "Run Plan"
- "Fault Simulation"
- "Size Over Corners"

Default: "Single Run, Sweeps and Corners"

For ADE Explorer, you can set this argument to only "Single Run, Sweeps and Corners" or "Monte Carlo Sampling".

`?callback t_callback`

A callback procedure to be executed after completing the simulation.

`?run t_runPlan`

Name of the run plan. Use this argument in the scripts you save to run the plans created in the Run Plan assistant in ADE Assembler.

To run simulation for all the runs in the run plan, set this argument to "All".

`?waitUntilDone g_waitUntilDone`

Virtuoso ADE SKILL Reference

Run-Related Functions

Boolean value that determines if the tool should wait for this run to complete before executing the next command in the script.

Valid values are:

- `t`: Specifies that the tool should wait for the completion of this run.
- `nil`: Specifies that the tool should not wait for the completion of this run. You can use this option when you intend to run multiple runs in parallel. This is the default value.

`?returnRunId g_returnRunId`

Boolean value that determines if the tool should return the run ID instead of the default return value, history name.

Valid values:

- `t`: Returns the run ID of the simulation run. You can use this run ID as an argument in other SKILL functions that require run ID instead of history name.
- `nil`: Returns the history name of the simulation run. This is the default value.

Value Returned

`t_histName`

Name of the history created after the simulation run.

`x_runID`

Run ID of the simulation run

This value is returned only when the `?returnRunId` argument is set to `nil`.

`nil`

The command was not successful.

Examples

The following example code runs simulation in Single Run, Sweeps, and Corners modes for the current session:

```
maeOpenSetup("solutions" "amptest" "maestro" ?mode "a")
=> "session1"
maeRunSimulation()
=> "Interactive.1"
;; returns the name of the history created by the run
```


Virtuoso ADE SKILL Reference

Run-Related Functions

The following example code runs Monte Carlo Sampling for the current session and executes the code given with the `?callback` argument after completing the run:

```
maeRunSimulation(?runMode "Monte Carlo Sampling" ?callback "printf(\"Run Complete\\")")
=> "MonteCarlo.1"
```

The following example code runs a simulation for the current session and executes a callback procedure after completion:

```
;; Loading setup
maeOpenSetup("testLib" "demo_top" "maestro" ?mode "a")

;; Post-simulation callback. Called after each run.
define( RunFinishedCallback(session runID)
printf("Run ID %L has finished" runID)
)
; Execute simulation
maeRunSimulation(?callback "RunFinishedCallback")
```

The following example code shows how you can run simulations for multiple sessions together:

```
;; Loading setup
session1 = maeOpenSetup("testLib" "demo_top" "maestro" ?mode "a")
session2 = maeOpenSetup("testLib" "block1" "maestro" ?mode "a")

;; Post-simulation callback. Called after each run.
define( RunFinishedCallback(session runID)
printf("Run ID %L has finished" runID)
)
; Run simulations
history1 = maeRunSimulation(?session session1 ?callback "RunFinishedCallback")
history2 = maeRunSimulation(?session session2 ?callback "RunFinishedCallback")

maeWaitUntilDone('All)

;;Generate unique file names where you want to save results

file1=strcat(history1 "_" axlGetSessionLibName(session1) "_"
axlGetSessionCellName(session1) "_" axlGetSessionViewName(session1))

file2=strcat(history2 "_" axlGetSessionLibName(session2) "_"
axlGetSessionCellName(session2) "_" axlGetSessionViewName(session2))

; file names are now available in file1 and file2 variables

;; Export Results to CSV

maeExportOutputView(?fileName file1 ?session session1 ?historyName history1)
maeExportOutputView(?fileName file2 ?session session2 ?historyName history2)

exit()
```

The following example code shows how to check the simulation status for each point:

```
maeRunSimulation()
x_history=axlGetHistoryEntry(x_mainSDB "Interactive.3")
```

Virtuoso ADE SKILL Reference

Run-Related Functions

```
rdbPath=axlGetHistoryResults(x_history)
r=axlOpenResDB(rdbPath)
r->testStatus("myTest1_1" 1) ; myTest1 is the test name, 1 is the point ID
; returns an integer indicating the simulation status of a point
; 1 - Pending
; 2- Running
; 3- Done
;10 -Disabled status
```

maeStopSimulation

```
maeStopSimulation(  
    [ ?historyName l_historyNames ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Stops simulation runs for the given histories.

Arguments

?historyName l_historyNames

List of history names.

If this argument is not provided, all the simulations that are currently running are stopped.

?session t_sessionName

Name of the session.

If not specified, the currently active session is used.

Value Returned

t Simulation was stopped for the given histories.

nil The command was not successful.

Examples

The following example code runs the Single Run, Sweeps and Corners for the current session:

```
maeOpenSetup("solutions" "amptest" "maestro" ?mode "a")  
=> "session1"  
maeRunSimulation()  
=> "Interactive.1"  
;; returns the name of the history created by the run  
maeStopSimulation()
```

maeSuspendSimulation

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

Description

Suspends the simulation run for the specified session. If not specified, the current simulation run is suspended.

Arguments

<i>t_sessionName</i>	Name of the ADE Explorer or ADE Assembler session. Default: Current session.

Value Returned

<i>t</i>	The simulation run was suspended.
<i>nil</i>	The command could not be run.

Examples

The following example shows how to suspend a simulation run:

```
maeRunSimulation()  
=> "Interactive.9"  
maeSuspendSimulation()  
=> t
```

maeSetPreRunScript

```
maeSetPreRunScript(  
    t_scriptName  
    [ ?tests l_testNames ]  
    [ ?enabled g_enabled ]  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets the given script as a pre-run script for the given test and sets its status.

Arguments

<i>t_scriptName</i>	Name of the script.
<i>?tests l_testNames</i>	List of the tests for which the script needs to be enabled.
<i>?enabled g_enabled</i>	Status of the script. The valid values are: ■ <i>t</i>: Enabled ■ <i>nil</i>: Disabled Default value: <i>t</i>
<i>?session t_sessionName</i>	Name of the session. If not specified, the currently active session is used.

Virtuoso ADE SKILL Reference

Run-Related Functions

Value Returned

<code>t</code>	The given script was set as the pre-run script.
<code>nil</code>	The command was not successful.

Examples

Sets `preRun.il` as the pre-run script for all tests and marks it enabled for all tests.

```
maeSetPreRunScript("preRun.il")  
=> t
```

Sets `preRun.il` as the pre run script for tests `test1` and `test2`.

```
maeSetPreRunScript("preRun.il" ?tests '("test1" "test2"))  
=> t
```

maeSetRunOption

```
maeSetRunOption(  
    t_mode  
    t_runOptionName  
    t_runOptionValue  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Sets value for a run option.

Arguments

<i>t_mode</i>	Name of the run mode. Valid values are: "Sampling", "Global Optimization", "Local Optimization", "Monte Carlo Sampling"

Virtuoso ADE SKILL Reference

Run-Related Functions

<i>t_runOptionName</i>	Name of the run option. The run option should be relevant for the run mode specified in the <i>t_mode</i> argument. Possible value when <i>t_mode</i> is set to "Sampling": ■ <i>points</i>: Number of sampling points Possible values when <i>t_mode</i> is set to "Global Optimization": ■ <i>tillsatisfied</i>: Optimization stops when all goals are met ■ <i>timelimit</i>: Optimization stops when the program reaches the time limit (in minutes) ■ <i>numpoints</i>: Optimization stops when the program reaches the number of points ■ <i>ptswithnoimprovement</i>: Optimization stops when there is no improvement for the number of points Possible values when <i>t_mode</i> is set to "Local Optimization": ■ <i>effort</i>: Optimization effort ■ <i>tillsatisfied</i>: Optimization stops when all goals are met ■ <i>timelimit</i>: Optimization stops when the program reaches the time limit (in minutes) ■ <i>numpoints</i>: Optimization stops when the program reaches the number of points ■ <i>ptswithnoimprovement</i>: Optimization stops when there is no improvement for the number of points. Possible values when <i>t_mode</i> is set to "Monte Carlo Sampling": ■ <i>mcmethod</i>: Monte Carlo Sampling method ■ <i>mcnumpoints</i>: Number of Monte Carlo sampling points Note: Typically, this number should be at least the number of statistical variables.
<i>t_runOptionValue</i>	Value of the run option.

Virtuoso ADE SKILL Reference

Run-Related Functions

`?session t_sessionName`

Name of the session.

If not specified, the currently active session is used.

Value Returned

`t` Value for the run option was set.

`nil` The command could not be run.

Examples

Sets a value for sampling mode mode

```
maeOpenSetup("solutions" "amptest" "maestro" ?mode "a")
=> "session1"
maeSetRunOption("Sampling" "points" "10")
=> t
maeRunSimulation()
=> "Interactive.1"
```

maeWaitUntilDone

```
maeWaitUntilDone(  
    g_historyNames  
    [ ?session t_sessionName ]  
)  
=> t /nil
```

Description

Specifies the names of historic checkpoints for which the tool must wait before proceeding.

Arguments

g_historyNames Specifies one or more history checkpoints for which the tool must wait to complete simulation before proceeding to the next command.

The valid values are:

- Name of a history.
- A list of history names for which the simulation must be complete before the tool runs the next command.
- 'All to specify that the tool must wait for the completion of all active runs.

?session *t_sessionName*

Name of the ADE Assembler or Explorer session.

Default value: Current session

Value Returned

<i>t</i>	The specified history checkpoints have been set.
<i>nil</i>	The command could not be run.

Examples

Runs a simulation for the given setup. Assembler is made to wait for the completion of the simulation for multiple histories.

Virtuoso ADE SKILL Reference

Run-Related Functions

```
maeOpenSetup("solutions" "amptest" "maestro")
;; the above function loads the given cellview and returns the session object
=> "session0"
maeSetVar("rload" 10k)
;; sets value for the rload variable
=> t
maeRunSimulation()
;; the above function runs the simulation for the given setup and returns the name
of the history
=> "Interactive.2"
maeWaitUntilDone('All)
=> (0)
maeExportOutputView()
;; the above function writes the results in a csv file in the current working
;; directory and returns the file path when successful.
=> "/servers/scratch02/testcases/adexl_workshop/design/Interactive.2.csv"
```

;;To make Assembler wait for the completion of simulation for a particular history:

```
hist=maeRunSimulation()
=> "Interactive.3"
maeWaitUntilDone(hist)
```

;; To make Assembler wait for the completion of simulation for multiple histories,
;; provide a list of all history names to the maeWaitUntilDone function, as shown
;; below:

```
hist=maeRunSimulation()
=> "Interactive.3"
```

```
hist1=maeRunSimulation()
=> "Interactive.4"
```

```
hist2=maeRunSimulation()
=> "Interactive.5"
```

```
maeWaitUntilDone(list(hist hist1))
maeExportOutputView()
=> "/servers/scratch02/testcases/adexl_workshop/design/Interactive.4.csv"
```

maeWriteScript

```
maeWriteScript(  
    t_fileName  
    [ ?session t_session ]  
    [ ?shouldRunActive g_shouldRunActive ]  
    [ ?runPlans l_runPlans ]  
    [ ?histories l_histories ]  
)  
=> t / nil
```

Description

Creates a script with the specified setup details. This script can be run from the command line.

Arguments

<i>t_fileName</i>	Name of the script file to be created
<i>?session t_session</i>	Name of the session from which setup details are to be written to the script By default, the current session and its active state is used.tam
<i>?shouldRunActive g_shouldRunActive</i>	Boolean value that specifies if the active setup should be run or not. When this argument is set to t, <code>maeRunSimulation()</code> is added to the script saved by <code>maeWriteScript</code>. Default value: t
<i>?runPlans l_runPlans</i>	List containing the names of run plans to be added to the script. By default, all the run plans defined in the active setup are saved in the script.
<i>?histories l_histories</i>	List of histories.

Virtuoso ADE SKILL Reference

Run-Related Functions

Value Returned

t	Successful operation.
nil	The command could not be run. An error message is also displayed in the log.

Examples

The following example code creates a script for the current cellview with the given file name:

```
maeOpenSetup("solutions" "amptest" "maestro")  
;; the above function loads the given cellview and returns the session object  
=> "session0"  
maeWriteScript("runPlan.il")  
=> t
```

Related Topics

[Executing Runs Using a Script](#)

axlExportOutputView

```
axlExportOutputView(  
    t_sessionName  
    t_fileName  
    t_viewType  
    [ ?history g_historyName ]  
    [ ?testName g_testName ]  
    [ ?filterName t_filterName ]  
    [ ?clearAllFilters g_clearAllFilters]  
)  
=> t / nil
```

Description

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

Arguments

<i>t_sessionName</i>	Name of the ADE Assembler or ADE Explorer session. Default value: current session
<i>t_fileName</i>	Path and name of the file to which results are to be exported.
<i>t_viewType</i>	Name of the output view to be exported. Valid values are: ■ "Detail" ■ "Detail - Transpose" ■ "Status" ■ "Summary" ■ "Yield" ■ "Checks/Asserts" ■ "Fault" ■ "Current" Default value: "Current "

?history g_historyName

Virtuoso ADE SKILL Reference

Run-Related Functions

Name of the history for which outputs are to be exported.

Default value: " "

?testName *g_testName*

Name of the test for which outputs are to be exported. This argument is useful when you are exporting results from a multi-test cellview.

This argument is supported only for the `Checks/Asserts` and `Fault` result views.

?filterName *t_filterName*

Name of a saved filter to be applied while exporting the results.

?clearAllFilters *g_clearAllFilters*

A Boolean value that specifies whether or not to clear the filters before exporting outputs.

Value Returned

<code>t</code>	The output view is successfully exported to the specified file.
<code>nil</code>	The command could not be run.

Examples

The following examples export the results view to the specified `.csv` and `.html` files.

```
axlExportOutputView(axlGetWindowSession() "./abc.csv" "Yield")
=> t
axlExportOutputView(axlGetWindowSession() "./abc.html" "Detail-Transpose")
=> t
axlExportOutputView(axlGetWindowSession() "./abcd.html" "Yield" ?history
"Interactive.20")
=> t
```

Related Topics

[Exporting Outputs to a CSV File](#)

axlGetAllSweepsEnabled

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

Description

Returns the selection status of the *Sweep* check box in the Run Summary assistant.

Argument

<i>x_hsdb</i>	Setup database handle.

Value Returned

<i>t</i>	The <i>Sweep</i> check box is enabled.
<i>nil</i>	The <i>Sweep</i> check box is disabled.

Examples

The following example checks for the status of the *Sweep* check box in the given setup database.

```
axlGetAllSweepsEnabled 1003  
=> t
```

Related Topics

[Run Summary Assistant](#)

axlGetCurrentRunMode

```
axlGetCurrentRunMode(  
    x_hsdb  
)  
=> t_mode / nil
```

Description

Returns the current simulation run mode of a given session.

Argument

<i>x_hsdb</i>	Setup database handle.

Value Returned

<i>t_mode</i>	The current simulation run mode, which could be one of the following values: ■ Single Run, Sweeps and Corners ■ Monte Carlo Sampling ■ Global Optimization ■ Local Optimization ■ Improve Yield ■ Sensitivity Analysis
<i>nil</i>	The command could not be run.

Examples

The following example returns the run mode set in the current session:

```
sdb = axlGetMainSetupDB(axlGetWindowSession())  
runMode = axlGetCurrentRunMode(sdb)  
=> "Single Run, Sweeps and Corners"
```

The following example finds the run mode of the given history name:

```
(defun CCRaxlGetRunModeFromHistoryName (sdbh histName)  
  (let (checkPoint)  
    checkPoint = (axlGetHistoryCheckpoint (axlGetHistoryEntry sdbh histName))
```

Virtuoso ADE SKILL Reference

Run-Related Functions

```
(axlGetCurrentRunMode checkPoint)
)
)
```

```
histName = "MonteCarlo.0"
sess = axlGetWindowSession()
sdbh = (axlGetMainSetupDB sess)
runMode = (CCRaxlGetRunModeFromHistoryName sdbh histName) (printf "Run mode for
history %s = \"%s\"\\n" histName runMode)
=> Run mode for history Interactive.0 = Single Run, Sweeps, and Corners
```

Related Topics

Specifying the Run Mode

axlGetParasiticRunMode

```
axlGetParasiticRunMode(  
    x_mainSDB  
)  
=> t_runMode
```

Description

Gets the parasitic run mode name from the active setup or history checkpoint.

Argument

<i>x_mainSDB</i>	Handle to the active setup database or a checkpoint history.

Value Returned

<i>t_runMode</i>	Name of the parasitic run mode.

Examples

The following example returns the name of the parasitic run mode set in the active session:

```
s1 = axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
axlGetParasiticRunMode(1001)  
=> "Extracted (Parasitics/LDE)"
```

Related Topics

[axlGetMainSetupDB](#)

[axlGetWindowSession](#)

[Parasitic Mode Toolbar](#)

axlGetParasiticParaLCV

```
axlGetParasiticParaLCV(  
    t_sessionName  
    t_paraRunMode  
)  
=> t_cellViewName
```

Description

Gets the name of the parasitic cellview attached to the parasitic run mode in the active setup or history checkpoint.

Arguments

<i>t_sessionName</i>	Name of the session or checkpoint history. Alternatively, you can provide a handle to the session or checkpoint history.
<i>t_paraRunMode</i>	Name of the parasitic run mode.

Value Returned

<i>t_cellViewName</i>	Name of the parasitic cellview.

Examples

The following example returns the name of the parasitic cellview attached to the `Extracted` parasitic run mode in the active session:

```
s1 = axlGetWindowSession()  
=> "session0"  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
axlGetParasiticRunMode(x_mainSDB)  
=> "Extracted (Parasitics/LDE)"  
axlGetParasiticParaLCV(1001 "Extracted (Parasitics/LDE)")  
=> "opAmp_test:osc:av_extracted_Ronly"
```

Related Topics

[Parasitics/LDE - Setup](#)

axlGetParasiticSchLCV

```
axlGetParasiticSchLCV(  
    t_sessionName  
    t_parasiticRunMode  
)  
=> t_cellViewName
```

Description

Gets the name of the schematic cellview attached to the parasitic run mode in the active setup or history checkpoint.

Arguments

t_sessionName Name of the session or checkpoint history. Alternatively, you can provide a handle to the session or checkpoint history.

t_parasiticRunMode
 Name of the parasitic run mode.

Value Returned

t_cellViewName Name of the schematic cellview.

Examples

The following example returns the name of the schematic cellview attached to the Extracted parasitic run mode in the active session:

```
s1 = axlGetWindowSession()  
=> "session0"  
x_mainSDB=axlGetMainSetupDB( s1 )  
=> 1001  
axlGetParasiticRunMode(x_mainSDB)  
=> "Extracted (Parasitics/LDE)"  
axlGetParasiticParaLCV(1001 "Extracted (Parasitics/LDE)")  
=> "opAmp_test:osc:av_extracted_Ronly"  
axlGetParasiticSchLCV(1001 "Extracted (Parasitics/LDE)")  
=> "opAmp_test:osc:schematic"
```

Related Topics

Parasitics/LDE - Setup

axlGetPreRunScript

```
axlGetPreRunScript(  
    t_sessionName  
    t_testName  
)  
=> t_filePath
```

Description

Returns the path to the pre-run script file attached to the given test.

Arguments

<i>t_session</i>	Name of the session or a handle to it.
<i>t_testName</i>	Name of the test for which you want to get the path to the pre-run script.

Value Returned

<i>t_filePath</i>	Path to the script file. If no script is set for the given test, returns a blank string.

Examples

The following example returns the details of the pre-run script for the `AmpTest1` test:

```
axlsession=axlGetWindowSession( hiGetCurrentWindow() )  
"session0"  
axlGetPreRunScript(axlsession, "AmpTest1")  
=> "./myScript"
```

Related Topics

[axlSetPreRunScript](#)

[axlSetPreRunScriptEnabled](#)

[axlImportPreRunScript](#)

axlGetRunDistributeOptions

```
axlGetRunDistributeOptions(  
    x_hsdb  
)  
=> r_runOptions / nil
```

Description

Returns the current run option settings for the given setup database.

Argument

<i>x_hsdb</i>	Handle to the setup database.

Value Returned

<i>r_runOptions</i>	Struct of run options specified for the given setup database. A struct must contain the following elements: ■ RunIn: Describes how multiple simulations need to run. Valid values are <code>Parallel</code> or <code>Serial</code>. ■ DivideJobs: Describes how the ICRPs can be divided among the simulation runs. Valid values are <code>Specify</code> or <code>Equally</code>. ■ JobLimit: Describes the maximum number of jobs that can run when <code>DivideJobs</code> is set to <code>Specify</code>.
<i>nil</i>	The command could not be run.

Examples

The following example returns the run option settings for the `sdb` setup database.

```
sdb = axlGetMainSetupDB(axlGetWindowSession())  
runOpt = axlGetRunDistributeOptions(sdb)  
runOpt~>??  
=> (JobLimit 4 DivideJobs Specify RunIn Parallel  
)
```

Virtuoso ADE SKILL Reference

Run-Related Functions

Related Topics

[Specifying a Distribution Method](#)

axlGetRunData

```
axlGetRunData(  
    t_sessionName  
    x_runID  
)  
=> x_historyHandle / nil
```

Description

Returns the handle to the history obtained after running a simulation in the given session. You can use this handle to get access to the history results or to get setup details by using the history checkpoint.

Arguments

<i>t_sessionName</i>	Name of the session or a handle to it.
<i>x_runID</i>	Unique run ID obtained after running a simulation.

Value Returned

<i>x_historyHandle</i>	Handle to the history of the simulation run.
<i>nil</i>	The command could not be run.

Example

The following example shows how to get the history handle and use it to work with the results:

```
session=axlGetWindowSession()  
=> "session0"  
  
runid=axlRunSimulation(?session session)  
=> 0  
; here, 0 is the run ID  
  
x_historyHandle=axlGetRunData(session 0)  
=> 3234  
  
; you can use this history handle to get the details of history or to get results  
  
t_historyName=axlGetHistoryName(x_historyHandle)  
=> "Interactive.20"  
  
; the following function call loads the results for the history in the GUI  
  
axlViewHistoryResults(session x_historyHandle)
```

Virtuoso ADE SKILL Reference

Run-Related Functions

; the history name can be used to access the results by using SKILL code, as shown below.

```
rdbObj=axlReadHistoryResDB(t_historyName ?session session)
=>axlrdb@0x1949a418
```

; This results data object can be used to access different elements from the results database for that history.

Related Topics

[axlReadResDB](#)

axlGetRunMode

```
axlGetRunMode (
    x_hsdb
    t_mode
)
=> x_mode / nil
```

Description

Returns a handle to the named run mode in the specified setup database.

Arguments

<i>x_hsdb</i>	Setup database handle.
<i>t_mode</i>	A valid run mode name.
	You can get a list of valid run mode names using the <code>axlGetRunModes</code> function.

Value Returned

<i>x_mode</i>	Run mode handle.
<i>nil</i>	The command could not be run.

Examples

Retrieves a handle to Global Optimization run mode in the `x_mainSDB` setup database.

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>1001
axlGetRunMode( x_mainSDB "Global Optimization" )
=> 1058
```

Related Topics

[axlReadResDB](#)

[axlRunAllTests](#)

[axlRunAllTestsWithCallback](#)

axlGetRunModes

```
axlGetRunModes (
    x_mainSDB
)
=> l_modes / nil
```

Description

Returns a list of available run modes from the specified setup database.

Argument

<i>x_mainSDB</i>	Setup database handle.

Value Returned

<i>l_modes</i>	List of handle to run modes and the run mode names.
<i>nil</i>	The command could not be run.

Examples

Get a list of all the available run modes:

```
axlGetRunModes( x_mainSDB )
=> (1182
("Local Optimization" "Global Optimization" "Monte Carlo Sampling" "Improve Yield"
"High Yield Estimation" "Sensitivity Analysis" "Create Worst Case Corners" "Single
Run, Sweeps and Corners" "Size Over Corners" )
```

axlGetRunOption

```
axlGetRunOption(  
    x_hsdb  
    t_mode  
    t_runoptName  
)  
=> x_runOption / nil
```

Description

Returns a handle to the named run option (*t_runoptName*) in the setup database for the specified run mode (*t_mode*).

Arguments

<i>x_hsdb</i>	Setup database handle.
<i>t_mode</i>	Run mode. Valid values are: ■ Sampling ■ Global Optimization ■ Local Optimization ■ Monte Carlo Sampling
<i>t_runoptName</i>	Run option name. Valid Values depend on <i>t_mode</i> as follows: For Sampling: ■ <i>points</i>: Number of sampling points. For Global Optimization: ■ <i>tillsatisfied</i>: Optimization stops when all goals are met. ■ <i>timelimit</i>: Optimization stops when the program reaches the time limit (in minutes). ■ <i>numpoints</i>: Optimization stops when the program reaches the number of points.

Virtuoso ADE SKILL Reference

Run-Related Functions

- `ptswithnoimprovement`: Optimization stops when there is no improvement for the number of points.

For Local Optimization:

- `effort`: Optimization effort
- `tillsatisfied`: Optimization stops when all goals are met.
- `timelimit`: Optimization stops when the program reaches the time limit (in minutes).
- `numpoints`: Optimization stops when the program reaches the number of points.
- `ptswithnoimprovement`: Optimization stops when there is no improvement for the number of points.

For Monte Carlo Sampling:

- `mcmethod`: Monte Carlo Sampling method.
- `mcnumpoints`: Number of Monte Carlo sampling points.

Typically, this number should be at least the number of statistical variables.

Value Returned

<code>x_runOption</code>	Handle to the named run option.
<code>nil</code>	The command could not be run.

Examples

Returns a handle to the named run option (`t_runoptName`) in the setup database for the Monte Carlo Sampling run mode

```
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
runopth = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "savemismatch")
=> 3862
axlGetRunOptionValue(runopth)
=> "0"
```

Virtuoso ADE SKILL Reference

Run-Related Functions

Related Topics

[axlGetRunOptionName](#)

[axlSetRunOptionName](#)

axlGetRunOptionName

```
axlGetRunOptionName(  
    x_runOption  
)  
=> t_runoptName / nil
```

Description

Returns the run option name.

Argument

<i>x_runOption</i>	Handle to the run option.

Value Returned

<i>t_runoptName</i>	Run option name.
nil	The command was unsuccessful.

Examples

Returns the name of run option for which run handle is set to runopth.

```
runopth = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "savemismatch")  
=> 1035  
axlGetRunOptionName( runopth )  
=> "savemismatch"
```

Related Topics

[Run option handle](#)

[axlGetRunOption](#)

[axlPutRunOption](#)

axlGetRunOptions

```
axlGetRunOptions(  
    x_hsdb  
    t_runModeName  
)  
=> l_list / nil
```

Description

Returns a list containing a handle to all run option names for the specified run mode.

Arguments

<code>x_hsdb</code>	Setup database handle.
<code>t_runModeName</code>	Name of the run mode for which you want to get run options. The name of the run mode must be same as it is displayed in the <i>Run Mode</i> list in GUI.

Value Returned

<code>l_list</code>	List containing a handle to all the run options in the setup database and a list of all run option names.
<code>nil</code>	The command was unsuccessful.

Examples

The following example retrieves the names of run options for the Monte Carlo run mode:

```
session = axlGetWindowSession()  
x_mainSDB = axlGetMainSetupDB(session)  
axlGetRunOptions(x_mainSDB "Monte Carlo Sampling")  
  
(1452 ("dutsummary" "ignoreflag" "mcmethod" "mcnumpoints" "mcnumbins"  
      "mcStopEarly" "mcStopMethod" "samplingmode" "saveprocess" "savemismatch"  
      "mcreferencepoint" "donominal" "saveallplots" "montecarloseed"  
      "mcstartingrunnumber" "mcYieldTarget" "mcYieldAlphaLimit")  
)  
; you can further get the value of one of the run options, as shown below.  
x_runOption = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "mcnumpoints")  
axlGetRunOptionValue(x_runOption)  
"200"
```

axlGetRunOptionValue

```
axlGetRunOptionValue(  
    x_runOption  
)  
=> t_runoptValue / nil
```

Description

Returns the value associated with the provided run option.

Argument

<code>x_runOption</code>	<u>Run option handle.</u>

Value Returned

<code>t_runoptValue</code>	Value of the given run option.
<code>nil</code>	Unsuccessful operation.

Examples

The following example code shows how you can view the value for a Monte Carlo run option:

```
session = axlGetWindowSession()  
x_mainSDB = axlGetMainSetupDB(session)  
axlGetRunOptions( x_mainSDB "Monte Carlo Sampling")  
  
(1452  
    ("dutsummary" "ignoreflag" "mcmethod" "mcnumpoints" "mcnumbins"  
     "mcStopEarly" "mcStopMethod" "samplingmode" "saveprocess" "savemismatch"  
     "mcreferencepoint" "donominal" "saveallplots" "montecarloseed"  
     "mcstartingrunnumber" "mcYieldTarget" "mcYieldAlphaLimit"  
    )  
)  
  
x_runOption = axlGetRunOption(1001 "Monte Carlo Sampling" "mcnumpoints")  
axlGetRunOptionValue(x_runOption)  
=> "200"
```

Related Topics

[axlGetRunOption](#)

[axlPutRunOption](#)

axlGetRunStatus

```
axlGetRunStatus(  
    t_sessionName  
    [ ?optionName t_optionName ]  
    [ ?historyName t_historyName ]  
)  
=> l_statusValues
```

Description

Returns the completion status in terms of the number of points, tests, or corners completed for all histories running in the given session or for the specified history.

Arguments

t_sessionName Name of the session

?optionName t_optionName

A value that specifies the elements for which the status value is to be returned.

Valid values are:

all: Returns the number of points for which simulation is complete and the total number of points.

Tests: Returns the number of tests for which simulation is complete and the total number of tests.

Corners: Returns number of corners for which simulation is complete and the total number of corners.

Default value: all

?historyName t_historyName

Name of the history item for which the status is to be returned.

The default value is " "

If this argument is not provided, the tool returns the completion status for the given session. In this case, the total number of points, tests, or corners in the returned value is the total number of points, tests, or corners run across all the histories running in the given session.

Virtuoso ADE SKILL Reference

Run-Related Functions

Value Returned

l_statusValues

The returned list contains the following two values:

- The number of points, tests, or corners completed
- The total number of points, tests, or corners to be run in the given session

Examples

The following example code shows how the completion status for a simulation run is returned:

```
axlSession=axlGetWindowSession( hiGetCurrentWindow())
=> "session0"
axlGetRunStatus("session0")
=> (4 14)
; The returned value suggests that out of the total 14 points to be run in the
session session0, simulations are complete for 4 points.
;
axlGetRunStatus("session0" ?historyName "Interactive.10" ?optionName "tests")
=> (1 2)
; The returned value suggests that out of the total two tests in the Interactive.10
history of session0, simulations are complete for one test.
;
axlGetRunStatus("session0" ?historyName "Interactive.10" ?optionName "corners")
=> (4 8)
; The returned value suggests that out of the total eight corners to be run in the
Interactive.10 history of session0, simulations are complete for four corners.
```

Related Topics

Starting a Simulation

axlIsSimUsingStatParams

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

Description

Returns `t` if statistical variables are being set or varied for a particular simulation run. For example, statistical parameters in a Monte Carlo run or a statistical corner for Improve Yield.

Arguments

None

Value Returned

<code>t</code>	Successful operation
<code>nil</code>	The command was unsuccessful.

Examples

The following example checks whether statistical variables are being set for the current simulation run:

```
axlIsSimUsingStatParams()  
=> t
```

axlPutRunOption

```
axlPutRunOption(  
    x_hsdb  
    t_mode  
    t_runoptName  
)  
=> x_runOption / nil
```

Description

Adds a run option to the setup database or edits an existing one and returns the handle to the option. The list of valid option names (*t_runoptName*) depends on the run mode (*t_mode*).

Note: Any unsupported option names you specify will have unspecified effects on the behavior of the run mode. There are no run options for the Single Run, Sweeps and Corners run mode.

Arguments

<i>x_hsdb</i>	Setup database handle.
<i>t_mode</i>	Run mode. The valid values are: ■ Sampling ■ Global Optimization ■ Local Optimization ■ Monte Carlo Sampling

Virtuoso ADE SKILL Reference

Run-Related Functions

t_runoptName

Run option name.

The valid values depend on *t_mode* as follows:

For Sampling:

- *points*: Number of sampling points

For Global Optimization:

- *tillsatisfied*: Optimization stops when all goals are met.
- *timelimit*: Optimization stops when the program reaches the time limit (in minutes).
- *numpoints*: Optimization stops when the program reaches the number of points.
- *ptswithnoimprovement*

For Local Optimization:

- *effort*: Optimization effort.
- *tillsatisfied*: Optimization stops when all goals are met.
- *timelimit*: Optimization stops when the program reaches the time limit (in minutes).
- *numpoints*: Optimization stops when the program reaches the number of points.
- *ptswithnoimprovement*: Optimization stops when there is no improvement for the number of points.

For Monte Carlo Sampling:

- *mcmethod*: Monte Carlo Sampling method.
- *mcnumpoints*: Number of Monte Carlo sampling points.

Note: Typically, this number should be at least the number of statistical variables.

Virtuoso ADE SKILL Reference

Run-Related Functions

Value Returned

<code>x_runOption</code>	Handle to run option.
<code>nil</code>	Unsuccessful operation.

Examples

The following example adds the `points` run option in Monte Carlo mode to the setup database:

```
axlPutRunOption( 1004 "Monte Carlo Sampling" "points")  
=> 1048
```

Related Topics

[axlGetRunOption](#)

[axlGetRunOptions](#)

[axlRunAllTests](#)

[axlRunAllTestsWithCallback](#)

axlRunAllTests

```
axlRunAllTests(  
    t_session  
    t_mode  
)  
=> x_runid / nil
```

Description

Starts a run with all enabled tests.

Arguments

<i>t_session</i>	Session name.
<i>t_mode</i>	Specifies the run mode. Valid values are: ■ Single Run, Sweeps and Corners ■ Sampling ■ Global Optimization ■ Local Optimization ■ Monte Carlo Sampling

Value Returned

<i>x_runid</i>	Run ID.
<i>nil</i>	The command could not be run.

Examples

The following example starts a run:

```
axlRunAllTests( "session0" "Single Run, Sweeps and Corners" )  
=> 1
```

Virtuoso ADE SKILL Reference

Run-Related Functions

Related Topics

[Starting a Simulation](#)

axlRunAllTestsWithCallback

```
axlRunAllTestsWithCallback(  
    t_session  
    t_mode  
    t_callback  
)  
=> x_runid / nil
```

Description

Starts a simulation run with all enabled tests and specifies a SKILL expression to call upon their completion.

Arguments

<i>t_session</i>	Session name.
<i>t_mode</i>	Run mode. Valid values are: ■ Single Run, Sweeps and Corners ■ Sampling ■ Global Optimization ■ Local Optimization ■ Monte Carlo Sampling
<i>t_callback</i>	SKILL expression for callback.

Value Returned

<i>x_runid</i>	Run ID of the simulation.
<i>nil</i>	The command was unsuccessful.

Examples

The following example starts a simulation run for session `data_session` in mode `Single Run, Sweeps and Corners`.

Virtuoso ADE SKILL Reference

Run-Related Functions

```
axlRunAllTestsWithCallback( ( axlCreateSession "data_session" ) "Single Run,  
Sweeps and Corners" "( "printf(\"run complete\")" )  
=> 1001
```

Related Topics

[axlCreateSession](#)

axlRunSimulation

```
axlRunSimulation(  
    [ ?session t_session ]  
    [ ?callback t_callback ]  
)  
=> x_runid / nil
```

Description

Starts a simulation run of all enabled tests and specifies a SKILL expression to call upon completion.

Arguments

?session t_session Session name.

?callback t_callback

SKILL expression for callback.

Value Returned

x_runid Run ID.

nil Unsuccessful operation.

Examples

The following example runs simulation for all enabled tests in the current session.

```
axlRunSimulation()  
=> 1001
```

In the following example, the tool runs simulation for all enabled tests in the current session. After the simulation is complete, the function will also execute the callback script provided by the callback argument.

```
axlRunSimulation( ?session "session0" ?callback "printf(\"Run Complete\")")  
=> 1001
```

Related Topics

Starting a Simulation

axlSetCurrentRunMode

```
axlSetCurrentRunMode(  
    x_hsdb  
    t_mode  
)  
=> t / nil
```

Description

Sets the current simulation run mode for the given database.

Arguments

x_hsdb

Setup database handle.

t_mode

Run mode.

Valid Values for ADE Explorer:

- "Single Run, Sweeps and Corners"
- "Monte Carlo Sampling"

Valid Values for ADE XL and ADE GXL:

- "Single Run, Sweeps and Corners"
- "Monte Carlo Sampling"
- "Global Optimization"
- "Local Optimization"
- "Improve Yield"
- "High Yield Estimation"
- "Sensitivity Analysis"

Virtuoso ADE SKILL Reference

Run-Related Functions

Valid Values for ADE Assembler:

- "Single Run, Sweeps and Corners"
- "Monte Carlo Sampling"
- "Global Optimization"
- "Local Optimization"
- "Improve Yield"
- "High Yield Estimation"
- "Sensitivity Analysis"
- "Feasibility Analysis"
- "Worst Case Corners"
- "Manual Tuning"
- "Size Over Corners"
- "Run Plan"
- "Fault Simulation"
- Feasibility Analysis"
- "Worst Case Corners"
- "Manual Tuning"
- "Size Over Corners"
- "Manual Optimization"
- "Conjugate Gradient Optimization"
- "Feasibility Analysis"
- "Worst Case Corners"
- "Manual Tuning"
- "Size Over Corners"

Virtuoso ADE SKILL Reference

Run-Related Functions

Value Returned

t	Successful operation.
nil	Unsuccessful operation.

Examples

Sets the simulation run mode for the given database:

```
sdb = axlGetMainSetupDB(axlGetWindowSession())  
runMode = "Monte Carlo Sampling"  
axlSetCurrentRunMode(sdb runMode)
```

Related Topics

[Run Toolbar](#)

axlImportPreRunScript

```
axlImportPreRunScript(  
    t_sessionName  
    t_testName  
    t_preRunScriptName  
)  
=> t_preRunScriptName / nil
```

Description

Imports and attaches the given script to the specified test. In the GUI of the application, you can right-click the test name and choose *Add/Edit Pre Run Script* to view or edit the pre-run script attached to the test.

Note: If you make any changes to the test script in the *Add/Edit Pre Run Script* form, the original script is not modified. Instead, the changes are saved in a new script, which is then attached to the test.

Arguments

<i>t_session</i>	Name of the session.
<i>t_testName</i>	Name of the test to which the imported script is attached.
<i>t_preRunScriptName</i>	Location path and name of the script to be imported.

Value Returned

<i>t_preRunScriptName</i>	Name of the pre-run script imported for the given test.
<i>nil</i>	Unsuccessful operation.

Examples

The following example attaches script `myscript` to test `AmpTest1` in the current session.

```
axlSession=axlGetWindowSession( hiGetCurrentWindow() )  
=> "session0"  
axlImportPreRunScript("session0", "AmpTest1", "./myscript")  
=> "myscript"
```

Virtuoso ADE SKILL Reference

Run-Related Functions

Related Topics

[axlSetPreRunScriptEnabled](#)

axlSetParasiticRunMode

```
axlSetParasiticRunMode(  
    x_mainSDB  
    t_runModeName  
  
)  
=> t / nil
```

Description

Sets the parasitic run mode for the given setup.

Arguments

<i>x_mainSDB</i>	Handle to the setup database
<i>t_runModeName</i>	Run mode to be set
	Possible values:
	"No Parasitics/LDE"
	"Schematic Estimates (Parasitics)"
	"Extracted (Parasitics/LDE)"
	"Layout (Parasitics/LDE)"

Value Returned

t	Successful operation
nil	Unsuccessful operation

Examples

The following example returns the name of the parasitic run mode set in the active session:

```
axlSession=axlGetWindowSession()  
=>"session0"  
x_mainSDB=axlGetMainSetupDB( axlSession )  
=> 1001  
axlGetParasiticRunMode(x_mainSDB "Extracted (Parasitics/LDE)")
```

Related Topics

[axlGetMainSetupDB](#)

Virtuoso ADE SKILL Reference

Run-Related Functions

axlGetWindowSession

axlGetParasiticRunMode

axlSetPreRunScript

```
axlSetPreRunScript(  
    t_sessionName  
    t_testName  
    t_scriptName  
)  
=> t_filePath / nil
```

Description

Sets or adds a pre-run script for a test.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>t_testName</i>	Name of the test for which you want to set or add a pre-run script.
<i>t_scriptName</i>	Path to the file that contains the script.

Value Returned

<i>t_filePath</i>	Path to the script file.
<i>nil</i>	The command was unsuccessful.

Examples

The following example adds a pre-run script for the `AmpTest1` test:

```
axlsession=axlGetWindowSession( hiGetCurrentWindow() )  
=> "session0"  
axlSetPreRunScript(axlsession, "AmpTest1", "./myScript")  
=> "./myScript"  
axlSetPreRunScriptEnabled(axlsession, "AmpTest1", t)  
=> t
```

Related Topics

[axlGetPreRunScript](#), [axlSetPreRunScriptEnabled](#)

axlSetPreRunScriptEnabled

```
axlSetPreRunScriptEnabled(  
    t_sessionName  
    t_testName  
    g_enabled  
)  
=> t / nil
```

Description

Enables or disables execution of pre-run scripts before running simulations.

Arguments

<i>t_session</i>	Name of the session.
<i>t_testName</i>	Name of the test for which you want to enable or disable execution of a pre-run script.
<i>g_enabled</i>	Specifies if the execution of a pre-run script is to be enabled or disabled.

Value Returned

<i>t</i>	Execution of pre-run scripts was enabled or disabled.
<i>nil</i>	The command was unsuccessful.

Examples

The following example enables the execution of pre-run script for the `AmpTest1` test:

```
axlSession=axlGetWindowSession( hiGetCurrentWindow() )  
"session0"  
axlSetPreRunScriptEnabled(axlSession, "AmpTest1", t)  
=> t
```

Related Topics

[axlGetPreRunScript](#)

[axlSetPreRunScriptEnabled](#)

Virtuoso ADE SKILL Reference

Run-Related Functions

axlImportPreRunScript

axlSetRunDistributeOptions

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

Description

Sets the specified run option settings for the given setup database. These settings are also visible in the Run Options form.

Arguments

<i>x_hsdb</i>	Handle to the setup database.
<i>?RunIn t_runIn</i>	Describes how multiple simulations need to run. Valid values: Parallel, Serial.
<i>?DivideJobs t_divideJobs</i>	Specifies how the ICRPs can be divided among the simulation runs. Valid values: Specify, Equally.
<i>?JobLimit n_jobLimit</i>	Specifies the maximum number of jobs that can run when <i>?DivideJobs</i> is set to Specify. Note: This value is not considered when <i>?DivideJobs</i> is set to Equally.

Virtuoso ADE SKILL Reference

Run-Related Functions

Value Returned

t	Run option settings are set for the given setup database.
nil	The command was unsuccessful.

Examples

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

```
sdb = axlGetMainSetupDB(axlGetWindowSession())
axlSetRunDistributeOptions(sdb ?RunIn 'Parallel ?DivideJobs 'Specify ?JobLimit 3)
=> t
```

Related Topics

[axlGetRunDistributeOptions](#)

axlSetRunOptionName

```
axlSetRunOptionName(  
    x_runOption  
    t_runoptName  
)  
=> t / nil
```

Description

Sets the run option name.

Arguments

<i>x_runOption</i>	<u>Run option handle.</u>
<i>t_runoptName</i>	Run option name.

Value Returned

t	Run option name was set.
nil	The command was unsuccessful.

Examples

The following example sets a name for the given run option:

```
runopth = axlGetRunOption( x_mainSDB "Monte Carlo Sampling" "savemismatch")  
=> 1048  
axlSetRunOptionName( runopth "points" )  
t
```

Related Topics

[axlGetRunOption](#)

[axlGetRunOptionName](#)

[axlGetRunOptions](#)

axlStop

```
axlStop(  
    t_session  
    x_runid  
)  
=> t / nil
```

Description

Stops a run based on its ID.

Arguments

<i>t_session</i>	Session Name.
<i>x_runid</i>	Run ID.

Value Returned

<i>t</i>	The run was stopped.
<i>nil</i>	The command was unsuccessful.

Examples

The following stops a run with the specified ID:

```
sess=axlGetWindowSession(window(3))  
=> "session0"  
id = (axlRunAllTests sess "Global Optimization")  
=> 0  
axlStop(sess id)  
=> t
```

axlStopAll

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

Description

Stops all runs currently evaluating in the session.

Argument

<i>t_session</i>	Session name.

Value Returned

t	Stopped all runs in the current session.
nil	The command was unsuccessful.

Examples

```
sess=axlGetWindowSession(window(3))  
=> "session0"  
axlStopAll( sess)  
=> t
```

axlReadHistoryResDB

```
axlReadHistoryResDB(  
    t_historyName  
    [ ?session t_sessionName ]  
)  
=> h_ResultsDBObj / nil
```

Description

Returns a handle to the results database saved with the specified history.

This handle is similar to the handle that is returned by the [axlReadResDB](#) function, You can use this handle to access various database objects in the result. For more details, refer to [axlReadResDB](#).

Arguments

<i>t_historyName</i>	Name of the history.
<i>?session t_sessionName</i>	Name of the session.

Value Returned

<i>h_ResultsDBObj</i>	Handle to the results database.
<i>nil</i>	The command was unsuccessful.

Examples

The following code returns a handle `rdbHandle` to the results database for a history named `CornerResults`:

```
sess=axlGetWindowSession(window(3))  
rdbHandle=axlReadHistoryResDB("CornerResults" ?session sess)  
=>axlrdb@0x1949a418
```

axlReadResDB

```
axlReadResDB(  
    t_ResultsDBFileName  
)  
=> h_ResultsDBObj / nil
```

Description

Returns a handle to the specified results database. This handle provides read-only access to the results database that contains objects of the following types:

- point: a design point
- corner: a corner defined for a particular design point
- test: a test defined for a particular corner
- param: a parameter defined for a particular corner
- output: an output defined for a particular test

There is a hierarchical relationship between the instances of these objects. For example, a point is associated with one or more corners. Each corner is associated with one or more tests. Each test is associated with zero or more outputs, and so on. As a result of this relationship, for an object, you can access the properties of the object itself and other objects related to it.

Each object has:

- The following properties:
 - name: Returns the name of the object
 - value: Returns its value; and a property that returns ID of the parent object. For example, an object of type corner has a property
 - pointID: Returns the ID of the parent point object.
- A set of member functions that return the instances of that object type and other related types. For example, using an instance of type output, you can get the value of an output object and its parent test instance. Using the functions given for a test instance, you can get a corner instance. For a corner instance, you can get the parameters which were used to generate the output, as well as the point and its ID.

In addition, the result database provides a function `help()` for each object of the above mentioned types to displays the list and description of functions that can be called using that particular object. For example, function call `help('corner')` displays a list of all the

Virtuoso ADE SKILL Reference

Run-Related Functions

functions that can be called using an object of type `corner`. `help('all')` displays help for all the object types.

Note: Alternatively, you can call the `axlReadHistoryResDB` function to return a handle to the specified results database.

Argument

<code>t_ResultsDBFileName</code>	Name of the results database file.

Value Returned

<code>h_ResultsDBObj</code>	Handle to the results database.
<code>nil</code>	The command was unsuccessful.

Examples

The following code returns a handle `rdb` to the results database for a history named `CornerResults`:

```
historyname="Interactive.0"
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=>
x_history=axlGetHistoryEntry(x_mainSDB historyname)
=>
rdbPath=axlGetHistoryResults(x_history)
=>
rdb=axlReadResDB(rdbPath)
=> axlrdb@0x1949a438
; the returned value is a handle to the results database
```

The following code opens the results database and displays built-in help:

```
rdb->help()
```

Virtuoso ADE SKILL Reference

Run-Related Functions

The help is displayed in the CIW, as shown in the following figure:

```
Toplevel Help:

Functions:
  corner(t_cornerName x_pointID) => o_cornerInst
    Returns the corner indicated by name and point ID.
  corners([?name t_cornerName] [?point x_pointID] [?sortBy 'name'|'point']) => l_cornerInst
    Returns list of corner instances, which may be narrowed by supplying corner name or point ID.
  help(['point'|'corner'|'test'|'output'|'param'|'all']) => t
    Displays help for a particular instance type ("all" for all types)
  output(t_outputName t_testName t_cornerName x_pointID) => o_outputInst
    Returns the output instance for a given test, point, and corner.
  outputs([?type 'expr'|'signal'|'devCheck'] [?sortBy 'name'|'point'|'corner'|'test'|'value'|'type'] [?evalType
"point"|"corners"|"sweeps"|"all"] [?filterStats t|nil]) => l_outputInst
    Returns list of output instances, which may be narrowed by supplying the output type.
  param(t_paramName t_cornerName x_pointID) => o_paramInst
    Returns a parameter instance for a given point and corner.
  params([?name t_name] [?corner t_cornerName] [?point x_pointID] [?type 'fixed'|'design'|'corner'] [?sortBy
'name'|'point'|'corner'|'value'|'type']) => l_paramInst
    Returns the list of parameter instances, which may optionally be filtered by type.
  point(x_pointID) => o_pointInst
    Returns a specific point instance for a given point ID.
  points([?point x_pointID] [?limit x_numBestPoints] [?sortBy 'id'|'best']) => l_pointInst
    Returns a list of all point instances, which may optionally be limited to a subset of best points.
  relxSetup(t_name t_age t_cornerName x_pointID) => o_relxSetupInst
    Returns a specific reliability setup instance.
  relxSetups([?name t_name] [?age t_age] [?point x_pointID] [?corner t_cornerName] [?sortBy
'name'|'point'|'corner']) => l_relxSetupInst
    Returns list of reliability setup instances, which may be narrowed by supplying the point ID, corner
name, age value or reliability setup name.
  test(t_name t_cornerName x_pointID) => o_testInst
    Returns a specific test instance.
  tests([?point x_pointID] [?corner t_cornerName] [?name t_name] [?sortBy 'name'|'point'|'corner']) =>
l_testInst
    Returns list of test instances, which may be narrowed by supplying the point ID, corner name, or tes
name.
t
```

The example code given below shows how to use the handle to the results database to explore the result values. The handle to the first point in the results database is obtained to display the outputs of type expression with their values. The results are sorted by corner.

```
historyname="Interactive.1"
x_mainSDB=axlGetMainSetupDB(axlGetWindowSession())
=> 1001

; returns handle to the setup database of the current session
x_history=axlGetHistoryEntry(x_mainSDB historyname)
=> 4932

; returns handle to the given history
rdbPath=axlGetHistoryResults(x_history)
=> "/servers/user1/opamplib/ampTest/maestro/results/maestro/Interactive.82.rdb"

rdb=axlReadResDB(rdbPath)
=> axlrdb@0x1949a438

; The following statement returns the point object for design point 1.
pt = rdb->point(1)
```


Virtuoso ADE SKILL Reference

Run-Related Functions

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

The following functions calls describe how to access test objects in a result database:

```
; The following function lists all test instances and their properties.
```

```
rdb->tests()
```

```
; The following function returns a test instance with the given test name, from
; the given corner for point 1.
```

```
tst = rdb->test(<t_testName> <t_cornerName> 1)
```

```
; The following statement returns the execution host for the test instance.
```

```
tst->host
```

```
; The following statement returns the run status for the test
```

```
tst->status
```

```
; The following statement returns the start time and stop time the test in the
; default format, such as 'Mon Sep 9 22:25:01 EDT 2013'
```

```
Test->startTime()
```

```
Test->stopTime()
```

```
; You can change the format in which the start time and stop time is displayed. For
; this, specify the format as shown below. You can change the fromat, as
appropriate.
```

```
tst->startTime("h:m:s ap")
```

```
; Time would be displayed in this format: : '10:25:1 pm'
```

```
tst->stopTime("h:m:ss:zzz ap")
```

```
; Time would be displayed in this format: : '12:25:01:035 pm'
```

```
; The following code gets the parent corner for the test and returns the parameters
;(sorted by name) for that corner and their values.
```

```
foreach(mapcar p tst->corner()->params(?sortBy 'name)
```

```
  list( p->name p->value)
```

```
)
```

Consider an example setup that contains a test with one output that is measured across corners and four outputs measured across sweeps. Other outputs are evaluated for each point and each corner individually.

Virtuoso ADE SKILL Reference

Run-Related Functions

By default, the `outputs` function of the results database object returns all outputs for all points except those that are measured across corners, sweeps, or all sweep points and corners.

```
; The following function call returns the names of all outputs for two points in
; the setup:
```

```
rdb->outputs()
```

```
=> rdb->outputs()~>name
("/vin" "/out" "pw" "VT" "ymax2" "/net1" "pw" "/vin" "/out" "pw" "VT" "ymax2"
"/net1" "pw" "/vin" "/out" "pw" "VT" "ymax2" "/net1" "pw" "/vin" "/out" "pw")
```

The following code shows how to read the results of expressions measured across corners or sweeps:

```
; To access the outputs measured across corners, use the ?evalType argument of the
; outputs function:
```

```
rdb->outputs(?evalType "corners")~>name
```

```
("macOut" "macOut")
```

```
; here, the name of output remains same for all points.
```

```
; The following function call returns the output value of macOut for each point in
; the setup:
```

```
rdb->outputs(?evalType "corners")~>value
```

```
(-0.0317678 -0.0287915)
```

```
; The following code prints the names and values of the outputs evaluated across
; corners for each point
```

```
ops = rdb->outputs(?evalType "corners")
```

```
=>(axlrbocorners@0x27b57410 axlrbocorners@0x27b57428)
```

```
foreach(op ops printf("%s      %N      %N\n" op->name op->value ))
```

```
=>macOut      -0.0317678
```

```
macOut      -0.0287915
```

```
(axlrbocorners@0x27b57410 axlrbocorners@0x27b57428)
```

```
; Similarly, the following code prints the names and values of the outputs evaluated
; across sweeps
```

```
ops = rdb->outputs(?evalType "sweeps" ?name "MAS")
```

```
=>( axlrbosweeps@0x27b57458 axlrbosweeps@0x27b57470
axlrbosweeps@0x27b57488 axlrbosweeps@0x27b574a0 ...)
```

```
;
```

```
foreach(op ops printf("%s      %N\n" op->name op->value))
```

```
=> MAS      0.03313052
```

```
MAS_Dep      1.033131
```

```
MAS_wave1      "wave"
```

```
MAS      0.06797625
```

```
MAS_Dep      1.067976
```

```
....
```

axlSetRunOptionValue

```
axlSetRunOptionValue(  
    x_runOptionHandle  
    t_runOptionValue  
)  
=> t / nil
```

Description

Sets a value for the given run option.

Arguments

x_runOptionHandle

Handle to the run option for which you need to set a value.

To find the valid names of run options for a run mode, use the [axlGetRunOptions](#) function.

t_runOptionValue

Value of the given run option.

Value Returned

t A value was set for the given run option.

nil The command was unsuccessful.

Examples

The following example code displays and changes the value for a Monte Carlo run option:

```
session = axlGetWindowSession()  
x_mainSDB = axlGetMainSetupDB(session)  
axlGetRunOptions( x_mainSDB "Monte Carlo Sampling")  
  
=> (1452  
    ("dutsummary" "ignoreflag" "mcmethod" "mcnumpoints" "mcnumbins"  
    "mcStopEarly" "mcStopMethod" "samplingmode" "saveprocess" "savemismatch"  
    "mcreferencepoint" "donominal" "saveallplots" "montecarloseed"  
    "mcstartingrunnumber" "mcYieldTarget" "mcYieldAlphaLimit"  
    )  
)
```

Virtuoso ADE SKILL Reference

Run-Related Functions

```
x_runOption = axlGetRunOption(x_mainSDB "Monte Carlo Sampling" "mcnumpoints")
axlGetRunOptionValue(x_runOption)
=> "200"
;; changing the value of mcnumpoints to 400

axlSetRunOptionValue(x_runOption "400")
=> t

;; setting a value for the savemismatch option for Monte Carlo
(axlSetRunOptionValue (axlPutRunOption (axlGetActiveSetup (axlGetMainSetupDB (
axlGetWindowSession ))) "Monte Carlo Sampling" "savemismatch") "1")
```

Related Topics

[axlGetRunOptions](#)

Virtuoso ADE SKILL Reference

Run-Related Functions

History Functions

This topic lists the SKILL functions that let you work with histories in ADE Explorer and ADE Assembler.

[maeGetHistoryCloudUUID](#)

[axlGetCurrentHistory](#)

[axlGetDataViewHistoryUserMenu](#)

[axlGetHistory](#)

[axlGetHistoryCheckpoint](#)

[axlGetHistoryEntry](#)

[axlGetHistoryGroup](#)

[axlGetHistoryLock](#)

[axlGetHistoryName](#)

[axlGetHistoryPrefix](#)

[axlGetHistoryResults](#)

[axlGetOverwriteHistory](#)

[axlGetOverwriteHistoryName](#)

[axlLoadHistory](#)

[axlSetHistoryLock](#)

[axlSetHistoryName](#)

Virtuoso ADE SKILL Reference

History Functions

[axlSetHistoryPrefixInPreRunTrigger](#)

[axlSetOverwriteHistory](#)

[axlSetOverwriteHistoryName](#)

[axlOpenResDB](#)

[axlPutHistoryEntry](#)

[axlReEvaluateHistory](#)

[axlRemoveSimulationResults](#)

[axlRestoreHistory](#)

[axlViewHistoryResults](#)

[axlWriteMonteCarloResultsCSV](#)

maeGetHistoryCloudUUID

```
maeGetHistoryCloudUUID(  
    t_sessionName  
    t_historyName  
)  
=> t_UUID / nil
```

Description

Returns the UUID for the specified history saved for a simulation run on the cloud for the given session. UUID is the unique identifier used by ADE Assembler for the histories saved on the cloud. You can use this function when the simulation was run using the `Cloud` distribution method.

Arguments

<i>t_sessionName</i>	Name of the ADE Assembler session.
<i>t_historyName</i>	Name of the history saved for a simulation run on the cloud.

Value Returned

<i>t_UUIDs</i>	UUID, if the history is available on the cloud and was saved by using the <code>Cloud</code> distribution method.
<i>nil</i>	When the session is not found or the history is not of a simulation run on the cloud.

Examples

Returns the UUID of the simulation results saved for a history named `Interactive.0` in Amazon S3 at the
S3://mypath/home/usrName/cloud/d7679895-c748-4380-b086-b1ce2c60866a location.

```
maeGetHistoryCloudUUID("fnxSession0" "Interactive.0")  
"d7679895-c748-4380-b086-b1ce2c60866a"  
=> t
```


axlGetCurrentHistory

```
axlGetCurrentHistory(  
    t_sessionName  
)  
=> x_historyHandle / nil
```

Description

Returns the internal integer value representing the current history entry in active use.

Arguments

<i>t_sessionName</i>	Session name

Value Returned

<i>x_historyHandle</i>	Integer value representing the handle to the currently active history entry is returned.
<i>nil</i>	Integer value representing the handle to the currently active history entry is not returned.

Examples

Returns the integer value 1002 that represents the current history entry.

```
sess=axlGetWindowSession()  
"session0"  
axlGetCurrentHistory( "session0" )  
=> 1002
```

axlGetDataViewHistoryUserMenu

```
axlGetDataViewHistoryUserMenu(  
    t_sessionName  
    x_historyHandle  
)  
=> l_menuStructItems
```

Description

This callback function is called when you right-click a history item in the *History* tab of the Data View assistant. You can override this function in `.cdsinit` to add customized menu items in the popup menus of the history items.

Arguments

<code>t_sessionName</code>	Session name
<code>x_historyHandle</code>	Handle to the currently active history entry.

Value Returned

`l_menuStructItems` A list of menu items that need to be added to the popup menu of the history. Each item in the list contains any one of the following:

- A single menu list with three elements: `list("menu_name" "menu_callback" {"true","false"})`, where `true` sets the menu as disabled and `false` sets it as enabled.
- A list of submenu items:
`list("submenu_name" list("menu_name" "menu_callback" {"true","false"}))`, where the second list is a list of submenu items.

You can create multiple menu and submenus.

Examples

You can override this callback function to add customized menu items to retrieve details of simulation directories or the results database files.

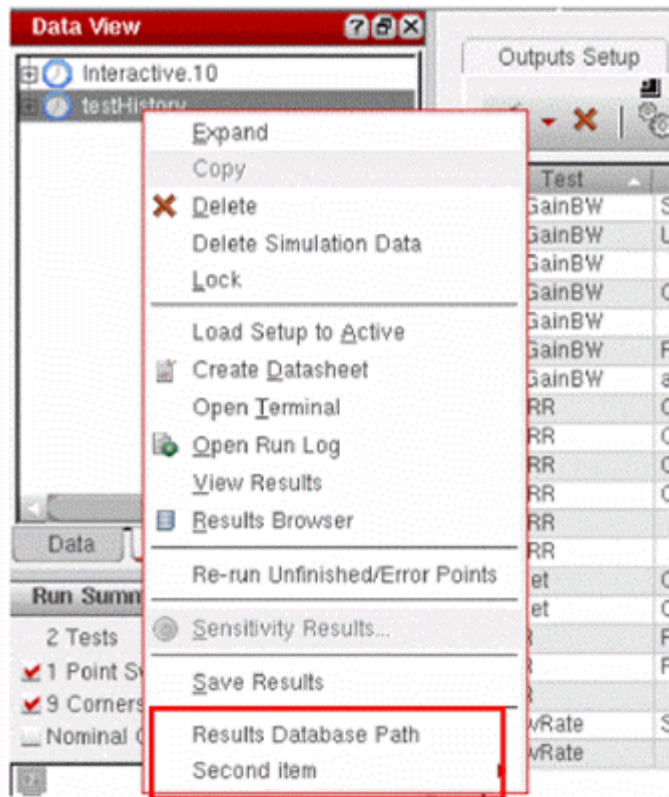
Virtuoso ADE SKILL Reference

History Functions

The following example shows how you can add a new menu item *Simulation Directory Name* to print the location of results directory for the given history. It also adds another dummy menu list with two submenus.

```
define( axlGetDataViewHistoryUserMenu( axlSession historySDB)
val=strcat("Result DB for '" axlGetHistoryName(historySDB) "' is at "
axlGetHistoryResults(historySDB))
    list(list("Results Database Path" "printf(\"%L\" val)" "false")
; 'false' enables the menu item
    list("Second item" list(
        list("first submenu item" "callbackProcedure1" "true")
; 'true' disables the menu item
        list("second submenu item" "callbackProcedure2" "false")))
))
```

This code adds new menu commands to the pop-up menu of history items in the *Data View* pane.



axlGetHistory

```
axlGetHistory(  
    x_hsdb  
)  
=> l_history / nil
```

Description

Returns a list containing a handle to all history entries in the setup database and a list of all the history entries.

Arguments

<i>x_hsdb</i>	Setup database handle

Value Returned

<i>l_history</i>	List containing a handle to all history entries in the setup database is returned along with a list of all the history entries.
<i>nil</i>	Indicates an error.

Examples

Returns a handle to all history entries in the setup database along with a list of all the history entries.

```
sess=axlGetWindowSession()  
"session0"  
sdb=axlGetMainSetupDB(sess)  
1001  
axlGetHistory(sdb)  
=> (1045 ("Interactive.0" "Interactive.1"))
```

axlGetHistoryCheckpoint

```
axlGetHistoryCheckpoint(  
    x_history  
)  
=> x_checkpoint / nil
```

Description

Returns the handle to the checkpoint of a history entry.

Arguments

<i>x_history</i>	Handle to a history entry

Value Returned

<i>x_checkpoint</i>	The handle to the checkpoint of a history entry is returned.
<i>nil</i>	Indicates an error.

Examples

Checks for the existence of a history named `data_design_verification` and loads the history.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
axlLoadHistory( data_sdb )  
=>1203
```

Finds the run mode of the given history name.

```
(defun CCRaxlGetRunModeFromHistoryName (sdbh histName)  
(let (checkPoint)  
  checkPoint = (axlGetHistoryCheckpoint (axlGetHistoryEntry sdbh histName))  
  (axlGetCurrentRunMode checkPoint)  
)  
)  
  
histName = "MonteCarlo.0"  
sess = (axlGetWindowSession)  
sdbh = (axlGetMainSetupDB sess)  
runMode = (CCRaxlGetRunModeFromHistoryName sdbh histName) (printf "Run mode for  
history %s = \"%s\"\\n" histName runMode)
```

Virtuoso ADE SKILL Reference

History Functions

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

[axlLoadHistory](#)

[axlGetHistoryEntry](#)

axlGetHistoryEntry

```
axlGetHistoryEntry(  
    x_hsdb  
    t_historyName  
)  
=> x_history / nil
```

Description

Finds the specified history entry in the setup database and returns the handle to that entry.

Arguments

<i>x_hsdb</i>	Setup database handle
<i>t_historyName</i>	History entry name

Value Returned

<i>x_history</i>	The handle to a history entry is returned.
<i>nil</i>	The handle to a history entry is not returned.

Examples

Finds the history entry `data_design_verification` in the setup database and returns its handle.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
if( axlGetHistoryEntry( data_sdb "data_design_verification" )==0  
error( "Failed to get history item named 'data_design_verification'" ) )  
=> 1004
```

Related Topics

[axlSetMainSetupDB](#)

[axlCreateSession](#)

axlGetHistoryGroup

```
axlGetHistoryGroup(  
    x_hbdb  
    t_histgrpName  
)  
=> x_history / nil
```

Description

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

Arguments

<i>x_hbdb</i>	Setup database handle
<i>t_histgrpName</i>	History group name

Value Returned

<i>x_history</i>	The handle to history group is returned.
nil	The handle to history group is not returned.

Examples

Returns the handle to the history group `ImproveYield.1` in the setup database.

```
axlGetHistoryGroup(1048 "ImproveYield.1")  
=> 2096
```


axlGetHistoryLock

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

Description

Returns the lock status of the given history. When a history item is locked, the corresponding setup details and results cannot be deleted.

Arguments

<i>x_historyHandle</i>	Handle to a history in the setup database

Value Returned

<i>t</i>	Indicates that the specified history is locked in the database
<i>nil</i>	Indicates that the specified history is not locked in the database

Examples

Returns the lock status of the current history.

```
session=axlGetWindowSession()  
=> "session1"  
history1=axlGetCurrentHistory(session)  
=> 1067  
axlGetHistoryLock(history1)  
=> t
```

Returns the lock status of the given checkpoint history.

```
session=axlGetWindowSession()  
=> "session1"  
x_mainSDB=axlGetMainSetupDB(session)  
=> 2468  
handleHistory=axlGetHistoryCheckpoint( axlGetHistoryEntry(x_mainSDB  
"Interactive.1"))  
=> 1067  
axlGetHistoryLock(handleHistory)  
=> nil  
  
; The returned value nil shows that the Interactive.1 history is currently unlocked  
; in the database.
```

Virtuoso ADE SKILL Reference

History Functions

Related Topics

[axlGetCurrentHistory](#)

[axlGetHistoryCheckpoint](#)

[axlSetHistoryLock](#)

axlGetHistoryName

```
axlGetHistoryName(  
    x_historyEntry  
)  
=> t_historyName / nil
```

Description

Returns the name of the history item that holds the data for the latest simulation run.

Arguments

<i>x_historyEntry</i>	Integer argument representing the history entry.

Value Returned

<i>t_historyName</i>	The string value representing the name of the history item is returned.
<i>nil</i>	The string value representing the name of the history item is not returned.

Examples

Returns the name of the history item that holds the data for the latest simulation run.

```
axlGetHistoryName( axlGetCurrentHistory( "session0" ) )  
=> "Interactive.0"
```

axlGetHistoryOutputs

```
axlGetHistoryOutputs(  
    t_sessionName  
    x_historyEntry  
    ?testName x_testName )  
  
=> l_outputStructures / nil
```

Description

Returns a list of output structures for the given history and test. If the test name is not given, returns a list of all outputs from the given history. If a session with the given history is not already created, this function opens the history and creates that session too.

Arguments

<i>t_sessionName</i>	Name of the session.
<i>x_historyEntry</i>	Integer argument representing the history entry.
<i>?testName x_testName</i>	Name of the test for which outputs are to be returned.

Value Returned

<i>l_outputStructure</i>	List of output structures.
<i>s</i>	
<i>nil</i>	History or test is not found in the given session.

Examples

The following example returns the list of outputs for test `TRAN_EXTR_EV2` in history `Interactive.0`:

```
session=axlGetWindowSession()  
=> "fnxSession1"  
sdb = axlGetMainSetupDB(session)  
=> 2708  
axlGetHistoryEntry(2708 "Interactive.0")  
=> 5716  
axlGetHistoryOutputs(session 5716 ?testName "TRAN_EXTR_EV2")  
=> ((sevOutputStruct@0x2dd97ec0 sevOutputStruct@0x2dd97ed8  
sevOutputStruct@0x2dd97ef0 sevOutputStruct@0x2dd97f08 )  
)
```

axlGetHistoryPrefix

```
axlGetHistoryPrefix(  
    x_sessionName  
)  
=> t_historyPrefix / nil
```

Description

Returns the current history prefix value from the given session. The prefix value depends on the run mode selected in the session.

Arguments

<i>t_sessionName</i>	Name of the session

Value Returned

<i>t_historyPrefix</i>	The current history prefix value in the session is returned.
<i>nil</i>	The current history prefix value in the session is not returned.

Examples

Returns the current history prefix from the session.

```
session=axlGetWindowSession()  
=> "session1"  
axlGetHistoryPrefix(session)  
=> "Interactive"
```

Related Topics

[axlGetWindowSession](#)

axlGetHistoryResults

```
axlGetHistoryResults(  
    x_history  
)  
=> t_results / nil
```

Description

Gets the results database from a history entry. This function calls `axlGetResultsLocation` to get the results location.

Argument

<i>x_history</i>	Handle to a history entry

Value Returned

<i>t_results</i>	The results database for the specified history entry is returned.
<i>nil</i>	The results database for the specified history entry is not returned.

Examples

Returns the results database from the specified history entry.

```
data_session = axlCreateSession( "data_session" )  
design_data = axlRunAllTestsWithCallback( data_session "Single Run, Sweeps and  
Corners" "( callbackProcedure )" )  
...  
axlGetHistoryResults( axlGetRunData( data_session design_data ) )  
=> "Interactive.0.rdb"
```

Related Topics

[axlCreateSession](#)

[axlRunAllTestsWithCallback](#)

[axlGetResultsLocation](#)

axlGetOverwriteHistory

```
axlGetOverwriteHistory(  
    x_history  
)  
=> t / nil
```

Description

Returns a Boolean value specifying the status of the *Overwrite History* check box in the Save Options form for the active setup.

Arguments

<i>x_history</i>	Specifies a handle to the active setup.

Value Returned

t	The <i>Overwrite History</i> check box is enabled.
nil	The <i>Overwrite History</i> check box is disabled.

Examples

Returns a boolean value specifying the status of the *Overwrite History* check box in the Save Options form.

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))  
=> 2417  
axlGetOverwriteHistory(x_activeSetup)  
=> t  
axlGetOverwriteHistoryName(x_activeSetup)  
=> "Next History Run"
```

axlGetOverwriteHistoryName

```
axlGetOverwriteHistoryName(  
    x_setup  
)  
=> t_historyName / nil
```

Description

Returns the name of the history that is set to be overwritten for the active setup.

Arguments

<i>x_setup</i>	Handle to the active setup

Value Returned

<i>t_historyName</i>	The name of the history set to be overwritten is returned.
nil	Indicates an error.

Examples

Returns the name of the history that is set to be overwritten for the active setup.

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))  
=> 2417  
axlGetOverwriteHistory(x_activeSetup)  
=> t  
axlGetOverwriteHistoryName(x_activeSetup)  
=> "Next History Run"
```


axlLoadHistory

```
axlLoadHistory(  
    x_to  
    x_from  
)  
=> x_hsdb / nil
```

Description

Copies the setup database branch and returns the handle to the copy.

Arguments

<i>x_to</i>	Handle to target (copied) setup database.
<i>x_from</i>	Handle to source setup database branch.

Value Returned

<i>x_hsdb</i>	The handle to the copied setup database is returned.
<i>nil</i>	The setup database is not copied.

Examples

Copies the setup database branch and returns the handle to the copy.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
axlLoadHistory( data_sdb  
axlGetHistoryCheckpoint( axlGetHistoryEntry( data_sdb "data_design_verification" )  
) )  
=> 1050
```

Related Topics

[axlCreateSession](#)

[axlGetHistoryEntry](#)

[axlSetMainSetupDB](#)

axlSetHistoryLock

```
axlSetHistoryLock(  
    x_handleHistory  
    g_enable  
)  
=> t / nil
```

Description

Locks the specified checkpoint history. After it is locked, you cannot delete the history or the simulation data saved for it.

Arguments

<i>x_handleHistory</i>	Handle to the history in the setup database for which the lock status is to be changed
<i>g_enable</i>	The lock status to be set for the specified history. Valid values: <code>t</code> and <code>nil</code>

Value Returned

<code>t</code>	The specified history is locked successfully.
<code>nil</code>	The specified history is not locked.

Examples

Locks the current history.

```
session=axlGetWindowSession()  
=> "session1"  
handleHistory=axlGetCurrentHistory(session)  
=> 1067  
axlSetHistoryLock(handleHistory t)  
=> t
```

Locks the specified checkpoint history.

```
session=axlGetWindowSession()  
=> "session1"  
x_mainSDB=axlGetMainSetupDB(session)  
=> 2468  
handleHistory=axlGetHistoryCheckpoint( axlGetHistoryEntry(x_mainSDB  
"Interactive.1"))  
=> 1067
```

Virtuoso ADE SKILL Reference

History Functions

```
axlSetHistoryLock(handleHistory t)  
=> t
```

```
; The returned value t shows that the Interactive.1 history has been locked  
; in the database.
```

Related Topics

[axlGetCurrentHistory](#)

[axlGetHistoryCheckpoint](#)

[axlGetHistoryLock](#)

axlSetHistoryName

```
axlSetHistoryName (
    x_historyHandle
    t_newHistoryName
)
=> t / nil
```

Description

Sets a new name for the specified history.

Arguments

<i>x_historyHandle</i>	Handle to the history that you want to rename.
<i>t_newHistoryName</i>	New name to be set for the history.

Value Returned

<i>t</i>	History name is changed successfully.
<i>nil</i>	History name is not changed.

Examples

Renames the specified history to *newHistoryName*.

```
session=axlGetWindowSession()
=> "session0"
x_mainSDB = axlGetMainSetupDB(session)
=> 1001
historyHandle=axlGetHistoryEntry(x_mainSDB "SingleRun.1")
=> 4127
axlSetHistoryName( historyHandle "newHistoryName")
=> t
```

axlSetHistoryPrefixInPreRunTrigger

```
axlSetHistoryPrefixInPreRunTrigger (
    t_session
    t_historyPrefix
)
=> t_historyPrefix / nil
```

Description

Sets a prefix to be used in the history name for a new run.

Arguments

<i>t_session</i>	Name of the current session
<i>t_historyPrefix</i>	Prefix to be used for the new history

Value Returned

<i>t_historyPrefix</i>	The prefix value is set.
<i>nil</i>	The prefix value is not set.

Examples

Sets the history name before running a simulation.

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

axlSetHistoryPrefixInPreRunTrigger(session "newName")
=> "newName"

axlRunSimulation()
```

axlSetOverwriteHistory

```
axlSetOverwriteHistory(  
    x_setup  
    g_overwriteStatus  
)  
=> t / nil
```

Description

Enables or disables the *Overwrite History* check box in the Save Options form for the active setup.

Arguments

<i>x_setup</i>	Handle to the active setup
<i>g_overwriteStatus</i>	Enables or disables the <i>Overwrite History</i> check box in the Save Options form for the specified setup.

Value Returned

<i>t</i>	The <i>Overwrite History</i> check box is enabled or disabled.
<i>nil</i>	Indicates an error.

Examples

Enables the *Overwrite History* check box in the Save Options form.

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))  
=> 2417  
axlSetOverwriteHistory(x_activeSetup t)  
=> t  
axlSetOverwriteHistoryName(x_activeSetup "Interactive.1")  
=> "2535"
```

axlSetOverwriteHistoryName

```
axlSetOverwriteHistoryName(  
    x_setup  
    t_overwriteHistoryName  
)  
=>t / nil
```

Description

Sets the name of the history to be overwritten for the specified active setup.

Arguments

x_setup Handle to the active setup.

t_overwriteHistoryName
Name of the history to be overwritten.

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

Value Returned

t The specified history is overwritten successfully.

nil The specified history is not overwritten

Examples

Overwrites the specified history.

```
x_activeSetup=axlGetActiveSetup(axlGetMainSetupDB(axlGetWindowSession()))  
=>2417  
axlSetOverwriteHistory(x_activeSetup t)  
=>t  
axlSetOverwriteHistoryName(x_activeSetup "Interactive.1")  
=>"2535"
```

axlOpenResDB

```
axlOpenResDB(  
    t_fileName  
)  
=> o_obj / nil
```

Description

Opens the database file specified by *t_fileName*. If the file does not exist, it is created.

Arguments

<i>t_fileName</i>	Database file to be opened.

Value Returned

<i>o_obj</i>	The object handle to the database.
<i>nil</i>	Indicates an error.

Examples

Opens the database file *resDB*.

```
resDB=axlGetHistoryResults(axlGetRunData(session runid))  
obj = axlOpenResDB(resDB)
```

Related Topics

[axlGetHistoryResults](#)

axlPutHistoryEntry

```
axlPutHistoryEntry(  
    x_hsdb  
    t_historyName  
)  
=> x_history / nil
```

Description

Inserts or finds a history entry in the setup database and returns a handle to that entry.

Arguments

<i>x_hsdb</i>	Setup database handle.
<i>t_historyName</i>	History entry name.

Value Returned

<i>x_history</i>	The handle to a history entry is returned.
<i>nil</i>	The handle to a history entry is not returned.

Examples

Inserts or finds the specified history entry in the setup database and returns the handle to that entry.

```
data_sdb = axlGetMainSetupDB(axlGetWindowSession())  
axlPutHistoryEntry( data_sdb "data_design_verification" )  
=> 1006
```

Related Topics

[axlCreateSession](#)

[axlSetMainSetupDB](#)

axlReEvaluateHistory

```
axlReEvaluateHistory(  
    t_sessionName  
    x_historyHandle  
    S_mode  
)  
=> t / nil
```

Description

Re-evaluates the history in the specified mode with respect to the active setup for a given session.

Arguments

<i>t_sessionName</i>	Name of the adexl or maestro view.
<i>x_historyHandle</i>	Setup database handle.
<i>S_mode</i>	Symbol representing the mode. The valid values are: ■ all: re-evaluates expressions and specifications ■ specs: re-evaluates specifications ■ exprs: re-evaluates expressions ■ partial: re-evaluates both expressions and specifications using the partially complete simulation data Default value is all .

Value Returned

<i>t</i>	The history is re-evaluated successfully.
<i>nil</i>	The history is not re-evaluated.

Examples

Re-evaluates the history in the specified mode with respect to the active setup for a given session.

Virtuoso ADE SKILL Reference

History Functions

```
session = axlGetWindowSession( hiGetCurrentWindow() )
history = axlGetCurrentHistory(session)
axlReEvaluateHistory(session history 'all)
```

axlRemoveSimulationResults

```
axlRemoveSimulationResults(  
    x_historySDB  
)  
=> t / nil
```

Description

Removes the simulation results data for the given history. The function removes only the results saved by the simulator. The results database and the history item is not removed.

Note: If you need to remove the entire history, use `axlRemoveElement` for the history element.

To remove the simulation results data for a history from the GUI, right-click the history name in the Data View assistant and choose *Delete Simulation Data* from the context-sensitive menu.

Arguments

<code>x_historySDB</code>	Handle to the history for which the simulation data is to be deleted. This cannot be a handle to the checkpoint of a history.

Value Returned

<code>t</code>	Simulation results data for the given history is successfully deleted.
<code>nil</code>	Deletion of the simulation results data is not successful.

Examples

Deletes the simulation data for the history `Interactive.9`.

```
session = (axlGetWindowSession)  
=> "session0"  
x_mainSDB = axlGetMainSetupDB(session)  
=> 1001  
x_historySDB=axlGetHistoryEntry(x_mainSDB "Interactive.9")  
=> 1115  
axlRemoveSimulationResults(x_historySDB)  
=> t
```

Virtuoso ADE SKILL Reference

History Functions

Related Topics

[axlRemoveElement](#)

axlRestoreHistory

```
axlRestoreHistory(  
    t_session  
    x_historyEntry  
)  
=> t
```

Description

Sets the given history as the active setup in the given session.

Arguments

<i>t_session</i>	Name of session in which setup from the history is to be applied.
<i>x_historyEntry</i>	Setup database handle for a history item.

Value Returned

<i>t</i>	The specified history is set as the active setup.
<i>nil</i>	The specified history is not set as the active setup.

Examples

Sets the specified history as the active setup in the given session.

```
session = (axlGetWindowSession)  
=> "session0"  
main_sdb = axlGetMainSetupDB(session)  
=> 1001  
x_historyEntry=axlGetHistoryEntry(x_mainSDB "Interactive.1")  
=> 1112  
axlRestoreHistory(session x_historyEntry)  
=> t
```

axlViewHistoryResults

```
axlViewHistoryResults(  
    t_session  
    x_hsdb  
)  
=> t
```

Description

Display the results for the specified history item on the *Results* tab of the given session.

Arguments

<i>t_session</i>	Name of session in which the results should be displayed.
<i>x_hsdb</i>	Setup database handle for a history item.

Value Returned

t	The results for the specified history item are displayed in the <i>Results</i> tab.

Examples

Displays the results for the specified history item on the *Results* tab.

```
session = (axlGetWindowSession)  
=> "session0"  
main_sdb = axlGetMainSetupDB(session)  
first_history_sdb = axlGetHistoryEntry(main_sdb caadr( axlGetHistory(  
main_sdb)))  
axlViewHistoryResults(session first_history_sdb)
```

axlWriteMonteCarloResultsCSV

```
axlWriteMonteCarloResultsCSV
    t_session
    t_historyName
    [ ?testName t_testName ]
    [ ?cornerName t_cornerName ]
    [ ?outputName t_outputPath ]
)
=> t / nil
```

Description

Writes the results of the given Monte Carlo run history in CSV format. The tool saves the results for each corner in a separate CSV file.

Virtuoso ADE SKILL Reference

History Functions

Arguments

<code>t_session</code>	Session name or the handle to the session
<code>t_historyName</code>	Name of the Monte Carlo history.
<code>?testName t_testName</code>	Name of the test for which the results are to be exported. Default: <code>nil</code>, which means that the results for all the tests are exported.
<code>?cornerName t_cornerName</code>	Name of the corner for which the results are to be exported. Default: <code>nil</code>, which means that the results for all the corners are exported.
<code>?outputName t_outputPath</code>	Path where the exported .csv files are to be saved. Default: The directory of the cellview

Value Returned

<code>t</code>	The results of the given Monte Carlo run history are written in CSV format.
<code>nil</code>	The results of the given Monte Carlo run history are not written in CSV format.

Examples

Writes the results of the given Monte Carlo run history in CSV format.

```
session = axlGetWindowSession()
=> "session0"
axlWriteMonteCarloResultsCSV("session0" "MonteCarlo.1" ?testName "AC" ?cornerName
"C1" ?outputPath "/tmp/csvfiles/")
=> t
```

Job Policy Functions

This topic lists the SKILL functions that let you configure and manage job policies in ADE Explorer and ADE Assembler.

[axlAddJobPolicy](#)

[axlAttachJobPolicy](#)

[axlDeleteJobPolicy](#)

[axlDetachJobPolicy](#)

[axlJobIntfcDebugPrintf](#)

[axlJobIntfcDebugToFile](#)

[axlJobIntfcDebugp](#)

[axlJobIntfcExitMethod](#)

[axlJobIntfcHealthMethod](#)

[axlJobIntfcSetDebug](#)

[axlJobIntfcStartMethod](#)

[axlJPGUICustDiffer](#)

[axlJPGUICustHiddenSections](#)

[axlJPGUICustHIFields](#)

[axlJPGUICustOffset](#)

[axlJPGUICustReadFromForm](#)

[axlJPGUICustSelected](#)

Virtuoso ADE SKILL Reference

Job Policy Functions

axlRegisterJobIntfc

axlRegisteredJobIntfcNames

axlRegisterJPGUICust

axlGetAttachedJobPolicy

axlGetJobPolicy

axlGetJobPolicyTypes

axlIsCRPPProcess

axlSaveJobPolicy

axlSetJobPolicyProperty

axlStopAllJobs

axlStopJob

maeClearAllTestJobPolicies

maeClearTestJobPolicy

maeCreateNetlistForCorner

maeGetAllJobPolicies

maeGetJobControlMode

maeGetJobPolicy

maeGetJobPolicyByName

maeHasTestJobPolicy

maelsEvaluatorProcess

maelsNetlistProcess

maeSetJobControlMode

maeSetJobPolicy

maeStopAllJobs

Virtuoso ADE SKILL Reference

Job Policy Functions

maeStopJob

axlAddJobPolicy

```
axlAddJobPolicy(  
    t_jobPolicyName  
    t_selectedPath  
    l_jobPolicyProperties  
)  
=> t / nil
```

Description

Adds or saves a job policy at the specified location.

This function does not apply the job policy. It only saves a new job policy at the given location. To use this job policy, you need to set this policy in the Job Policy form or by using the `defaultJobPolicy` environment variable. To apply a job policy while defining its properties, use the `axlAttachJobPolicy` function.

Arguments

<i>t_jobPolicyName</i>	Job policy name.

Virtuoso ADE SKILL Reference

Job Policy Functions

t_selectedPath

Location of the `.cadence` directory where the job policy file is to be stored.

You can store the job policy file in the `.cadence` directory in one of the following default locations specified in the `setup.loc` file at `<your_inst_dir>/share/cds/setup` in your Cadence installation, or customize the `setup.loc` file to specify more locations to save the job policy file:

- `.cadence` directory in the current directory
- The `.cadence` directory in the path specified in the `CDS_WORKAREA` environment variable.
- `$HOME/.cadence` (the `.cadence` directory in your home directory)
- The `.cadence` directory in the path specified in the `CDS_PROJECT` environment variable.
- The `.cadence` directory in the path specified in the `CDS_SITE` environment variable.

Note: Ensure that you have write permissions in the `.cadence` directory where you want to store the job policy file.

The job policy file is saved in the `jobpolicy` directory under the specified `.cadence` directory. The job policy file has the `.jp` extension. For more information, see "Saving a Job Policy".

l_jobPolicyProperties

List of job policy property name-value pairs.

Value Returned

<code>t</code>	The job policy is added.
<code>nil</code>	The job policy is not added.

Examples

Saves the `mypolicy` policy in the `.cadence` directory:

```
axlAddJobPolicy( "mypolicy"  
                "./.cadence"  
                '( nil distributionmethod "LBS"  
                  configuretimeout "1200"
```

Virtuoso ADE SKILL Reference

Job Policy Functions

```
maxjobs "5"  
name "LBS_Policy"  
runtimeout "3600"  
starttimeout "300" ) )  
=> t
```

After saving a job policy, you can set this as the default policy for a session by using the `defaultJobPolicy` environment variable.

Related Topics

[axlAttachJobPolicy](#)

[defaultJobPolicy](#)

[axlAddJobPolicy](#)

axlAttachJobPolicy

```
axlAttachJobPolicy(  
    t_sessionName  
    t_jobPolicyName  
    t_toolName  
    l_jobPolicyProperties  
    [ t_testName ]  
)  
=> t / nil
```

Description

Adds and attaches a job policy to the setup.

The fourth argument is a disembodied property list of job policy properties. The function overwrites properties of the named job policy if it already exists.

Arguments

<i>t_sessionName</i>	The session name
<i>t_jobPolicyName</i>	Job policy name.
<i>t_toolName</i>	Tool name. Valid Values: "ICRP"
<i>l_jobPolicyProperties</i>	List of job policy property name-value pairs.
<i>t_testName</i>	Name of the test to which the job policy is attached.

Value Returned

<i>t</i>	The job policy is added and attached successfully.
<i>nil</i>	The job policy is added and attached unsuccessfully.

Examples

Defines a job policy, mypolicy, and attach it to a test Test1:

```
session0 = axlGetWindowSession(hiGetCurrentWindow())  
axlAttachJobPolicy( "session0" "mypolicy" "ICRP" '( nil distributionmethod "LBS"
```


Virtuoso ADE SKILL Reference

Job Policy Functions

```
configuretimeout "300" maxjobs "5" name "default" runtimeout "3600" starttimeout  
"300" ) "Test1")  
=> t
```

Related Topics

[axlAddJobPolicy](#)

[Setting Up Job Policies](#)

axlDeleteJobPolicy

```
axlDeleteJobPolicy(  
    t_jobPolicyName  
)  
=> t / nil
```

Description

Deletes the named job policy from the setup.

Arguments

t_jobPolicyName Job policy name.

Value Returned

<i>t</i>	The named job policy is deleted.
<i>nil</i>	The named job policy is not deleted.

Examples

Delete the job policy, *mypolicy*:

```
axlDeleteJobPolicy( "mypolicy" )  
=> t
```

axlDetachJobPolicy

```
axlDetachJobPolicy(  
    t_sessionName  
    t_jobType  
    t_testName  
=> t / nil
```

Description

Detaches a job policy from the specified test.

After the job policy is detached, the test uses the default job policy specified for the given session.

Arguments

<i>t_sessionName</i>	Specifies the name of the session from which the job policy needs to be detached.
<i>t_jobType</i>	Specifies the type of the job policy to be detached. Valid Values: "ICRP"
<i>t_testName</i>	Specifies the name of the test from which job policy is to be detached.

Value Returned

t	The parameter value is successfully updated.
nil	The parameter value is not updated.

Examples

Detaches the ICRP job policy from test AC in session session0.

```
axlDetachJobPolicy("session0" "ICRP" "AC")  
=> t
```

axlJobIntfcDebugPrintf

```
axlJobIntfcDebugPrintf(  
    t_formatString  
    [g_arg1 ... g_argn]  
)  
=> t / nil
```

Description

Formats and writes output to the log if interface debugging is enabled.

Arguments

<i>t_formatString</i>	String name
<i>g_arg1</i>	Argument passed

Value Returned

<i>t</i>	Indicates that the interface debugging is enabled.
<i>nil</i>	Indicates that the interface debugging is disabled.

Examples

Formats and writes output to the log.

```
axlJobIntfcDebugPrintf("hello world")  
=> "hello world"  
=> t
```

axlJobIntfcDebugToFile

```
axlJobIntfcDebugToFile(  
    t_file_name  
)  
=> t / nil
```

Description

Enables the interface job debugging and sets the output to a file.

Arguments

<i>t_file_name</i>	File to be opened

Value Returned

t	The interface job debugging is enabled and the output is set to a file.
nil	Indicates an error.

Examples

Enables the interface job debugging and sets the output to a file

```
axlJobIntfcDebugToFile("~/intfc_debug")  
=> t
```

axlJobIntfcDebugp

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

Description

Specifies whether interface debugging is enabled.

Arguments

None

Value Returned

t	Successful operation.
nil	if debugging is disabled.

Examples

Indicates that the interface debugging is disabled.

```
axlJobIntfcDebugp()  
=> nil
```

axlJobIntfcExitMethod

```
axlJobIntfcExitMethod(  
    g_inst  
)  
=> t / nil
```

Description

Job Interface member function used to exit a job. The application usually attempts to call `exit(0)` on remote job cleanup the job's resources properly. However, if this fails or if the tool is forced to kill all jobs, the exit method will be called on every remote job. This method may be called after the job has already exited. This job can also be called multiple times.

Arguments

<i>g_inst</i>	Subclass of <code>axlJobIntfc</code>

Value Returned

<i>t</i>	Indicates that the job is exited.
<i>nil</i>	Indicates that the job is not exited.

Examples

Exits the specified job.

```
;; example setup  
(defclass IPCJob ( axlJobIntfc )  
    (  
        (ipcHandle)  
    ))  
;; axlJobIntfcExitMethod  
(defmethod axlJobIntfcExitMethod ( (inst IPCJob) )  
    (ipcKillProcess inst->ipcHandle)  
)
```

axlJobIntfcHealthMethod

```
axlJobIntfcHealthMethod(  
    g_inst  
    S_currentHealth  
)  
=> S_newHealth
```

Description

Job Interface member function used to return the current health of the job. The application calls this function regularly (currently, every 5 seconds) on each job in order to recognize health changes. The available health types are: `unknown`, `alive`, or `dead`.

- `unknown`: If the job Interface detects that the process is still pending.
- `alive`: If the Job Interface detects that the job has been launched. Otherwise, the tool will eventually receive a start message from the remote job and automatically change the current health.
- `alive`: If the Job Interface detects that the job has exited.
- `dead`: Indicates that the remote communication has failed. Once marked `dead`, the health will not be queried. This is a terminal state.

If no change is detected, the current state should be returned.

Arguments

<i>g_inst</i>	Subclass of <code>axlJobIntfc</code>
<i>S_currentHealth</i>	The current health type of the job. The valid values are <code>unknown</code> , <code>alive</code> , or <code>dead</code> .

Value Returned

<i>S_newHealth</i>	The health type of the job is returned.

Examples

Returns the health type of the job.

```
;; example setup  
(defclass IPCJob ( axlJobIntfc )
```


Virtuoso ADE SKILL Reference

Job Policy Functions

```
(
  (ipcHandle)
)
;; health method
(defmethod axlJobIntfcHealthMethod ( (inst IPCJob) current_health )
  (cond
    ((and (current_health == "unknown")
          (null (zerop (ipcGetExitStatus
inst->ipcHandle))))
      "dead")
    ((and (current_health == "alive")
          (null (ipcIsAliveProcess
inst->ipcHandle))))
      "dead")
    (t
      current_health)))
```

axlJobIntfcSetDebug

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

Description

Enables or disables printing of the job interface diagnostics to the CIW.

Arguments

<i>g_enable</i>	Enables or disables debugging. The valid values are <code>t</code> and <code>nil</code> .

Value Returned

<code>t</code>	The printing of the job interface diagnostics to the CIW is enabled or disabled.
<code>nil</code>	The printing of the job interface diagnostics to the CIW is not enabled or disabled.

Examples

Disables the printing of the job interface diagnostics to the CIW.

```
axlJobIntfcSetDebug(nil)  
=> t
```

axlJobIntfcStartJobArrayMethod

```
axlJobIntfcStartJobArrayMethod(  
    g_inst  
    x_startIndex  
    x_endIndex  
)  
=> t / nil
```

Description

Job interface member function to start a job array where the job IDs are set between the range of the specified start and end indexes for the array. For example, if the start and end indexes are set to 4 and 10 respectively, the jobs with IDs 4, 5, 6, 7, 8, 9, and 10 would be launched.

For each index in the specified range, a new instance of the selected interface class is created. This interface class can be registered using the function `axlRegisterJobIntfc`. Basic properties are set to each of the created instances, such as job ID, job policy, session name, ADE service, standard output file, and error output file. This lets the tool keep control of each individual instance and be able to query the health of each of the jobs started with the job array.

Note: The set properties can be retrieved using the command `inst-><property_name>`. The last instance of the created interface class is passed to the function `axlJobIntfcStartJobArrayMethod`.

Arguments

<i>g_inst</i>	Specifies a subclass of <code>axlJobIntfc</code> .
<i>x_startIndex</i>	Specifies the start index of the job array. This index value represents the ID of the first job in the array.
<i>x_endIndex</i>	Specifies the end index of the job array. This index value represents the ID of the last job in the array.

Value Returned

<i>t</i>	The job array is started successfully.
<i>nil</i>	Indicates an error.

Virtuoso ADE SKILL Reference

Job Policy Functions

Examples

Starts a job array where the job IDs are determined by the specified start and end indexes.

```
;; example setup
(defclass groupSubmitJobIntfc (axlJobIntfc)
)

;; set jobArray to t to enable the job array method, otherwise
axlJobIntfcStartMethod is used
(defClass groupSubmitJobIntfcGUICust (axlJPGUICust)
  (jobArray @initform t)
)

defmethod( axlJobIntfcStartJobArrayMethod ((inst groupSubmitJobIntfc) startIndex
endIndex)
  (let (commandFile fport)
    commandFile = (makeTempFileName "/tmp/jobScript_XXXXX")
    fport = (outfile commandFile "w")
    (fprintf fport "#!/bin/sh\n")
    (fprintf fport "bsub -J \"%s-%s\" inst->davinciService inst->sessionName)
    (fprintf fport "[%s]\"\\n\" sprintf(nil "%d-%d" startIndex endIndex))
    (drain fport)
    (close fport)
    (system sprintf (nil "chmod +x %s" commandFile))
    (ipcBeginProcess commandFile)
  )
)
```

axlJobIntfcStartMethod

```
axlJobIntfcStartMethod(  
    g_inst  
)  
=> t / nil
```

Description

Job Interface member function to start a job. For each new job ID, a new instance of the selected interface class will be created. After some basic properties are set on the instance, it will be passed to the start method of the class.

Arguments

<i>g_inst</i>	Subclass for axlJobIntfc.

Value Returned

t	The job starts successfully.
nil	The job does not start.

Examples

Starts the job.

```
;; example setup  
(defclass IPCJob ( axlJobIntfc )  
  (  
    (ipcHandle)  
  ))  
;; axlJobIntfcStartMethod  
(defmethod axlJobIntfcStartMethod ( (inst IPCJob) )  
  (let ((ipc_handle (ipcBeginProcess inst->command)))  
    inst->ipcHandle = ipc_handle  
    ;; ipc_handle will be nil on failure; non-nil on success  
    ipc_handle))
```

axlJPGUICustDiffer

```
axlJPGUICustDiffer(  
    g_inst  
    l_propList1  
    l_propList2  
)  
=> t / nil
```

Description

Job Policy GUI Customization member function that determines whether *l_propList1* and *l_propList2* differ. The Job Policy GUI uses this method to determine if the GUI settings differ from those already attached to the tool.

Arguments

<i>g_inst</i>	Instance of the class axlJPGUICust.
<i>l_propList1</i>	First DPL. Any properties stored in the DPL would have been written by a previous call to axlJPGUICustReadFrom Form.
<i>l_propList2</i>	Second DPL. Any properties stored in the DPL would have been written by a previous call to axlJPGUICustReadFrom Form.

Value Returned

t	Indicates that the <i>l_propList1</i> and <i>l_propList2</i> differ in a way that causes reapplication of the job policy
nil	Indicates an error.

Examples

```
;; example setup  
(defclass RemoteHost ( axlJPGUICust )  
  ()  
  (defmethod axlJPGUICustReadFromForm ((inst RemoteHost) form dataDpl)  
    (putprop dataDpl form->RemoteHostName->value 'remotehostname))  
  ;; differ method.  
  (defmethod axlJPGUICustDiffer ((inst RemoteHost) dp1 dp2 )  
    (nequal dp1->remotehostname dp2->remotehostname))
```

axlJPGUICustHiddenSections

```
axlJPGUICustHiddenSections(  
    g_inst  
)  
=> l_hiddenSections
```

Description

Job Policy GUI Customization member function that returns a list of sections or fields to be hidden from the Job Policy Setup form.

Note: This function allows hiding of the *Timeouts (in Secs.)* section only.

Arguments

<i>g_inst</i>	An instance of the class axlJPGUICust.

Value Returned

<i>l_hiddenSections</i>	List of sections to be hidden from the Job Policy Setup form is returned.

Examples

Specifies that the *Timeouts (in Secs.)* section is to be hidden from the Job Policy Setup form:

```
;; example setup  
(defclass RemoteHost ( axlJPGUICust )  
  ())  
  
(defmethod axlJPGUICustHiddenSections ((inst RemoteHost) )  
  '(timeout))
```

axlJPGUICustHIFields

```
axlJPGUICustHIFields(  
    g_inst  
    x_offset  
)  
=> l_fields
```

Description

Job Policy GUI Customization member function to create the HI field displayed for a particular JP GUI customization. The Job Policy GUI calls this method once during initialization. If no customizations are desired, the function does not need to be specialized.

Arguments

<i>g_inst</i>	Instance of the class axlJPGUICustHIFields.
<i>x_offset</i>	Initial form offset. Any HI fields created must be based on this offset

Value Returned

<i>l_fields</i>	List of HI form elements is returned.

Examples

Returns the list of HI form elements.

```
;; example setup  
(defclass RemoteHost ( axlJPGUICust )  
    ())  
;; hifields method  
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)  
    `((, (hiCreateStringField  
        ?name 'RemoteHostName  
        ?prompt "Host"  
        ?invisible nil)  
        (20 ,yoffset)  
        (370 30) 147))  
    )
```


axlJPGUICustOffset

```
axlJPGUICustOffset(  
    g_inst  
)  
=> x_offset
```

Description

Job policy GUI customization member function to return the *y* size of the HI field customizations provided with the `axlJPGUICustHIFields` method. The value is obtained by adding 10 to the *y* position of the last HI element. The Job Policy form uses this value as the *y* offset for the form elements underneath the customization area.

Arguments

<i>g_inst</i>	Instance of the class <code>axlJPGUICust</code>

Value Returned

<i>x_offset</i>	The offset is returned.

Examples

Returns the *y* size of the HI field customizations provided with the `axlJPGUICustHIFields` method.

```
;; example setup  
(defclass RemoteHost ( axlJPGUICust )  
    ())  
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)  
    `((, (hiCreateStringField  
        ?name 'RemoteHostName  
        ?prompt "Host"  
        ?invisible nil)  
        (20 ,yoffset)  
        (370 30) 147))  
    )  
;; offset method. 40 is the height of the stringfield (30) + 10  
(defmethod axlJPGUICustOffset ((inst RemoteHost))  
    40)
```

axlJPGUICustReadFromForm

```
axlJPGUICustReadFromForm(  
    g_inst  
    g_form  
    l_dataDpl  
)  
=> nil
```

Description

Job Policy GUI Customization member function to read any HI customization into a property list that will be saved as a job policy.

Arguments

<i>g_inst</i>	Instance of the class axlJPGUICust.
<i>g_form</i>	Form to be read. HI field customizations added by axlJPGUICustHIFields will be properties on the form.
<i>l_dataDpl</i>	Property list to modify.

Value Returned

None

Examples

```
;; example setup  
(defclass RemoteHost ( axlJPGUICust )  
  ())  
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)  
  `((, (hiCreateStringField  
      ?name 'RemoteHostName  
      ?prompt "Host"  
      ?invisible nil)  
    (20 ,yoffset)  
    (370 30) 147))  
  )  
;; readfromform method. This will store the GUI's value as the job policy  
property remotehostname  
(defmethod axlJPGUICustReadFromForm ((inst RemoteHost) form dataDpl)  
  (putprop dataDpl form->RemoteHostName->value 'remotehostname))
```

axlJPGUICustSelected

```
axlJPGUICustSelected(  
    g_inst  
    g_form  
    g_enabled  
)  
=> nil
```

Description

Job Policy GUI Customization member function to enable or disable any HI customizations. The Job Policy GUI calls this function every time the job policy type is changed.

Arguments

<i>g_inst</i>	Instance of the class axlJPGUICust.
<i>g_form</i>	Form to be read. HI field customizations added by axlJPGUICustHIFields are the properties on the form.
<i>g_enabled</i>	Boolean set if customization is being shown and nil if hidden

Value Returned

None

Examples

Enables the HI customizations.

```
:: example setup  
(defclass RemoteHost ( axlJPGUICust )  
    ())  
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)  
    `((, (hiCreateStringField  
        ?name 'RemoteHostName  
        ?prompt "Host"  
        ?invisible nil)  
        (20 ,yoffset)  
        (370 30) 147))  
    )  
:: selected method  
(defmethod axlJPGUICustSelected ((inst RemoteHost) form enabled)  
    form->RemoteHostName->invisible = !enabled  
    form->RemoteHostName->enabled = enabled)
```

axlRegisterJobIntfc

```
axlRegisterJobIntfc(  
    s_displayName  
    s_className  
    [ ?isInitializedFun u_isInitFun ]  
    [ ?displayInGUI g_shouldDisplay ]  
)  
=> t
```

Description

Registers a job interface class into the application. Job interfaces should be registered before use, preferably during virtuoso initialization.

Arguments

<i>s_displayName</i>	String name to store in the job policy and display in the Job Policy GUI.
<i>s_className</i>	Name of a derived class from <code>axlJobIntfc</code> .
<i>?isInitializedFun u_isInitFun</i>	Specifies a function to be called on initialization, before any jobs are submitted. The function shall accept two string arguments: the first is the <code>displayName</code>, the second, the <code>className</code>. It shall return <code>t</code> if initialization was succesful and <code>nil</code> otherwise. If initialization fails, the application will display an error and use the built in job policy. By default, the function only returns <code>t</code>.
<i>?displayInGUI g_shouldDisplay</i>	Boolean that switches whether the display name will be listed in the <i>Distribution Method</i> cyclic field on the Job Policy Setup form. It would be useful to pass <code>nil</code> in order to provide a policy-specific Job Policy GUI Customization (as registered with <code>axlRegisterJPGUICust</code>) which could be used to provide additional interface arguments.

Virtuoso ADE SKILL Reference

Job Policy Functions

Value Returned

t The job interface class is registered into the application.

Examples

Registers the job interface class into the application.

```
axlRegisterJobIntfc("Localhost IPC" '_axlIPCJobIntfc)  
=> t
```

axlJPGUICustWriteToForm

```
axlJPGUICustWriteToForm(  
    g_inst  
    g_form  
    l_dataDpl  
)  
=> nil
```

Description

Job Policy GUI Customization member function to load a property list that is saved as a job policy into HI customizations.

Arguments

<i>g_inst</i>	Instance of the class axlJPGUICust.
<i>g_form</i>	Form to be read. HI field customizations added by axlJPGUICustHIFields are the properties on the form.
<i>l_dataDpl</i>	Property list to read. The property names would have been saved by a previous call to axlJPCustReadFromForm.

Value Returned

None

Examples

Loads the property list that is saved as a job policy into HI customizations.

```
;; example setup  
(defclass RemoteHost ( axlJPGUICust )  
    ())  
(defmethod axlJPGUICustHIFields ((inst RemoteHost) yoffset)  
    `((, (hiCreateStringField  
        ?name 'RemoteHostName  
        ?prompt "Host"  
        ?invisible nil)  
      (20 ,yoffset)  
      (370 30) 147))  
    )  
;; readfromform method. This will set the GUI's value from the job policy  
property remotehostname
```

Virtuoso ADE SKILL Reference

Job Policy Functions

```
(defmethod axlJPGUICustWriteToForm ((inst RemoteHost) form dataDpl)
  form->RemoteHostName->value = dataDpl->remotehostname)
```

axlRegisteredJobIntfcNames

```
axlRegisteredJobIntfcNames(  
    )  
=> l_names / nil
```

Description

Retrieves the registered job interfaces.

Arguments

None

Value Returned

<i>l_names</i>	The list of registered names is returned.
<i>nil</i>	The list of registered names is not returned.

Examples

Retrieves the registered job interfaces.

```
axlRegisteredJobIntfcNames()  
=> ("my_interface")
```


axlRegisterJPGUICust

```
axlRegisteredJPGUICust (
    S_name
    g_inst
)
=> t
```

Description

Job Policy GUI Customization member function to register a customization into the Job Policy Setup form. Any registered customizations will appear the next time the job policy Setup form is displayed.

Arguments

<i>S_name</i>	String name for distribution method type.
<i>g_inst</i>	Instance of the class axlJPGUICust.

Value Returned

t	The customization is registered in the Job Policy Setup form.

Examples

Registers the customization in the Job Policy Setup form.

```
;; example setup
(defclass RemoteHost ( axlJPGUICust )
  ())
;; axlRegisterJPGUICust method
axlRegisterJPGUICust ("Remote Machine" makeInstance ('RemoteHost))
=> t
```

axlGetAttachedJobPolicy

```
axlGetAttachedJobPolicy(  
    [ t_sessionName ]  
    [ t_toolType ]  
    [ t_testName ]  
)  
=> l_jobPolicyProperties / nil
```

Description

Returns the current job policy attached to the setup or to the given test.

Note: If the specified job control mode for the session is LSCS, use the `maeGetJobPolicy` function to retrieve the attached job policy.

Arguments

<i>t_sessionName</i>	Returns the job policy attached to the session name. If not specified, the default job policy for the current session is returned.
<i>t_toolType</i>	Returns the policy associated with the distribution tool type. Valid Value: ICRP Default Value: ICRP
<i>t_testName</i>	Name of the test to which the job policy is attached.

Value Returned

<i>l_jobPolicyProperties</i>	Disembodied property list (DPL) of the properties for the current attached job policies is returned.
<i>nil</i>	The disembodied property list (DPL) is not returned for the properties of the current attached job policies.

Examples

Returns the job policy attached to the current session:

Virtuoso ADE SKILL Reference

Job Policy Functions

```
axlGetAttachedJobPolicy()  
=> (nil blockemail "1" configuretimeout "300" distributionmethod "Local"  
lingertimeout "300" maxjobs "1" name "Maestro Default" preemptivestart "1"  
runtimeout "-1" showerrorwhenretrying "1" showoutputlogerror "0"  
startmaxjobsimmed "1" starttimeout "300")
```

Returns the job policy attached to test AC in the current session:

```
;;  
axlGetAttachedJobPolicy(axlGetWindowSession() "ICRP" "AC")  
=>(nil blockemail "1" configuretimeout "200" distributionmethod "LBS"  
lingertimeout "300" maxjobs "5" name "mypolicy" preemptivestart "1"  
reconfigureimmediately "1" runtimeout "3600" showerrorwhenretrying  
"1" showoutputlogerror "0" startmaxjobsimmed "1" starttimeout "300" )
```

Related Topics

[axlAddJobPolicy](#)

[maeGetJobPolicy](#)

axlGetJobPolicy

```
axlGetJobPolicy(  
    t_jobPolicyName  
)  
=> l_jobPolicyProperties / nil
```

Description

Returns a disembodied property list containing property-value pairs for the job policy.

Arguments

t_jobPolicyName Job policy name.

Value Returned

l_jobPolicyProperties

Disembodied property list (DPL) of job policy properties is returned.

nil Indicates that the named job policy was not found.

Examples

Returns the property name-value list for the Maestro Default policy:

```
axlGetJobPolicy( "Maestro Default" )  
'( nil configuretimeout "300" distributionmethod "Local" maxjobs "1" runtimeout  
"3600" starttimeout "300" )  
axlGetJobPolicy( "default1" )  
=> nil
```

Related Topics

[axlAddJobPolicy](#)

axlGetJobPolicyTypes

```
axlGetJobPolicyTypes(  
    )  
=> l_jobPolicyNames / nil
```

Description

Returns a list containing names of all available job policies.

Arguments

None

Value Returned

l_jobPolicyNames

List containing names of all job policies in the setup is returned.

nil

Indicates that no job policy is available.

Examples

Returns a list containing names of all available job policies.

```
axlGetJobPolicyTypes( )  
=> '( "mypolicy" "default" )
```

Indicates that no job policy is available.

```
axlGetJobPolicyTypes( )  
=> nil
```

axlIsICRPPProcess

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

Description

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

Arguments

None

Value Returned

<code>t</code>	Indicates that the child IC remote process is running.
<code>nil</code>	Indicates that the child IC remote process is not running.

Examples

Specifies that the code is being run in a remote child process for the tool.

```
axlIsICRPPProcess( )  
=> t
```

axlSaveJobPolicy

```
axlSaveJobPolicy(  
    t_policyName  
    [t_targetDirectory]  
)  
=> t / nil
```

Description

Saves the policy given by the *t_policyName* argument.

Arguments

<i>t_policyName</i>	Job policy name.
<i>t_targetDirectory</i>	Directory in which the policy name will be saved. If you have specified a path, the tool saves the policy in the <code>jobpolicy</code> directory in that path. If the <code>jobpolicy</code> directory is not found, it given an error.

Value Returned

<i>t</i>	The job policy specified through the <i>t_policyName</i> argument is saved.
<i>nil</i>	Indicates an error in case the directory does not exist or you do not have write permission.

Examples

Save the specified job policy:

```
axlSaveJobPolicy("mypolicy" "../..policies/")  
=> t
```

axlSetJobPolicyProperty

```
axlSetJobPolicyProperty(  
    t_jobPolicyName  
    t_jobPropertyName  
    t_jobPropertyValue  
)  
=> t / nil
```

Description

Sets a property name-value pair for the specified job policy. You can use this function to update the properties of an existing policy. To apply the updated properties to all the sessions, set the updated policy as the default policy for the tool by using the `defaultJobPolicy` environment variable.

Arguments

<i>t_jobPolicyName</i>	Name of an existing job to which you want to add a new property.
<i>t_jobPropertyName</i>	Job policy property.
<i>t_jobPropertyValue</i>	Job policy property value.

Value Returned

<i>t</i>	The specified property name and value is set for the specified job policy.
<i>nil</i>	The specified property name and value is not set for the specified job policy.

Examples

Returns the job policy attached to test `Test1` and change the `configuretimeout` property for it:

```
;; get the job policy attached to Test1  
axlGetAttachedJobPolicy("session0" "ICRP" "Test1")  
(nil blockemail "1" configuretimeout "200"  
distributionmethod "LBS" lingertimeout "300" maxjobs  
"5" name "mypolicy" preemptivestart "1")
```


Virtuoso ADE SKILL Reference

Job Policy Functions

```
reconfigureimmediately "1" runtimeout "3600" showerrorwhenretrying "1"  
showoutputlogerror "0" startmaxjobsimmed "1" starttimeout "300" )  
;; change the configuretimeout property  
axlSetJobPolicyProperty( "mypolicy" "configuretimeout" "500" )  
=> t
```

Related Topics

[defaultJobPolicy](#)

axlStopAllJobs

```
axlStopAllJobs(  
    [ t_sessionName ]  
    [ g_forceFlag ]  
)  
=> t / nil
```

Description

Stops all jobs you started during the current or specified session, regardless of their state (started, getting configured, running)

Arguments

<i>t_sessionName</i>	Stops the jobs associated with the specified session. When not specified, the jobs associated with the current session are stopped.
<i>g_forceFlag</i>	Forcefully terminates the jobs.

Value Returned

<i>t</i>	All jobs you started during the current or specified session are stopped.
<i>nil</i>	Indicates an error.

Examples

Stops all the jobs.

```
axlStopAllJobs( )  
=> t
```

Stops only the jobs associated with the session named `session0`.

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

Forcefully terminates all the jobs.

```
axlStopAllJobs( t )  
=> t
```

Forcefully terminates the jobs associated with the session named `session0`.

Virtuoso ADE SKILL Reference

Job Policy Functions

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

axlStopJob

```
axlStopJob(  
    t_sessionName  
    x_jobId  
    [ g_forceFlag ]  
)  
=> t / nil
```

Description

Stops a job.

Arguments

<i>t_sessionName</i>	Name of the session
<i>x_jobId</i>	Job ID
<i>g_forceFlag</i>	Forcefully terminates the job. By default, this is set to <code>nil</code> and the tool does not stop a job if a simulation is running on it.

Value Returned

<code>t</code>	The specified job is stopped.
<code>nil</code>	The specified job is not stopped

Examples

Stops the job with the ID 1.

```
axlStopJob( 1 )  
=> t
```

Forcefully terminates the job with the ID 1.

```
axlStopJob( 1 t )  
=> t
```

maeClearAllTestJobPolicies

```
maeClearAllTestJobPolicies(  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Clears the test-level job policy setup for each test in the setup. The job policy setup at the global level is not removed.

Arguments

?session t_sessionName

Name of the session. When not specified, it uses the current session.

Value Returned

t The test-level job policy setup for each test in the setup is cleared.

nil The test-level job policy setup is not cleared for the tests.

Examples

Clears the test-level job policies for the current session.

```
maeClearAllTestJobPolicies()  
=> t
```

Clears the test-level job policies for the given session.

```
maeClearAllTestJobPolicies(?session "fnxSession0")  
=> t
```

maeClearTestJobPolicy

```
maeClearTestJobPolicy(  
    t_testName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Clears the job setup of a given test and applies the global job setup.

Arguments

<i>t_testName</i>	Name of the test for which the job policies are to be removed.
<i>?session t_sessionName</i>	Name of the session. When not specified, it uses the current session.

Value Returned

<i>t</i>	The the job setup of the given test is cleared and the global job setup is applied.
<i>nil</i>	The the job setup of the given test is not cleared and the global job setup is not applied.

Examples

Clear the test-level job setup for the test.

```
maeClearTestJobPolicy("AMSBC:Inv:1")  
=> t
```

maeCreateNetlistForCorner

```
maeCreateNetlistForCorner(  
    t_testName  
    t_cornerName  
    t_netlistDir  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Generates a netlist for the specified corner. In case the setup contains a corner sweep, the netlist will be created for the first sweep point of the specified corner.

Arguments

<i>t_testName</i>	Name of the test for which the corner netlist is to be created.
<i>t_cornerName</i>	Name of the specified corner.
<i>t_netlistDir</i>	Path of the netlist directory in which the netlist is to be created.
<i>?session t_session</i>	Name of the session. When not specified, it uses the current session.

Value Returned

<i>t</i>	Netlist is created.
<i>nil</i>	Netlist is not created.

Examples

Creates a netlist in the netlist directory, C2NetlistDir, for the corner C2 in the test, test1, for the specified session.

```
axlSession =maeOpenSetup("testLib" "testCell" "maestro")  
=> "fnxSession2"  
maeCreateNetlistForCorner("test1" "C2" "C2NetlistDir" ?session axlSession)  
=> t
```

Virtuoso ADE SKILL Reference

Job Policy Functions

maeGetAllJobPolicies

```
maeGetAllJobPolicies()  
=> l_jobPolicyNames / nil
```

Description

Returns the job policies of all the tests in the specified session.

Arguments

None

Value Returned

<i>l_jobPolicyNames</i>	List of the job policy names is returned.
<i>nil</i>	List of the job policy names is not returned.

Examples

Returns all the job policies for the session `fnxSession5`.

```
maeGetAllJobPolicies()  
=> ("Job Policy LPF" "Maestro Default" "Maestro1" "Netlisting Default")
```


maeGetJobControlMode

```
maeGetJobControlMode(  
    [ ?session t_sessionName ]  
)  
=> t_controlMode / nil
```

Description

Returns the job control mode currently set in the given session.

Arguments

?session t_sessionName

Name of the session. When not specified, it uses the current session.

Value Returned

t_controlMode

Returns "LSCS" or "ICRP" depending on the job control mode set in the given session.

nil

The provided session is invalid.

Examples

Returns the job control mode set in the given session.

```
maeGetJobControlMode()  
=> "LSCS"
```

maeGetJobPolicy

```
maeGetJobPolicy(  
    [ ?session t_sessionName ]  
    [ ?testName t_testName ]  
    [ ?jobType t_jobType ]  
)  
=> l_jobPolicyProperties / nil
```

Description

Returns the details of the job policy attached to the given test. If no test name is given, the function returns the policy attached to the current setup.

Arguments

?session t_sessionName

Name of an ADE Assembler session.

If not specified, the currently active session is used.

?testName t_testName

Name of the test.

If not specified, the job policy attached to the currently active setup is returned.

?jobType t_jobType

Job type for which you want to get policy details.

- "simulation": Jobs that run simulations
- "netlisting": Jobs that create netlists

Value Returned

l_jobPolicyProperties

A disembodied property list (DPL) of job policy properties is returned.

nil

A disembodied property list (DPL) of job policy properties is not returned.

Virtuoso ADE SKILL Reference

Job Policy Functions

Examples

Returns the default global job policy for the default job control mode, ICRP:

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeGetJobPolicy()
=>'( nil configuretimeout "300" distributionmethod "Local" maxjobs "1" runtimeout
"3600" starttimeout "300" )
```

Returns the job policies set for the simulation and netlisting jobs run for the LSCS job control mode, and make the required modification:

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")
=> "session0"
maeGetJobControlMode()
=> "ICRP"

maeSetJobControlMode("LSCS")
=> t
; sets the job control mode
;the following function returns the policy for simulation jobs
jpncl = maeGetJobPolicy(?jobType "netlisting")
=> (nil autoresume "0" blockemail "1"
configuretimeout "300" defaultcpuvalue "1" defaultmemoryvalue "1000"
distributionmethod "Local" estimatememoryvalue "" estimationsimulationmode "0"
lingertimeout "300" maxjobs "4" name "Job Policy LPF" preemptivestart "1"
providecpuandmemorydata "1" reconfigureimmediately "0" runpointsvalue "5"
runtimeout "-1" scaleestimatedbycpu "100" scaleestimatedbymemory "100"
showerrorwhenretrying "1" showoutputlogerror "0" startmaxjobsimmed "1"
starttimeout "300" suspenddisklow "0" thresholdvalue "100" usesameprocess "1"
warndisklow "0" warnthresholdvalue "100" )

;the following function returns the policy for simulation jobs
jp = maeGetJobPolicy(?jobType "simulation")
=> (nil autoresume "0" blockemail "1" configuretimeout "300" defaultcpuvalue "1"
defaultmemoryvalue "1000" distributionmethod "Local" estimatememoryvalue ""
estimationsimulationmode "0" lingertimeout "300" maxjobs "2" name "Maestro Default"
preemptivestart "1" providecpuandmemorydata "1" reconfigureimmediately "0"
runpointsvalue "5" runtimeout "-1" scaleestimatedbycpu "100"
scaleestimatedbymemory "100" showerrorwhenretrying "1" showoutputlogerror "0"
startmaxjobsimmed "1" starttimeout "300" suspenddisklow "0" thresholdvalue "100"
usesameprocess "1" warndisklow "0" warnthresholdvalue "100" )

;change Max Jobs for simulation jobs
jp->maxjobs=10
=>10

maeSetJobPolicy(jp ?jobType "simulation")
=> t
jp = maeGetJobPolicy(?jobType "simulation")
=> (nil autoresume "0" blockemail "1" configuretimeout "300" defaultcpuvalue "1"
defaultmemoryvalue "1000" distributionmethod "Local" estimatememoryvalue ""
estimationsimulationmode "0" lingertimeout "300" maxjobs "10" name "Maestro
Default" preemptivestart "1" providecpuandmemorydata "1" reconfigureimmediately
"0" runpointsvalue "5" runtimeout "-1" scaleestimatedbycpu "100"
scaleestimatedbymemory "100" showerrorwhenretrying "1" showoutputlogerror "0"
startmaxjobsimmed "1" starttimeout "300" suspenddisklow "0" thresholdvalue "100"
usesameprocess "1" warndisklow "0" warnthresholdvalue "100" )
```

Virtuoso ADE SKILL Reference

Job Policy Functions

Returns the job policy for a particular test in the setup:

```
maeGetJobPolicy(?testName "test1")  
=>'( nil configuretimeout "100" distributionmethod "Local" maxjobs "1" runtimeout  
"3600" starttimeout "300" )
```

maeGetJobPolicyByName

```
maeGetJobPolicyByName(  
    t_policyName  
    [ ?session t_sessionName ]  
)  
=> l_jobPolicyProperties / nil
```

Description

Returns a disembodied property list containing property-value pairs for the given job policy.

Arguments

<i>t_policyName</i>	Name of the job policy
<i>?session t_sessionName</i>	Name of an ADE Assembler session
	If not specified, the currently active session is used.

Value Returned

<i>l_jobPolicyProperties</i>	A disembodied property list (DPL) of job policy properties is returned.
<i>nil</i>	The specified job policy is not found.

Examples

Returns the details of a specific job policy:

```
sess= maeOpenSetup("solutions" "ampTest" "maestro")  
=> "session0"  
  
maeGetJobPolicyByName("myPolicy")  
=>'( nil configuretimeout "300" distributionmethod "Local" maxjobs "1" runtimeout  
"3600" starttimeout "300" )
```

maeHasTestJobPolicy

```
maeHasTestJobPolicy(  
    t_testName  
    [ ?session t_sessionName ]  
)  
=> t / nil
```

Description

Checks if the given test has a test-specific job policy setup.

Arguments

<i>t_testName</i>	Name of the specified test.
<i>?session t_sessionName</i>	Name of the session. When not specified, it uses the current session.

Value Returned

<i>t</i>	The given test has a test-specific job policy setup.
<i>nil</i>	The given test does not have a test-specific job policy setup.

Examples

Checks if a given test has a test-specific job policy setup:

```
maeHasTestJobPolicy("AMSBC:Inv:1")  
=> t
```

maeIsEvaluatorProcess

```
maeIsEvaluatorProcess(  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Returns `t` if currently the expression evaluator service process is running for ADE Assembler or ADE Explorer. You can use this function in your `.cdsinit` file or in custom SKILL code.

Arguments

None

Value Returned

<code>t</code>	Indicates that the expression evaluator service process is running.
<code>nil</code>	Indicates that the the expression evaluator service process is not running.

Examples

Indicates that the the expression evaluator service process is not running.

```
maeIsEvaluatorProcess( )  
=> nil
```

maelsNetlistProcess

```
maeIsNetlistProcess(  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Returns `t` if currently the netlister service process is running for ADE Assembler or ADE Explorer. You can use this function in your `.cdsinit` file or in custom SKILL code.

Arguments

None

Value Returned

<code>t</code>	Indicates that the netlister service process is running.
<code>nil</code>	Indicates that the netlister service process is not running.

Examples

Indicates that the netlister service process is running.

```
maeIsNetlistProcess( )  
=> t
```


maeSetJobControlMode

```
maeSetJobControlMode
    t_mode(
    [ ?session t_sessionName ]
    )
=> t / nil
```

Description

Sets the job control mode in the given session.

Arguments

<i>t_mode</i>	The job control mode to be set.
<i>?session t_sessionName</i>	Name of the session. When not specified, it uses the current session.

Value Returned

<i>t</i>	The job control mode is set.
<i>nil</i>	The job control mode is not set.

Examples

Sets the specified job control mode for the current session

```
maeSetJobControlMode("ICRP")
=>t
```

Sets the specified job control mode for the given session

```
maeSetJobControlMode("ICRP" ?session "fnxSession0")
=>t
```

maeSetJobPolicy

```
maeSetJobPolicy (
    g_jobPolicyDPL
    [ ?testName t_testName ]
    [ ?jobType t_jobType ]
    [ ?session t_sessionName ]
)
=> t / nil
```

Description

Sets the given job policy to the specified setup for the given test.

Arguments

<i>g_jobPolicyDPL</i>	A distributed property list that can provide values for multiple properties of a job policy.
<i>?testName t_testName</i>	Name of the test for which this policy needs to be set. The default value is "Global".
<i>?jobType t_jobType</i>	Type of job for which the policy is to be set. ■ "simulation": (Default) Sets the job policy for the simulation jobs ■ "netlisting": Sets the job policy for the netlisting jobs
<i>?session t_sessionName</i>	Name of the session. If not specified, the current session is used.

Value Returned

<i>t</i>	The given job policy is set for the specified setup for the given test.
<i>nil</i>	The given job policy is not set for the specified setup.

Virtuoso ADE SKILL Reference

Job Policy Functions

Examples

Updates the default global job policy in the current maestro session to change the value of the `pointlifetime` property:

```
jp = maeGetJobPolicy()  
jp->pointlifetime=1  
maeSetJobPolicy(jp)
```

Updates the netlisting job policy for test test1:

```
jp = maeGetJobPolicy(?jobType "netlisting" ?testName "test1")  
jp->pointlifetime=1  
maeSetJobPolicy(jp)
```

maeStopAllJobs

```
maeStopAllJobs(  
  [ ?session t_sessionName ]  
  [ ?jobType t_jobType ]  
  [ ?force g_force ]  
  [ ?exitCode t_exitCode ]  
)  
=> t / nil
```

Description

Stops all the simulation or netlisting jobs you started during the current session regardless of their state (started, getting configured, running).

Arguments

`?session t_sessionName`

Stops the jobs associated with the specified session.

When not specified, the jobs associated with the current session are stopped.

`?jobType t_jobType` The type of the job to be stopped.

Possible values:

- "simulation"
- "netlisting"

When no value is specified, both types of jobs are stopped.

`?force g_force`

If set to `t`, kills the job by sending a signal, such as `SIGKILL`. Otherwise, it kills the job by using an internal communication mechanism.

`?exitCode t_exitCode`

Sets the specified exit code for the stopped jobs.

This argument can be used only when the `?force` argument is set to `nil`.

Virtuoso ADE SKILL Reference

Job Policy Functions

Value Returned

<code>t</code>	The simulation or netlisting jobs you started during the current session are stopped regardless of their state.
<code>nil</code>	Indicates that the specified session or job type is invalid.

Examples

Stops all the jobs for the current session.

```
maeStopAllJobs()  
=>t
```

Stops all netlister service jobs for the current session.

```
maeStopAllJobs("netlisting")  
=>t
```

Stops all jobs by sending a SIGKILL signal.

```
maeStopAllJobs(?force t)  
=>t
```

Stops all jobs and returns the exit code 4.

```
maeStopAllJobs(?exitCode 4)  
=>t
```

maeStopJob

```
maeStopJob(  
    [ ?session t_sessionName ]  
    [ ?jobType t_jobType ]  
    [ ?force g_force ]  
    [ ?exitCode t_exitCode ]  
)  
=> t / nil
```

Description

Stops the job for the specified job ID regardless of its state.

Arguments

<code>?jobType t_jobType</code>	The ID of the job to be stopped.
<code>?force g_force</code>	If set to <code>t</code> , kills the job by sending a signal, such as <code>SIGKILL</code> . Otherwise, it kills the job by using an internal communication mechanism.

Value Returned

<code>t</code>	The job is stopped for the specified job ID regardless of its state.
<code>nil</code>	The job is not stopped for the specified job ID regardless of its state.

Examples

Stops the job for the specified job ID.

```
maeStopJob(3)  
=>t
```

Stops the job for the specified job ID by sending a `SIGKILL` signal.

```
maeStopJob(5 ?force t)  
=>t
```

Virtuoso ADE SKILL Reference

Job Policy Functions

Core Netlister Functions

You can use the core netlister functions to manage netlisters. Core netlister functions can be categorized into the following categories:

- [Initialization Functions](#)
- [Netlisting Invocation Functions for Direct Integration](#)
- [Netlist Functions](#)
- [Netlisting Option Functions for Socket Interfaces](#)
- [Mixed-Signal Simulation Functions](#)

Initialization Functions

Initialization functions let you initialize the simulation environment for your simulator and register your simulator.

These functions can be further organized according to their purposes as described below.

- Tools and Sessions: Functions that let you change the values of existing variables, such as the default values of model paths or simulator options.
- Functions that help you integrate a simulator. An integrator is responsible for making the changes necessary to include a circuit (You need to purchase the OASIS product in order to integrate a simulator.)
- Workflow for Adding Features to Simulators: Functions that let you add new features to existing simulators.
- Changing Banner Menus: Functions that let you change the menus for the Simulation window.

Cadence has integrated simulators into the analog design environment by using the direct simulation approach. With this approach, the netlist uses the syntax of the simulator you are using, without any processing to evaluate expressions. The passed parameters, design variables, functions, and other inputs are all resolved by the simulator. The netlist is a direct reflection of the design.

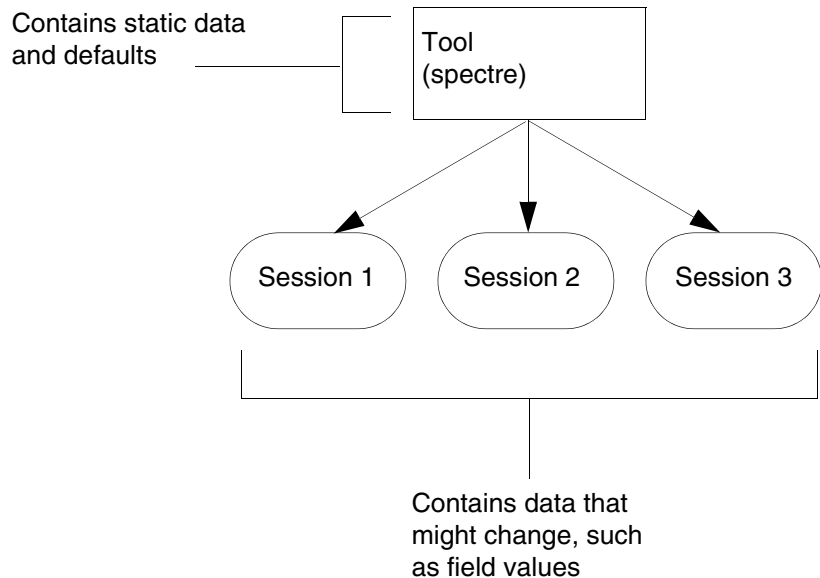
Tools and Sessions

Understanding the difference between a tool object and a session object will help you use many of the functions in this manual. A tool object is a data structure that contains all the default and static data for a simulator. For example, the fields on an analysis form are static (a transient analysis always has a *From*, *To*, and *By* field). A *session* object contains data that might change between simulator uses, such as the values of the *From*, *To*, and *By* fields, or design variables or model paths.

Virtuoso ADE SKILL Reference

Core Netlister Functions

Several sessions can be associated with one tool. This means you can start several Cadence sessions, each with different transient analysis settings. However, all these sessions are associated with one tool (`spectre`).



Most of the functions take the `tool` argument. Some take either the `tool` or the `session` argument, and some take only the `session` argument. For example, `asiSetEnvOptionVal` takes the following arguments:

```
asiSetEnvOptionVal( {o_tool | o_session} s_name g_value)
```

This means you can pass either a tool or a session object. If you pass a tool object, you are modifying the tool and every session created after this. If you pass a session object, your changes affect the current session only.

Examples

Consider Spectre tool with the model path set to `~/models`, and a schematic window open with your design. When you open a Simulation window, which is set to use the `spectre` simulator by default, you are starting a `spectre` session. If you bring up the Environment Options form, the model path is set to `~/models`. You can change the model path for the tool by typing the following in the CIW:

```
asiSetEnvOptionVal( asiGetTool('spectre') 'modelPath  
"~/processA/best/models")
```

Now, if you check the model path from your existing `spectre` simulation window, it is still set to `~/models`. This is because this session was previously created based on the original tool. However, if you open a second `spectre` simulation window, a new session is created based

Virtuoso ADE SKILL Reference

Core Netlist Functions

on the modified `spectre` tool. In this session, the model path is set to `~/processA/best/models`.

If you do not want to open another session, but you want to change the model path in the current session, you might type the following command in the CIW:

```
asiSetEnvOptionVal( asiGetCurrentSession() 'modelPath  
"~/processA/best/models")
```

This changes the model path in the existing session only. If you open another session, the `modelPath` value is still set to `~/models`. This is because the command did not change the default value of the model path, it changed the session value only.

You can think of a tool as the template from which sessions are created. Changing a tool affects all sessions created after the tool is changed, but does not affect existing sessions.

Note: The core netlist functions are contained primarily in the `oasis.cxt` and `analog.cxt` files. You must load these context files before using the functions. Bringing up the Simulation window automatically loads these contexts.

Use Models for Functions

There are two common use models for the functions that take the tool and session object.

Working with Tools

Often you need to modify an item or option related to a tool. These types of operations might include the following:

- Adding an option to a tool
- Changing a tool option
- Deleting a tool option
- Setting a tool option's value
- Getting a tool option (object)

You can set up a function in your `.cdsinit` file to do this. For example, suppose you wanted to set the following options for every new session.

- Model path set to `~/processA/best/models`
- Include file set to `~/processA/best/models/include`

You might add the following functions to your `.cdsinit` file:

Virtuoso ADE SKILL Reference

Core Netlister Functions

```
asiSetEnvOptionVal( asiGetTool('Spectre) 'modelPath  
"~/processA/best/models")  
asiSetEnvOptionVal( asiGetTool('Spectre) 'includeFile  
"~/processA/best/models/include")
```

Because these calls to `asiSetEnvOptionVal` are in the `.cdsinit` (before any sessions are opened), all subsequent sessions contain the specified model path and include file.

Getting Values from Sessions

Sometimes, you want to get a value from an existing session.

For example, when writing a custom SKILL code, you need to get the value of the transient to field for the existing session. In this case, you might use the following code:

```
session = asiGetCurrentSession()  
analysis = asiGetAnalysis(session 'tran)  
valueOfTo=asiGetAnalysisFieldVal( analysis 'to)
```

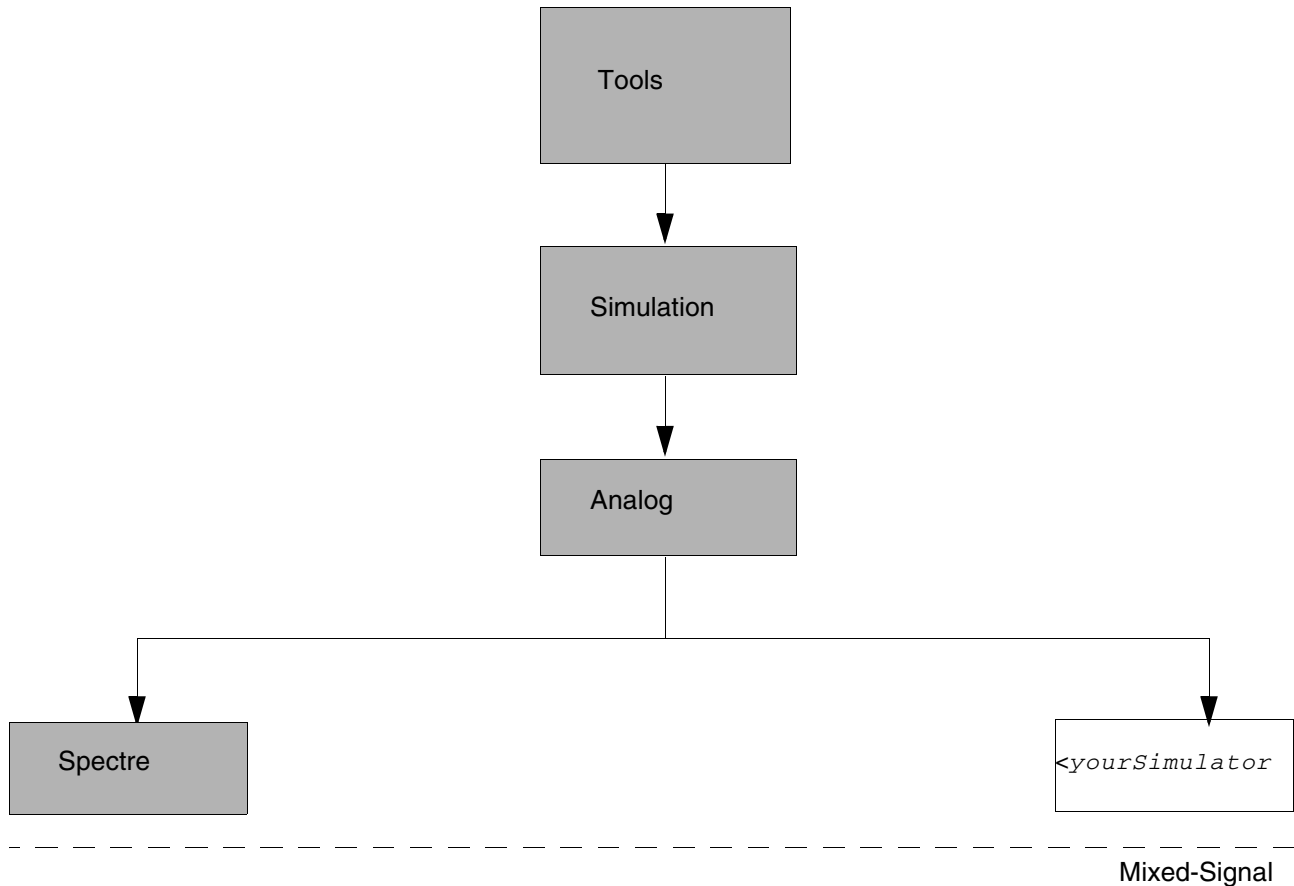
Class Structures

In Virtuoso ADE, much of the underlying code utilizes classes. A class determines the structure and behavior of its instances, which are known as objects. A class inherits information from its parent class (or super class) and passes information to its subclass. Understanding classes will help you use some of the functions that utilize the inheritance mechanism. These functions are called methods.

Virtuoso ADE SKILL Reference

Core Netlister Functions

The following figure shows how the classes are organized.



The class structure is organized as follows:

- The `Tools` class is the base or root class. Anything declared in the `Tools` class is inherited by all the other subclasses.
- The `Simulation` class inherits everything from the `Tools` class. `Simulation` is the base class for all simulators, and anything declared in the `Simulation` class is inherited by all the simulators.
- The `Analog` class inherits everything from the `Simulation` class. The `Analog` class adds all the generic analog simulation features to the set. Anything declared in the `Analog` class is inherited by all the analog simulators.

If an inherited feature from any level is not applicable to a lower level, you can delete or change the feature at the lower level.

Methods for Integrators

Some of the functions are defined as methods for integrators. A method is a function that can be overloaded. In other words, the same function can be defined in different ways depending on the class of the arguments that users pass to the function. With methods, the same function name can specify a general class of actions. The user, by supplying an initial argument of a particular class to the function, determines the specific action of the function. (The SKILL++ object system analyzes the class of the initial argument to determine which variant of the function to call.) The methods have examples for integrators.

When using methods, the space after the name of the procedure is required. (When using SKILL procedures, the space after the name of the procedure is not allowed.)

This space is required when using
methods.



```
defmethod(asiSendOptions ( (session analog_session) )  
    println("asiSendOptions for the analog class")  
)
```

You can use `defmethod` to create a method that is specialized for a particular class. If a user calls a function with an initial argument of this class, the SKILL++ object system looks for the corresponding method for that class. If there isn't a match for that class, the SKILL++ object system looks for a match in the class above, and so on.

Using methods gives you the following advantages:

- You can overload an existing method for your own simulator-specific classes. In other words, you can customize an existing function for your simulator.
- You can use the `callNextMethod` procedure to use an inherited method and add some code before or after it.

Related Topics

[Cadence SKILL++ Object System Reference.](#)

Overloading Methods

You can overload an existing method for your own simulator-specific classes. For example, there is a method to send simulation options (`asiSendOptions`) in the Analog Class. You probably need to create a different method to send your options. To do this, create your own method named `asiSendOptions` that is used by your simulator. You do not have to modify

Virtuoso ADE SKILL Reference

Core Netlist Functions

the function that calls `asiSendOptions`. Your procedure takes precedence over the method in the Analog Class.

For example, the following is an Analog Class method declaration to send options:

This space is required when using
methods.



```
defmethod(asiSendOptions ( (session analog_session) )
    println("asiSendOptions for the analog class")
)
```

To create your own `asiSendOptions` procedure, use the following declaration:

```
defmethod(asiSendOptions ( (session <yourSimulator>_session) )
    println("asiSendOptions for your simulator class")
)
```

When the tool sends the simulation options for your simulator, your `asiSendOptions` method is called because the first argument is of the class `<yourSimulator>_session`.

Note: You should replace `<yourSimulator>` with the name of your simulator. Do not include the angle brackets (`<>`).

Customizing an Inherited Method for Your Simulator

The `callNextMethod` procedure lets you use an inherited method and add some code before or after it.

For example, suppose you need to modify the transient analysis to add transient options for your simulator. You need to write a method to send the information to Cadence SPICE. There is already a method defined to send a transient analysis to Cadence SPICE. Because there are no transient options in the default analysis, the code does not send options. So you can use `callNextMethod()` to call the method to format the analysis, then call your procedure to format the options.

```
defmethod(asiFormatAnalysis (
    (analysis <yourSimulator>_tran_analysis) fp)
    callNextMethod()
    asiFormatAnalysisOption(analysis fp)
    t
)
```

In this example, `callNextMethod` calls the analog `asiFormatAnalysis` method to format the transient analysis and send it to Cadence SPICE. Then your `asiFormatAnalysis` method calls `asiFormatAnalysisOption` to format and send the options.

Related Topics

[Cadence SKILL++ Object System Reference.](#)

Workflow for Adding Features to Simulators

You can use certain functions to add new features to existing simulators. Create a SKILL file to hold your information.

1. Use the `asiGetTool` function to get the tool object associated with the simulator you want to modify.
2. Add the features you want by using the procedures and passing in the `tool` argument.

For example, you might use `asiAddEnvOption` to add an environment option or `asiAddSimOption` to add a simulator option.

3. If needed, add the code to send your option to Cadence SPICE (to send to your target simulator).
4. If needed, add the code to invalidate the flowchart step if the value of your option changes.
5. Load your code changes into your `.cdsinit` file.

Later, you can add your changes to your site `.cdsinit` file.

6. Add any necessary changes to the `.cdsenv` file.

- ❑ Type `virtuoso` in a UNIX shell.
- ❑ Type the following command in the CIW and substitute the appropriate simulator name for `<simulator>`. Do not include the angle brackets (`<>`).

```
asiCreateCdsenvFile('<simulator>')
```

A file called `<simulator>CdsenvFile` is created in your current working directory.

- ❑ Use this file to replace the existing `.cdsenv` file in the `<install_dir>/tools/dfII/etc/tools/<simulator>` directory. Be sure to save a backup copy of the existing `.cdsenv` file before completing this step. Use the following command to replace the existing `.cdsenv` file with your new file:

```
mv <simulator>CdsenvFile <install_dir>/tools/dfII/etc/  
tools/<simulator>/.cdsenv
```

Examples

The following example shows how you might add an environment option called `myFile` to the environment options form for Spectre. The steps from the previous procedure are called out in the comments.

Virtuoso ADE SKILL Reference

Core Netlist Functions

```
; get the Spectre tool
tool=asiGetTool('Spectre) ; This is step 1.
; add the environment option myFile
asiAddEnvOption( tool          ; This is step 2.
    ?name                'myFile
    ?type                'fileName
    ?prompt              "My File"
    ?value               ""
    ?invalidateFunc      'asiInvalidateControlStmts
                        ; This is step 4. See the note
                        ; following this example.
)

; modify the step that sends control statements, to send myFile also.
flowchart = asiGetFlowchart( tool ) ; This is step 3.
asiChangeFlowchartStep( flowchart
    ?name                'asiSendControlStmts
    ?postFunc            'mySendMyFile
)

; Write the function (mySendMyFile) to send (the contents of) myFile to Spectre.
defmethod( mySendMyFile ( ( session Spectre_session ) )
; This is also step 3.
    let( ( netlistDir customInclude customIncludeFile )

        ; Check if the option is set.
        if( nequal( asiGetEnvOptionVal( session 'myFile) "" ) then

            ; There is a mechanism in Spectre that allows you to
            ' create a file
            ; called: .customInclude in the netlist directory. If you
            ; then send
            ; a 'ptptop' command to Spectre, it includes the contents of
            ; .customInclude in the final netlist for your target
            ; simulator.

            ; Open the .customInclude file in the netlist directory.
            netlistDir = asiGetNetlistDir(session)
            when( rexMatchp( "Verilog" asiGetSimName( session ) )
                netlistDir = strcat( netlistDir "/analog" )
            )
            customInclude=strcat(netlistDir ".customInclude")
            customIncludeFile = outfile( customInclude "w")

            ; Add whatever you need to add to this file.
            ; In this case, perhaps it is the contents of 'myFile'.

            ; When you are finished, close the file.
            close( customIncludeFile )

            ; Send the 'ptprop' command to Spectre to include the
            ; contents of this file in the final netlist.
            asiSendSim( session "ptprop analog CustomIncludeFile 1" nil
                nil nil )
        else
            ; Send the command to turn off the inclusion of the file in
            ; the final netlist.
            asiSendSim( session "deprop analog CustomIncludeFile" nil
                nil nil )
        )
    )
t
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

```
)  
)
```

Note: `asiInvalidateControlStmts` is a wrapper to `asiInvalidateFlowchartStep`, which invalidates the `asiSendControlStmts` step (as shown below):

```
defmethod( asiInvalidateControlStmts ( ( session Spectre_session ) )  
asiInvalidateFlowchartStep( session 'asiSendControlStmts' )  
)
```

Changing Banner Menus

You can change the banner menus in the Simulation window by creating a copy of the `simui.menus` file and make required modifications required in it. The `simui.menus` file is located at

`<install_dir>/tools/dfII/etc/tools/menus/simui.menus`

The beginning of this file explains the syntax for menu definitions.

Place a copy of the file at the following location:

`<install_dir>/tools/dfII/local/menus/simui.menus`

This location is for site customization. You can also put `menus/simui.menus` in

- The work area (if there is any) or the current working directory
- The project area (if TDM is in use)
- Your home directory.

Note: You must create the `menus` directory to hold the `simui.menus` file.

Add the definition for the appropriate callback function and associated code to the appropriate SKILL files.

You can also add menus by using the SKILL `hi` calls.

Changing Banner Menus for a Particular Simulator

You can also customize the banner menus (in the Simulation window) for a particular simulator by modifying the `<simulator>.menus` file. Look for the file in the following location.

```
<install_dir>/tools/dfII/etc/tools/menus/<simulator>.menus
```

If the file you need is here, make a copy of the file and make any modifications you want. If the file you need is not here, you can use the `Spectre` file (or another simulator file) as an example. Rename this file for your simulator and make any changes you want. (Do not include the angle brackets (`<>`) in the file name.)

Note: The `<simulator>.menus` file can be for a Cadence simulator or a non-Cadence simulator.

Place a copy of the file in the following location.

```
<install_dir>/tools/dfII/local/menus/<simulator>.menus
```

This location is for site customization. You can also put `menus/<simulator>.menus` in

- The work area (if there is any) or the current working directory
- The project area (if TDM is in use)
- Your home directory

Add the definition for the appropriate callback function and associated code to your appropriate SKILL files.

You can also add menus by using the SKILL `hi` calls.

asiInit<yourSimulator>

```
asiInit<yourSimulator>(
    o_tool
)
=> t
```

Description

Calls the procedures to initialize your simulator's environment. This function must be defined for socket interfaces. Do not use it for direct interfaces.

You must write `asiInit<yourSimulator>`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	The simulator's environment is initialized.

Examples

Creates the procedure that initializes the simulator environment for the XYZ simulator.

```
procedure ( asiInitXYZ( tool )
    <insert your code>
)
=> t
```

asiRegisterTool

```
asiRegisterTool(  
    '<simulatorName>  
    [ ?class      s_className ]  
    [ ?private    s_private ]  
    [ ?initFunc   s_initFunc ]  
    [ ?mixedSig   s_mixedSig ]  
    )  
=> t
```

Description

Registers your simulator and your initialization function.

Replace *<yourSimulator>* with the name of your simulator. Do not include the angle brackets *<>*.

Arguments

<simulatorName> Name of your simulator preceded by an apostrophe.

?class s_className

Specifies the class from which your simulator inherits information.

Default value: 'analog.

?private s_private

Declares a simulator as “private”, which means it does not appear in the list of simulators in the UI.

Valid Values are:

- *t* indicates that the option does not appear in the list of simulators in the UI
- *nil* indicates that the option appears in the list of simulators in the UI
The default value is *nil*.

You can also use this argument to create a parent class from which other classes inherit information. Designers cannot see or directly use a parent class if *private* is set to *t*.

?initFunc s_initFunc

Virtuoso ADE SKILL Reference

Core Netlister Functions

Initialization function for your simulator.

Default value: `asiInit<yourSimulator>`, where `<yourSimulator>` is the name of your simulator.

`?mixedSig s_mixedSig`

Set this argument to non-nil to specify a mixed-signal simulator.

Default value: `nil`

Value Returned

`t` The simulator is registered.

Examples

Registers the XYZ simulator under `analog` class.

```
asiRegisterTool('XYZ')
=> t
```


asiInitDataAccessFunction

```
asiInitDataAccessFunction(  
    o_tool  
)  
=> t / nil
```

Description

Initializes the data access function for the tool. This function can be used by a third-party integrator to define their own data access functions.

Arguments

<i>o_tool</i>	Simulation tool object.

Value Returned

<i>t</i>	The data access function is initialized.
<i>nil</i>	The function was not run successfully.

Examples

Initializes the data access function for the specified simulation tool object.

```
(defmethod asiInitDataAccessFunction ( ( tool <yourSimulator> ) )  
    asiDefineDataAccessFunction( tool 'VT '<yourSimulator>VT))  
procedure( '<yourSimulator>VT(specifier dataDir simData)  
    asiGetDrlData('tran specifier dataDir))  
=> t
```

asiInitEnvOption

```
asiInitEnvOption(  
    o_tool  
)  
=> t / nil
```

Description

Initializes the tool-specific environment options for the tools that are derived from the `asiAnalog` class. This is not applicable for tools derived from the `asiSocket` class. This function can be used by third-party integrators to define their own environment options.

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	The tool-specific environment options are initialized.
<code>nil</code>	The function did not run successfully.

Examples

Initializes the tool-specific environment options for the tools that are derived from the `asiAnalog` class.

```
defmethod( asiInitEnvOption ( ( tool <yourSimulator> ) )  
    ;; Initialize the environment options from the base class.  
    callNextMethod()  
    asiAddEnvOption( tool  
        ?name 'MyOpt  
        ?prompt "Any String Option"  
        ?value "xyz"  
        ?type 'string  
    )  
)  
=> t
```

asiInitAnalysis

```
asiInitAnalysis(  
    o_tool  
)  
=> t / nil
```

Description

Initializes the tool-specific analysis options for the tools that are derived from the `asiAnalog` class. This is not applicable for tools derived from the `asiSocket` class. This function can be used by third-party integrators to define their own analysis options.

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	The tool-specific analysis options are initialized.
<code>nil</code>	The function does not run successfully.

Examples

Initializes the tool-specific analysis options for the given tool:

```
defmethod( asiInitAnalysis ( ( tool <yourSimulator> ) )  
    asiAddAnalysis( tool  
        ?name 'myAnal  
        ?prompt "My Analysis"  
    )  
)  
=> t
```

asiInitAdvAnalysis

```
asiInitAdvAnalysis(  
    o_tool  
)  
=> t / nil
```

Description

Initializes the tool-specific analysis options for the tools derived from the `asiAnalog` class. This method can be used by third-party integrators to define their own analysis options in ADE XL.

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	The tool-specific analysis options are initialized.
<code>nil</code>	The tool-specific analysis options are not initialized

Examples

Initializes the tool-specific analysis options for the given tool:

```
defmethod( asiInitAdvAnalysis (( tool <yourSimulator> ) )  
    asiAddAnalysis( tool  
        ?name 'myAdvancedAnalysis  
        ?prompt "My Advanced Analysis"  
    )  
)  
=> t
```

asiInitSimOption

```
asiInitSimOption(  
    o_tool  
)  
=> t / nil
```

Description

Initializes the simulation options for the tools that are derived from the `asiAnalog` class. This is not applicable for tools derived from the `asiSocket` class. This function can be used by third-party integrators to define their own simulation options.

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	The tool-specific simulation options are initialized.
<code>nil</code>	The function does not run successfully.

Examples

The following example initializes simulation options as per the specified settings:

```
(defmethod asiInitSimOption ( ( tool <yourSimulator> ) )  
    asiAddSimOption( tool  
        ?name 'mySimOption  
        ?prompt "My Simulation Option"  
    )  
)  
=> t
```

asiGetPageCallBack

```
asiGetPageCallBack(  
    o_obj  
)  
=> s_func / nil
```

Description

Returns the callback function name when changing tabs in a form.

Arguments

<i>o_obj</i>	The form object.

Value Returned

<i>s_func</i>	The callback function name.

Examples

The following example returns the callback function name when changing tabs in the given form:

```
asiGetPageCallBack(formObj)  
=> t
```

asiSetPageCallBack

```
asiSetPageCallBack(  
    o_obj  
    s_func  
)  
=> t/ nil
```

Description

Returns the status (whether successful or failed) of the specified callback function, when changing tabs in a form.

Arguments

<i>o_obj</i>	The form object.
<i>s_func</i>	The callback function name.

Value Returned

<i>t</i>	The callback function is set.
<i>nil</i>	Setting the callback function failed.

Examples

Returns the status of the specified callback function for the given form:

```
asiSetPageCallBack(formObj, 'callbackFunctionName)  
=> t
```

Netlisting Invocation Functions for Direct Integration

Use the netlisting invocation functions to invoke netlisting options for direct integration.

The `si` flow supports all OSS-based backend netlisters and the ADE netlisters `spectre` and `Ultrasm`. The `si` flow does not work for any other simulator netlister integrated in ADE.

To use the `si` executable for the ADE netlisters, copy the `si.env` file into a new directory and execute the following command:

```
si -batch -cdslib <complete path to cds.lib>
```

This command generates the netlist as well as runs the simulation. Alternatively, use either the following command or the SKILL function `nl` to only generate the netlist:

```
si -batch -command nl
```

Important

The `-command netlist` option of the `si` command does not work for ADE netlisters and generates an error. Therefore, use the `-command nl` option as depicted above.

Note: The variable `nlFormatterClass` is used in addition to the usual OSS variables. It represents the symbolic name of the formatter class.

List of netlisting invocation functions:

[asiGetNetlistFormatterClass](#)

[asiSetNetlistFormatterClass](#)

[asiCreateFormatter](#)

[asiCreateCdsenvFile](#)

[asiGetFormatter](#)

[asiGetSimInputFileName](#)

[asiGetSimInputFileSuffix](#)

asiGetNetlistFormatterClass

```
asiGetNetlistFormatterClass(  
    o_tool  
)  
=> s_class / nil
```

Description

Returns the netlist formatter class for the specified tool.

Arguments

<i>o_tool</i>	The object representing the simulator interface.

Value Returned

<i>s_class</i>	A symbol representing the formatter class.
<i>nil</i>	Indicates an error.

Examples

The following example returns the netlist formatter class for `tool_name`:

```
tool_name = asiGetTool( session )  
asiGetNetlistFormatterClass( <tool_name> )  
=> spectreFormatter
```

asiSetNetlistFormatterClass

```
asiSetNetlistFormatterClass(  
    o_tool  
    s_class  
)  
=> s_class
```

Description

Registers the netlist formatter class with the tool. This function is normally called from the `asiInitFormatter` method and should be defined for the interface.

Arguments

<i>o_tool</i>	The object representing the simulator interface.
<i>s_class</i>	A symbol representing the formatter class for the simulator of interest.

Value Returned

<i>s_class</i>	A symbol representing the formatter class for the simulator of interest.

Examples

The following example registers the netlist formatter class for the given simulator:

```
asiSetNetlistFormatterClass( o_tool 'spectreFormatter') => spectreFormatter
```

asiCreateFormatter

```
asiCreateFormatter(  
    o_session  
)  
=> o_formatter
```

Description

Creates a design object with the `nlCreateDesign` call, using the information on the OASIS session. Subsequently, the formatter is created with a call to `nlCreateFormatter`, using the information on the session. The formatter is added to the session and can be obtained with `asiGetFormatter`. This is a convenience function that you cannot redefine, and the interface should not call it.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>o_formatter</i>	The formatter object created.

Examples

Creates a design object for the specified OASIS session object:

```
asiCreateFormatter( session )  
=> Obj1
```

asiCreateCdsenvFile

```
asiCreateCdsenvFile(  
    s_toolName  
)  
=> t / nil
```

Description

Creates a `.cdsenv` file for the specified tool and dumps it to the current working directory. This is meant as a development utility for integrators only.

Arguments

<code>s_toolName</code>	The name of the tool. For example, Spectre.

Value Returned

<code>t</code>	The <code>.cdsenv</code> file is created.
<code>nil</code>	No <code>.cdsenv</code> file is created.

Examples

Creates a file `spectreCdsenvFile` containing Spectre tool information in your current working directory.

```
asiCreateCdsenvFile('spectre ')  
=> t
```

Creates file `./SpectreCdsenvFile` for Spectre tool.

```
asiCreateCdsenvFile('Spectre')  
=> t
```

asiGetFormatter

```
asiGetFormatter(  
    o_session  
)  
=> o_formatter / nil
```

Description

Returns the formatter created with the last `asiCreateFormatter` call. This is a convenience function that you should not redefine and the interface should not call.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>o_formatter</i>	The formatter object.
<i>nil</i>	No formatter object is available for the session.

Examples

The following example returns the name of the formatter object that was created using the last `asiCreateFormatter` call:

```
asiGetFormatter( session )  
=> Obj1
```

asiGetSimInputFileName

```
asiGetSimInputFileName(  
    o_session  
)  
=> t_name
```

Description

Returns the name of the simulator input file. For the `asiAnalog_session` class, this is input followed by the return value of `asiGetSimInputFileSuffix`.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>t_name</i>	The name of the simulator input file.

Examples

The following example returns the name of the simulator input file for the given OASIS session object:

```
asiGetSimInput  
FileName( session ) => "input.ckt"  
=> MyFile1
```

asiGetSimInputFileSuffix

```
asiGetSimInputFileSuffix(  
    o_session  
)  
=> t_name
```

Description

Returns the suffix used for the simulator input file. This method can be redefined, and must return a string, or a SKILL error will result.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>t_name</i>	The suffix of the simulator input file.

Examples

The following example returns the suffix for simulator input file used in the given session:

```
asiGetSimInputFileSuffix( session )  
=> ".ckt"
```

Netlist Functions

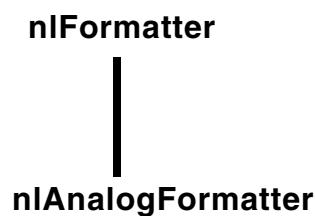
Netlist functions are categorized into the following types:

- [The nlAnalogFormatter Class](#)
- [The Netlister Object](#)
- [Methods for Instances](#)
- [Cellviews](#)
- [Designs](#)
- [Other Customization Procedures](#)
- [Other Backend Netlister Functions](#)
- [HSPICE Functions](#)
- [Name Mapping Variables](#)

The nlAnalogFormatter Class

The `nlAnalogFormatter` class represents the object that formats the design for the netlist file. It is responsible for printing the instances, the subcircuits, the comments, the global statements, and other information associated with the design connectivity.

The netlist calls several methods, such as `nlPrintInst`. The netlist traverses the design and calls the methods of the `nlAnalogFormatter` class. This class is tailored to the Spice 3 simulator and is derived from the more generic `nlFormatter` class:



Except as noted, these netlist methods might be redefined.

The SKILL functions in this class are listed below:

`nlGetNetlist`

`nlGetPCellParamSource`

`netlistDir`

`nlGetScratchInstance`

`nlGetSwitchMaster`

`nlGetToolName`

`nlInitialize`

`nlPrintHeader`

`nlIncludePspiceFile`

`nlIncludeVerilogaFile`

`nlIncludeVerilogFile`

`nlIncludeDbDSPFTextFile`

Virtuoso ADE SKILL Reference

Core Netlister Functions

[nlIncludeDbSPICEMODELTextFile](#)

[nlIsPcellInstance](#)

[nlIsPcellParam](#)

[nlIsSmartExtractedView](#)

[nlPrintFooter](#)

[nlPrintSubcktHeaderComments](#)

[nlPrintTopCellHeaderComments](#)

[nlPrintTopCellFooterComments](#)

[nlPrintTopCellHeader](#)

[nlPrintTopCellFooter](#)

[nlPrintSubcktHeader](#)

[nlPrintSubcktFooter](#)

[nlPrintSubcktFooterComments](#)

[nlPrintInstComments](#)

[nlPrintInst](#)

[nlPrintInstEnd](#)

[nlPrintSubcktBegin](#)

[nlPrintSubcktName](#)

[nlPrintSubcktEnd](#)

[nlPrintHeaderComments](#)

[nlPrintSubcktParameters](#)

[nlPrintSubcktTerminalList](#)

[nlPrintInstName](#)

[nlPrintInstSignals](#)

Virtuoso ADE SKILL Reference

Core Netlister Functions

nlPrintModelName

nlPrintInstParameters

nlGetNetlister

```
nlGetNetlister(  
    o_formatter  
)  
=> o_netlister
```

OR

```
nlGetNetlister(  
    o_instance  
)  
=> o_netlister
```

Description

Returns an object representing the netlister.

You cannot redefine this function.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The instance object.

Value Returned

<i>o_netlister</i>	The netlister object.

Examples

The following example retrieves an object representing a netlister:

```
o_netlister = nlGetNetlister(formatter)  
nlGetNetlister( inst )  
=> myNetlister1
```

nlGetPCellParamSource

```
nlGetPCellParamSource(  
    o_cellView  
    l_parameters  
)  
=> l_BooleanValues / nil
```

Description

Identifies the source of value of the Pcell parameters of the given cellview.

Arguments

<i>o_cellview</i>	CellView ID of the Pcell.
<i>l_parameters</i>	List of parameters for which the source needs to be determined

Value Returned

<i>l_BooleanValues</i>	List of Boolean values where each element of the list indicates whether corresponding parameter in input list is overridden on instances (for this Pcell variant) or is taking a default value. The value <i>t</i> in this list indicates that the parameter is overridden on instance and <i>nil</i> indicates that the parameter has taken the default value.
<i>nil</i>	There is an error because either the cellview is called outside the context of netlister or the cellview ID is not a valid Pcell.

Examples

Returns a list of Boolean values that indicates whether the parameter is overridden (*t*) or takes its default value (*nil*).

```
nlGetParamList (o_cellView)  
=> l_parameters  
nlGetPCellParamSource (o_cellView, l_parameters)  
=> t nil nil t
```

netlistDir

```
netlistDir(  
    t_dirName  
)  
=> g_undefined / nil
```

Description

Specifies the directory in which the final netlist is to be created.

Arguments

<i>t_dirName</i>	Name of the directory in which the final netlist is to be created.

Value Returned

<i>g_undefined</i>	An undefined value.
<i>nil</i>	The specified directory does not exist or there is an error.

Examples

Specifies the netlist directory to store the netlist files.

```
netlistDir("~/simulation/ckt/spectre/schematic/netlist") => "~/simulation/ckt/  
spectre/schematic/netlist"
```

Returns the path of the netlist directory.

```
netlistDir() => "~/simulation/ckt/spectre/schematic/netlist"
```

nlGetScratchInstance

```
nlGetScratchInstance(  
    t_libName  
    t_cellName  
    t_viewName  
    t_instanceName  
)  
=> d_databaseID / nil
```

Description

Returns the database ID of the scratch instance to be used for Pcell evaluations.

If a maestro view is not open, set the environment variable switchViewList before using this function.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>t_instanceName</i>	Name of the instance.

Value Returned

<i>d_databaseID</i>	The database ID.
<i>nil</i>	The specified arguments are invalid or the instance with the given name does not exist in the specified cell view.

Examples

The following example returns the database ID of the scratch instance to be used for Pcell evaluations:

```
nlGetScratchInstance("Two_Stage_Opamp" "OpAmp" "schematic" "M4")  
=> db:0x654321bb
```

nlGetScratchInstPcellParamValue

```
nlGetScratchInstPcellParamValue(  
    t_instID  
    t_paramName  
)  
=> t_value / nil
```

Description

Returns the parameter value of the scratch instance to be used for Pcell evaluations.

If a maestro view is not open, set the environment variable switchViewList before using this function.

Note: This function must be called from the netlist procedure during netlisting.

Arguments

<i>t_instID</i>	Specifies the database ID of the scratch instance.
<i>t_paramName</i>	Specifies the name of the Pcell parameter

Value Returned

<i>t_value</i>	Returns the value of the Pcell parameter value stored in given scratch instance.
<i>nil</i>	The specified arguments are invalid or the instance with the given name does not exist in the specified cellview.

Examples

The following example shows how this function returns the database ID of the scratch instance to be used for Pcell evaluations along with the its Pcell parameter value.

```
instId = nlGetScratchInstance("Two_Stage_Opamp" "OpAmp" "schematic" "M4")  
=> db:0x654321bb  
nlGetScratchInstPcellParamValue(instId "srcCell")  
=> "Analog"
```


nlGetSwitchMaster

```
nlGetSwitchMaster(  
    t_libName  
    t_cellName  
    t_viewName  
    t_instanceName  
    t_deviceParamName  
)  
=> d_databaseID / nil
```

Description

Returns the database ID of the switch master.

If a maestro view is not open, set the environment variable switchViewList before using this function.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>t_instanceName</i>	Name of the instance.
<i>t_deviceParamName</i>	Name of the device parameter.

Value Returned

<i>d_databaseID</i>	The database ID.
<i>nil</i>	The specified arguments are invalid or the instance with the given name does not exist in the specified cell view.

Examples

The following example returns the database ID of the switch master.

```
nlGetSwitchMaster("Two_Stage_Opamp" "OpAmp" "schematic" "M4")  
=> db:0x123456aa
```

nlGetToolName

```
nlGetToolName(  
    o_formatter  
)  
=> s_toolName
```

Description

Returns a symbol representing the simulator. It returns the value of the tool name. This name is used for the selection of the simulator information on the library component.

You cannot redefine this function.

Arguments

<i>o_formatter</i>	The formatter object.

Value Returned

<i>s_toolName</i>	Name of the tool.

Examples

The following example returns the tool name:

```
nlGetToolName( formatter )
```

Virtuoso ADE SKILL Reference

Core Netlist Functions

nlInitialize

```
nlInitialize(  
    o_formatter  
)  
=> o_formatter / nil
```

Description

For the `nlFormatter` class, this method initializes the netlist. This method can be redefined for the simulator-specific netlist and is called by `nlCreateFormatter`. This method initializes all simulator-specific aspects of netlisting such as name mapping. For the `nlAnalogFormatter` class, this method sets a number of netlist options. These options and their values are shown in the table below.

To inspect the value of an option, use `nlGetOption`.

Option	Value
<code>hierarchyDelimiter</code>	<code>"."</code>
<code>globalParamPrefix</code>	<code>"_gpar"</code>
<code>globalNetPrefix</code>	<code>" "</code>
<code>instNamePrefix</code>	<code>"_inst"</code>
<code>invalidNetNames</code>	<code>'(("gnd!" "0"))</code>
<code>inhModelName</code>	<code>t / nil</code> specifies whether the simulator provides support for model name passing. A <code>nil</code> value would result in an error if model name is passed through the hierarchy for a stopping instance.
<code>linePrefix</code>	<code>"+"</code>
<code>linePostfix</code>	<code>nil</code>
<code>netNamePrefix</code>	<code>"_net"</code>
<code>mapInstFirstChar</code>	special characters and numbers
<code>mapInstInName</code>	all special characters
<code>mapModelFirstChar</code>	all special characters and numbers
<code>mapModelInName</code>	all special characters
<code>mapNetFirstChar</code>	all special characters and numbers
<code>mapNetInName</code>	special characters

Virtuoso ADE SKILL Reference

Core Netlist Functions

Option	Value
maxNameLength	1024
modulePrefix	"_sub"
paramNamePrefix	"_par"
subcktIndentString	four spaces
useInstNamePrefix	t

To change any of these settings, use `callNextMethod()` to inherit as much as possible and then make the changes.

Arguments

o_formatter The formatter object.

Value Returned

o_formatter The formatter object.
nil The operation failed.

Examples

The following example uses `callNextMethod` to do the initialization in the `nlAnalogFormatter` class and then:

- Changes the hierarchy delimiter from “.” to “:”
- Changes the maximum name length from 1024 chars to 64 chars

```
defmethod( nlInitialize((formatter <yourSimulator>Formatter))
  let( (nlForm)
  callNextMethod( formatter)
  nlForm=nlGetNetlister( formatter )
  nlSetOption( nlForm 'hierarchyDelimiter ":")
  nlSetOption( nlForm 'maxNameLength 64)
  formatter
) )
=> formatter1
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

Related Topics

[The Netlister Object](#)

nlPrintHeader

```
nlPrintHeader(  
    o_formatter  
)  
=> t / nil
```

Description

This method writes the beginning comment, adds .GLOBAL, and prints header comments.

Arguments

<i>o_formatter</i>	The formatter object.

Value Returned

t	The comment is written.
nil	The comment cannot be written due to an error.

Examples

The following example prints the header comments:

```
defmethod( nlPrintHeader ((formatter nlAnalogFormatter))  
let( ((netlist nlGetNetlist(formatter))  
nlPrintString( netlist (nlGetOption netlist 'begComment))  
nlPrintString( netlist "\n")  
nlPrintString( netlist ".GLOBAL")  
foreach( glob  
    setof( x (nlGetGlobalNets netlist) (nequal x "gnd!"))  
    nlPrintString( netlist " ")  
    nlPrintString( netlist (nlMapGlobalNet netlist glob))  
)  
nlPrintHeaderComments( formatter)  
nlPrintString(netlist "\n")  
=> t  
)  
)
```

nlIncludePspiceFile

```
nlIncludePspiceFile(  
    o_formatter  
    t_fileName  
    t_masterName  
)  
=> t / nil
```

Description

Prints the include statement for `pspice` cellviews in the design. This function is called before printing the footer for the netlist for all `pspice` modules. If your simulator does not support `pspice` views, call [nlError](#) function in this method.

Arguments

<i>o_formatter</i>	The formatter object.
<i>t_fileName</i>	The file name in which the include statements should be printed.
<i>t_masterName</i>	The master name of the model added through include statement.

Value Returned

<i>t</i>	The include statement is printed in the specified file.
<i>nil</i>	There command cannot be run due to an error.

Examples

The following example prints the include statement for `pspice` cellviews in the `TopCell.pspice` file:

```
nlIncludePspiceFile(spectre_formatter "/home/TopCell.pspice" "TopCell_amplifier")  
=> t
```

nlIncludeVerilogaFile

```
nlIncludeVerilogaFile(  
    o_formatter  
    t_filename  
    t_master  
)  
=> t / nil
```

Description

Prints the include statement for Verilog-a cell views in the design. This is called before printing the footer for the netlist for all Verilog-a modules. If your simulator does not support Verilog-a views, call [nlError](#) in this method.

Arguments

<i>o_formatter</i>	The formatter object.
<i>t_filename</i>	The name of the file to include
<i>t_master</i>	The value for <code>-master</code> option on the include line

Value Returned

<i>t</i>	The include statement is printed.
<i>nil</i>	Unsuccessful operation.

Examples

The following example prints the include statement as specified:

```
(defmethod nlIncludeVerilogaFile ((obj <yourSimulator>Formatter) file master)  
(when master  
  (nlError (nlGetNetlister obj) "Cannot support multile modules with the same  
name"))  
  (nlPrintStringNoFold (nlGetNetlister obj) "ahdl_include \"" file "\"\n")  
)
```

Related Topics

[nlError](#)

nlIncludeVerilogFile

```
nlIncludeVerilogFile(  
    o_formatter  
    t_filename  
    t_master  
)  
=> t / nil
```

Description

Prints the include statement for Verilog type text cell view in the design. This is called before printing the footer for the netlist for all Verilog text cell views. If your simulator does not support Verilog-a views, call [nlError](#) in this method.

Arguments

<i>o_formatter</i>	The formatter object.
<i>t_filename</i>	The name of the file to include
<i>t_master</i>	The value for <code>-master</code> option on the include line

Value Returned

<i>t</i>	The include statement for Verilog type text cell view is printed.
<i>nil</i>	Unsuccessful operation.

Examples

The following example prints the include statement for Verilog type text cell view:

```
(defgeneric nlIncludeVerilogFile (obj_file_master)  
=> t)
```

Related Topics

[nlError](#)

nlIncludeDbDSPFTextFile

```
nlIncludeDbDSPFTextFile(  
    o_formatter  
    t_filename  
    t_master  
)  
=> t / nil
```

Description

Prints the `dspf_include` statement for the DSPF type text cellview in the design.

Arguments

<i>o_formatter</i>	The OASIS session object.
<i>t_filename</i>	The name of the file to include in the design.
<i>t_master</i>	The value for <code>-master</code> option on the include line.

Value Returned

<i>t</i>	The <code>dspf_include</code> statement is printed in the design.
<i>nil</i>	Unsuccessful operation.

Examples

The following example prints the `dspf_include` statement for the DSPF type text cellview for the `formatter` session object. The `cvFilePath` file is include in the design:

```
nlIncludeDbDSPFTextFile( formatter cvFilePath master )  
=> t
```

nlIncludeDbSPICEMODELTextFile

```
nlIncludeDbSPICEMODELTextFile(  
    o_formatter  
    t_filename  
)  
=> t / nil
```

Description

Prints the include statement for SPICEMODEL cellviews in the design. This function is called before printing the footer for the netlist for all SPICEMODEL modules. If your simulator does not support SPICEMODEL views, call the nlError function in this method.

Arguments

<i>o_formatter</i>	The formatter object.
<i>t_filename</i>	The file name in which the include statements should be printed.

Value Returned

<i>t</i>	The include statement is printed in the specified file.
<i>nil</i>	There is an error.

Examples

The following function prints the include statement for SPICEMODEL cellviews in the TopCell.mod file.

```
nlIncludeDbSPICEMODELTextFile (spectre_formatter "/home/TopCell.mod")  
=> t
```

Related Topics

[nlError](#)

nIsPcellInstance

```
nIsPcellInstance(  
    t_libName  
    t_cellName  
    t_viewName  
    t_instanceName  
)  
=> t / nil
```

Description

Checks if the specified instance name represents a schematic Pcell instance.

If a maestro view is not open, set the environment variable `switchViewList` before using this function.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>t_instanceName</i>	Name of the instance.

Value Returned

<i>t</i>	The specified instance name represents a schematic Pcell instance.
<i>nil</i>	■ The arguments specified for the function are invalid. ■ The instance with the given name does not exist in the specified cell view. ■ The specified instance does not represent a schematic Pcell instance.

Examples

The following functions check whether the specified instance represents a schematic Pcell instance.

The first function returns `t` indicating that the specified instance name represents a schematic Pcell instance.

The second function returns `nil` indicating that the specified instance name does not represent a schematic Pcell instance.

```
nlIsPcellInstance("Two_Stage_Opamp" "OpAmp" "schematic" "M4")
=> t
nlIsPcellInstance("Two_Stage_Opamp" "OpAmp" "schematic" "V0")
=> nil
```

Related Topics

[switchViewList](#)

nllsPcellParam

```
nllsPcellParam(  
    t_libName  
    t_cellName  
    t_viewName  
    t_instanceName  
    t_deviceParamName  
)  
=> t / nil
```

Description

Checks if the specified device parameter represents a Pcell parameter.

If a maestro view is not open, set the environment variable `switchViewList` before using this function.

Arguments

<i>t_libName</i>	Name of the library.
<i>t_cellName</i>	Name of the cell.
<i>t_viewName</i>	Name of the view.
<i>t_instanceName</i>	Name of the instance.
<i>t_deviceParamName</i>	Name of the device parameter.

Value Returned

<i>t</i>	The specified device parameter represents a Pcell parameter.
<i>nil</i>	The specified device parameter does not represent a Pcell parameter or the provided arguments are invalid.

Examples

The following functions check whether the specified device parameter represents a Pcell parameter.

Virtuoso ADE SKILL Reference

Core Netlist Functions

The first function returns `t` indicating that the specified device parameter represents a Pcell parameter.

The second function returns `nil` indicating that the specified device parameter does not represent a Pcell parameter.

```
nlIsPcellParam("Two_Stage_Opamp" "OpAmp" "schematic" "M4" "simM")
=> t
nlIsPcellParam("Two_Stage_Opamp" "OpAmp" "schematic" "M4" "w")
=> nil
```

Related Topics

[switchViewList](#)

nllsSmartExtractedView

```
nllsSmartExtractedView(  
    o_cellViewHandle  
)  
=> t / nil
```

Description

Identifies if the currently netlisted cellview is of the type *smart_view*. You can use this function in custom netlist procedures to modify the syntax printed to the netlist as required. For example, you can modify the syntax to determine if the printing of parenthesis needs to be enabled or disabled when printing the port connections of the instance.

Arguments

o_cellViewHandle A handle to the current cellview that is being netlisted.

Value Returned

t	The cellview is netlisted.
nil	There is an error.

Examples

```
(nllsSmartExtractedView ((nlGetCurrentCellView (nlGetNetlist inst)))  
=> t
```

The following snippet shows how to use this function in a custom netlist procedure:

```
(defun _myCustomNetlistProc (instance)  
  (let (netlist isNextGenExtrCV currentCV signalList)  
    netlist = (nlGetNetlist instance)  
    currentCV = (nlGetCurrentCellView netlist)  
    isNextGenExtrCV = (nllsSmartExtractedView currentCV)  
    signalList = (nlGetSignalList instance)  
    (when (not isNextGenExtrCV) nlPrintString(netlist sprintf(nil "(")))  
    (foreach signal signalList nlPrintString(netlist sprintf(nil "%s  
" signal)))  
    (when (not isNextGenExtrCV) nlPrintString(netlist sprintf(nil ")"))  
  ))  
)  
=> t
```


nlPrintFooter

```
nlPrintFooter(  
    o_formatter  
)  
=> t / nil
```

Description

This method is called at the end of netlisting. It does not print anything at the end of the netlist for the `nlAnalogFormatter` class.

Arguments

<i>o_formatter</i>	The formatter object.

Value Returned

t	The footer is printed.
nil	There is an error.

Examples

Prints the footer for the given formatter object.

```
defmethod( nlPrintFooter ((formatter xyzFormatter))  
    callNextMethod()  
    nlPrintComment( nlGetNetlister( xyzFormatter )  
                    "End of netlist\n")  
)  
=> t
```

nlPrintSubcktHeaderComments

```
nlPrintSubcktHeaderComments (
    o_formatter
    o_cellView
)
=> t
```

Description

Prints the comments for the subcircuit header and the mapping information when the `printSubcktComments` option is set.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

<i>t</i>	Always returns <i>t</i> .

Examples

The following example prints comments and mapping information:

```
defmethod( nlPrintSubcktHeaderComments ((formatter
                                         xyzFormatter) cv)
let( (nl)
    callNextMethod()
    nl = nlGetNetlist(formatter)
    nlPrintComment( nl "anything else")
) )
=> t
```

nlPrintTopCellHeaderComments

```
nlPrintTopCellHeaderComments (
    o_formatter
    o_cellView
)
=> t
```

Description

Calls `nlPrintSubcktHeaderComments` for the `nlAnalogFormatter` class.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

t	Always returns t.

Examples

The following example prints top cell header comments:

```
defmethod( nlPrintTopCellHeaderComments ((formatter
                                         xyzFormatter) cv)
nlPrintSubcktHeaderComments( formatter cv)
)
=> t
```

nlPrintTopCellFooterComments

```
nlPrintTopCellFooterComments (
    o_formatter
    o_cellView
)
=> t
```

Description

Returns `t` at the `analogFormatter` level.

Arguments

<code>o_formatter</code>	The formatter object.
<code>o_cellView</code>	The object representing the cell view.

Value Returned

<code>t</code>	Always returns <code>t</code> .

Examples

The following example prints the top cell footer comments:

```
defmethod( nlPrintTopCellFooterComments ((formatter
                                         xyzFormatter) cv)
let ( (nl)
    callNextMethod()
    nl = nlGetNetlister(formatter)
    nlPrintComment( nl "anything else")
) )
=> t
```

nlPrintTopCellHeader

```
nlPrintTopCellHeader(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Prints the header of the top-level circuit by calling `nlPrintTopCellHeaderComments`.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

<i>t</i>	The header is printed.
<i>nil</i>	There is an error.

Examples

The following example prints the header of the top-level circuit:

```
defmethod( nlPrintTopCellHeader((formatter xyzformatter) cv)  
    (nlPrintTopCellHeaderComments formatter cv)  
)  
=> t
```

nlPrintTopCellFooter

```
nlPrintTopCellFooter(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Writes the top cell view footer. This function prints an empty line and calls `nlPrintTopCellFooterComments`.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

<i>t</i>	The footer is printed.
<i>nil</i>	There is an error.

Examples

The following example prints the top cellview footer information:

```
defmethod( nlPrintTopCellFooter ((formatter  
                                xyzFormatter) cv)  
    callNextMethod()  
    myTopCellFooter()  
=> t  
)
```

nlPrintSubcktHeader

```
nlPrintSubcktHeader(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Writes the header for a subcircuit following these steps: prints comments by calling `nlPrintSubcktHeaderComments`; prints the subckt begin keyword by calling `nlPrintSubcktBegin`; prints the subckt name by calling `nlPrintSubcktName`; prints the subckt terminal list by calling `nlPrintSubcktTerminalList`.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

<i>t</i>	The header for the subcircuit is added.
<i>nil</i>	There is an error.

Examples

The following example prints the header for a subcircuit in a stepwise manner:

```
defmethod( nlPrintSubcktHeader((formatter xyzformatter) cv )  
let( (nl)  
nl = nlGetNetlister(formatter)  
    (nlPrintSubcktHeaderComments formatter cv)  
    (nlPrintSubcktBegin formatter cv)  
    (nlPrintSubcktName formatter cv)  
    (nlPrintSubcktTerminalList formatter cv)  
    (nlPrintString nl "\n")  
) )  
=> t
```

nlPrintSubcktFooter

```
nlPrintSubcktFooter(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Writes the footer for the subcircuit. For the `nlAnalogFormatter` class, it prints `.ends.`

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

<i>t</i>	The footer for the subcircuit is written.
<i>nil</i>	There is an error.

Examples

The following example writes the footer for the given subcircuit:

```
defmethod( nlPrintSubcktFooter ((formatter  
                                xyzFormatter) cv)  
nlPrintSubcktEnd( formatter cv)  
nlPrintTopCellFooterComments( formatter cv)  
=> t  
)
```


nlPrintSubcktFooterComments

```
nlPrintSubcktFooterComments(  
    o_formatter  
    o_cellView  
)  
=> t
```

Description

Prints the comments for the subcircuit footer by printing the string 'End of subcircuit definition.' preceded by the comment begin string.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cell view.

Value Returned

t	The subcircuit footer comments are printed.

Examples

The following example prints subcircuit footer comments:

```
defmethod( nlPrintSubcktFooterComments ((formatter  
                                         xyzFormatter) cv)  
let( (nl)  
    callNextMethod()  
    nl = nlGetNetlist(formatter)  
    nlPrintComment( nl "anything else")  
  ) )  
=> t
```

nlPrintInstComments

```
nlPrintInstComments(  
    o_formatter  
    o_instance  
)  
=> t
```

Description

Prints the comments for an instance.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance.

Value Returned

t	Comments for the instance are printed.

Examples

The following example prints the comments for the given instance:

```
defmethod( nlPrintInstComments ((formatter  
                                xyzFormatter) inst)  
let( (nl)  
    callNextMethod()  
    nl = nlGetNetlister(formatter)  
    nlPrintComment( nl "anything else")  
) )
```

nlPrintInst

```
nlPrintInst(  
    o_formatter  
    o_instance  
)  
=> t / nil
```

Description

Prints the netlist statement for an instance. This is the default netlist procedure for a component.

It prints the following:

- ❑ instance comments by calling `nlPrintInstComments`
- ❑ a string to indent (for the top-level circuit, this is an empty string; for subcircuits, this is the value of the `subcktIndentString` netlist option.)
- ❑ instance name
- ❑ signals
- ❑ model name
- ❑ instance parameters in name=value pairs.

If a netlist procedure is specified as a `netlistProcedure` in the CDF `simInfo`, that procedure is called instead of `nlPrintInst`.

The end of the instance line, `\n`, is not printed by this method or the netlist procedure. The end of the line is printed by the `nlPrintInstEnd` method.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance.

Value Returned

<i>t</i>	The netlist statement is printed.
<i>nil</i>	There is an error.

Virtuoso ADE SKILL Reference

Core Netlist Functions

Examples

The following example prints the netlist statement:

```
defmethod( nlPrintInst ((formatter nlAnalogFormatter) inst)
nlPrintInstComments( formatter inst)
nlPrintIndentString( nlGetNetlist( formatter ))
nlPrintInstName( formatter inst)
nlPrintInstSignals( formatter inst)
nlPrintModelName( formatter inst)
nlPrintInstParameters( formatter inst)
=> t
)
```

nlPrintInstEnd

```
nlPrintInstEnd(  
    o_formatter  
    o_instance  
)  
=> t / nil
```

Description

Prints the end of the instance statement, which is a return (`\n`). This method is called by the netlister after the netlist procedure or by `nlPrintInst`.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance.

Value Returned

<i>t</i>	The end of the instance statement is printed.
<i>nil</i>	There is an error.

Examples

The following example prints the end of the instance statement:

```
defmethod( nlPrintInstEnd ((formatter nlAnalogFormatter) inst)  
    nlPrintString( nlGetNetlister( formatter) "\n")  
    => t  
)
```

nlPrintSubcktBegin

```
nlPrintSubcktBegin(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Prints the `.subckt` keyword for the `nlAnalogFormatter` class. This method is called by `nlSubcktHeader`.

Arguments

<code>o_formatter</code>	The formatter object.
<code>o_cellView</code>	The object representing the cellview.

Value Returned

<code>t</code>	The keyword is printed.
<code>nil</code>	There is an error.

Examples

The following example prints the `nlAnalogFormatter` class:

```
defmethod( nlPrintSubcktBegin ((formatter nlAnalogFormatter) _cv)  
    nlPrintString( nlGetNetlist( formatter) ".subckt")  
    => t  
)
```

nlPrintSubcktName

```
nlPrintSubcktName(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Prints a space and the simulator name of the subcircuit. This method is used by nlSubcktHeader.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cellview.

Value Returned

<i>t</i>	The simulator name is printed.
<i>nil</i>	There is an error.

Examples

The following example prints the simulator name of the subcircuit:

```
defmethod( nlPrintSubcktName ((formatter nlAnalogFormatter) cv)  
    let( ( nl nlGetNetlist(formatter))  
        nlPrintString( nl " ")  
        nlPrintString( nl nlGetSimName (cv) )  
        t )  
)
```

nlPrintSubcktEnd

```
nlPrintSubcktEnd(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

For the `nlAnalogFormatter` class, prints the `.ends` keyword, followed by a space and the simulator name of the subcircuit, to mark the end of the subcircuit definition. It is called by `nlPrintSubcktFooter`.

Arguments

<code>o_formatter</code>	The formatter object.
<code>o_cellView</code>	The object representing the cellview.

Value Returned

<code>t</code>	The keyword followed by a space and the simulator name is printed.
<code>nil</code>	there is an error.

Examples

The following example prints the `.ends` keyword

```
defmethod( nlPrintSubcktEnd ((formatter nlAnalogFormatter) cv)  
    let( ( nl nlGetNetlist(formatter))  
        nlPrintString( nl ".ends")  
        nlPrintString( nl nlGetSimName (cv) )  
        nlPrintString( nl "\n")  
        t  
    ) )  
=> t
```


nlPrintHeaderComments

```
nlPrintHeaderComments(  
    o_formatter  
)  
=> t / nil
```

Description

Prints the comments for the netlist file, including the library, cell, and view names of the top-level cellview of the design netlisted. This method only has effect when the `printFileComments` netlist option is set.

Arguments

<i>o_formatter</i>	The formatter object.

Value Returned

<i>t</i>	The comments are printed.
<i>nil</i>	There is an error.

Examples

The following example prints the comments for the netlist file:

```
defmethod( nlPrintHeaderComments ((formatter  
                                   xyzFormatter) inst)  
let( (nl)  
  nl = nlGetNetlist(formatter)  
  callNextMethod()  
  nl = nlGetNetlist(formatter)  
  nlPrintComment( nl "anything else")  
) )  
=> t
```

nlPrintSubcktParameters

```
nlPrintSubcktParameters(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Prints the passed parameters for the subcircuit definition.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cellview.

Value Returned

t	The parameters are printed.
nil	There is an error.

Examples

The following example prints the subcircuit definition parameters:

```
defmethod( nlPrintSubcktParameters  
    ((formatter nlAnalogFormatter) cv)  
let( ((plist nlGetParamList( cv))  
      (nl nlGetNetlist( formatter)))  
when( plist  
    nlPrintString( nl ".PARAMETERS")  
    foreach( param plist  
        cond(  
            ((cadr param) nlPrintString( nl " "  
                (car param) "=" (cadr param)))  
            (t nlPrintString( nl " " (car param)))  
        )  
    )  
    nlPrintString( nl "\n")  
)))  
=> t
```

nlPrintSubcktTerminalList

```
nlPrintSubcktTerminalList(  
    o_formatter  
    o_cellView  
)  
=> t / nil
```

Description

Prints the simulator names of the signals connected to the terminals for a subcircuit definition and handles the signals resulting from inherited connections at a lower level .

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_cellView</i>	The object representing the cellview

Value Returned

t	The simulator names are printed.
nil	There is an error.

Examples

The following example prints the simulator names:

```
defmethod( nlPrintSubcktTerminalList ((formatter  
                                     xyzFormatter) cv )  
    let( ( (nl nlGetNetlister(formatter))  
          foreach( net nlGetSimTerminalNets( cv )  
                  nlPrintString( nl " " )  
                  nlPrintString( nl net ) )  
    ) )  
=> t
```

nlPrintInstName

```
nlPrintInstName(  
    o_formatter  
    o_instance  
)  
=> t / nil
```

Description

Prints the simulator name of the instance, taking the instance name prefix specified on the component into account when the simulator so requires. This is determined with the `useInstNamePrefix` netlist option.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance.

Value Returned

<i>t</i>	The simulator name is printed.
<i>nil</i>	There is an error.

Examples

The following example prints the simulator name:

```
defmethod( nlPrintInstName ((formatter  
                             xyzFormatter) inst)  
nlPrintString( nlGetNetlist( formatter)  
               nlGetSimName (inst ) ) )  
=> t
```

nlPrintInstSignals

```
nlPrintInstSignals(  
    o_formatter  
    o_instance  
)  
=> t / nil
```

Description

Prints the simulator names of the signals according to the terminal order specified on the component, using the `nlGetSignalList` method of the instance.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance.

Value Returned

<i>t</i>	The simulator names of the signals are printed.
<i>nil</i>	There is an error.

Examples

The following example prints the simulator names of the signals:

```
defmethod( nlPrintInstSignals ((formatter  
                                xyzFormatter) inst)  
let( (sigs (fp nlGetNetlist(formatter)) )  
    ;; print terminals  
    when( sigs = (nlGetSignalList inst)  
        foreach( sig sigs|  
            nlPrintString( fp sig " ")  
        ) ) )  
=> t
```

nlPrintModelName

```
nlPrintModelName(  
    o_formatter  
    o_instance  
)  
=> t / nil
```

Description

Prints the model name. The `nlGetModelName` for the instance is used for the model name.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance.

Value Returned

t	The model name is printed.
nil	There is an error.

Examples

The following example prints the model name:

```
defmethod( nlPrintModelName ((formatter xyzFormatter) inst)  
    nlPrintString( nlGetNetlist( formatter ) " "  
        nlGetModelName( inst ) )  
)  
=> t
```

If your simulator supports model name passing, you can use the function `nlIsModelNameInherited` and alter the formatting of the instance line for the scenario as per the requirements of your simulator.

nlPrintInstParameters

```
nlPrintInstParameters(  
    o_formatter  
    o_instance  
)  
=> t / nil
```

Description

Prints the instance parameters in name=value pairs.

Arguments

<i>o_formatter</i>	The formatter object.
<i>o_instance</i>	The object representing the instance

Value Returned

t	The instance parameters are printed.
nil	There is an error.

Examples

The following example prints the instance parameters:

```
defmethod( nlPrintInstParameters  
            ((formatter nlAnalogFormatter) inst)  
let( (val (nl nlGetNetlist( inst) ) )  
;; print properties  
foreach( par nlGetParamList( inst)  
    unless(  
        val = nlGetParamStringValue( inst par)  
        nlPrintString( nl  
            " " get_string (par) "=" val)  
        ))))  
=> t
```

The Netlister Object

The netlister object represents the engine that produces the netlist. This object drives the format object and provides all necessary services such as name mapping. The netlister is the incremental, hierarchical netlister. You can obtain the netlister object from the formatter using `nlGetNetlister`.



None of the methods defined for the netlister can be redefined.

The options that can be set with the `nlSetOption` method are given in the table below. The values for the `nlAnalogFormatter` are provided in [nlInitialize](#).

This object represents the incremental hierarchical netlister. The netlist options are set with `nlSetOption`. This section also lists the netlist options.

Netlist Options

You can use the following options for direct simulation.

Note: Call `nlGetOptionNameList` to get a complete list of netlist options.

Option Name	Description
<code>begComment</code>	Comment string for lines that begin with a comment for your simulator.
<code>constantsList</code>	A list of names which are used as constant names by your simulator. These names are mapped by the netlister when used as passed parameter names.
<code>forceAlwaysAddPrefixInInstName</code>	

Virtuoso ADE SKILL Reference

Core Netlister Functions

Determines whether to add a prefix in the instance name while generating the netlist.

When the `forceAlwaysAddPrefixInInstName` variable is set to `true`, it checks the value of `alwaysAddPrefixInInstName` and then during netlisting update the `simAlwaysAddPrefixInInstName` variable according to the value specified in the `alwaysAddPrefixInInstName` variable. It also restores the value of `simAlwaysAddPrefixInInstName` to its original value after the netlisting is finished.

When the `forceAlwaysAddPrefixInInstName` variable is set to `nil`, the default value of `simAlwaysAddPrefixInInstName` is used.

`alwaysAddPrefixInInstName`

Controls the behavior of `simAlwaysAddPrefixInInstName` variable for analog class simulators. When the `alwaysAddPrefixInInstName` variable is set to `true`, it sets the `simAlwaysAddPrefixInInstName` variable to `true` and when the `alwaysAddPrefixInInstName` variable is set to `nil`, it sets the `simAlwaysAddPrefixInInstName` variable to `nil`.

`globalNetPrefix`

Specifies the string the formatter should use as the prefix for all global nets in a netlist.

`globalParamPrefix`

Specifies the string the formatter should use as the prefix for all global parameters in a netlist.

`hierarchyDelimiter`

Character used by the simulator for delimiting levels of hierarchy when it flattens the hierarchical netlist. Used for mapping result names.

`instNameDifferentFrom`

A list that contains the name types that the instance names cannot collide with. The types can be one or more of "model", "net", and "parameter".

`instNamePrefix`

Specifies the name prefix to be used when the netlister generates a unique instance name.

`instTermDelimiter`

The delimiter between the instance and the terminal used by the simulator.

`invalidModelNames`

A list of names which are invalid as model names for your simulator.

Virtuoso ADE SKILL Reference

Core Netlist Functions

<code>invalidInstNames</code>	A list of names which are invalid as instance names for your simulator.
<code>invalidNetNames</code>	A list of names which are invalid as net names in the target tool net name space.
<code>invalidParamNames</code>	A list of names which are invalid as parameter names for your simulator.
<code>linePostfix</code>	String placed at the end of the current line when a line output to the netlist exceeds the maximum line length of the simulator and must be split into two lines. This variable is used by the <code>nlPrintString</code> function.
<code>linePrefix</code>	String placed at the beginning of the next line when a line output to the netlist exceeds the maximum line length of the simulator and must be split into two lines. This variable is used by the <code>nlPrintString</code> method.
<code>mapInstFirstChar</code>	A SKILL list that specifies which characters may not begin an instance name for the target simulator. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is <code>nil</code>, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping instance names. For example, if the value of this option is <pre>'(".")</pre> then the name <code>".inst24"</code> is mapped to a name that is generated by the netlist, starting with the value of the <code>instNamePrefix</code> option. For the <code>nlAnalogFormatter</code> class the name could be <code>"_inst3"</code>.

Virtuoso ADE SKILL Reference

Core Netlist Functions

`mapInstInName`

A SKILL list that specifies which characters may not be contained in an instance name for the target simulator. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character.

If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping instance names.

For example, if the value of this option is

```
'( "." ("<" "_") (">") )
```

then the name `"inst.24"` is mapped to a name that is generated by the netlist, starting with the value of the `instNamePrefix` option. For the `nlAnalogFormatter` class the name could be `"_inst3"`. The name `"inst<1>"` could be mapped to `"inst5_1"`.

`mapModelFirstChar`

A SKILL list that specifies which characters may not begin a subcircuit name for the target simulator. Each element in the list may be either a string or another list. If it is a string, it should contain a single character.

This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping subcircuit names. For example, if the value of this option is

```
'( "." )
```

then the name `".sub24"` is mapped to a name that is generated by the netlist, starting with the value of the `modulePrefix` option. For the `nlAnalogFormatter` class the name could be `"_sub3"`.

Virtuoso ADE SKILL Reference

Core Netlist Functions

`mapModelInName`

A SKILL list that specifies which characters may not be contained in an subcircuit name for the target simulator. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements.

The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping subcircuit names.

For example, if the value of this option is

```
'( "." )
```

then the name "sub.24" is mapped to a name that is generated by the netlist, starting with the value of the `modulePrefix` option. For the `nlAnalogFormatter` class the name could be "_sub3".

`mapNetFirstChar`

A SKILL list that specifies which characters may not begin a name for the target simulator. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character.

If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping net names. For example, if the value of this option is

```
'( "." )
```

then the name ".net24" is mapped to a name that is generated by the netlist, starting with the value of the `netNamePrefix` option. For the `nlAnalogFormatter` class the name could be "_net3".

Virtuoso ADE SKILL Reference

Core Netlist Functions

<code>mapNetInName</code>	A SKILL list that specifies which characters may not be contained in a name for the target simulator. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name containing this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is <code>nil</code>, the character specified by the first element is removed. If the second element is a string, that string replaces the character when it is encountered. This list is used when mapping net names. For example, if the value of this option is <pre>'("." ("<" "_") (">"))</pre> then the name <code>"net.24"</code> is mapped to a name that is generated by the netlist, starting with the value of the <code>netNamePrefix</code> option. For the <code>nlAnalogFormatter</code> class the name could be <code>"_net3"</code>. The name <code>"net5<1>"</code> could be mapped to <code>"net5_1"</code>.
<code>mapSubcktTermsToOrder</code>	When set, the subcircuit terminal names are mapped to the order of the terminal.
<code>maxLineLength</code>	Maximum line length of a line output to the netlist file when using the <code>nlPrintString</code> function. When the maximum is reached, the line is broken at the first blank space before this limit is and continued on the next line. If there is no blank space on the line, the line is broke at the limit and continued on the next line.
<code>maxNameLength</code>	Maximum number of character allowed in a simulator name. Used for mapping.
<code>modelNameDifferentFrom</code>	A list that contains the name types that the subcircuit names cannot collide with. The types can be one or more of "instance", "net", and "parameter".
<code>modulePrefix</code>	Specifies the string the formatter uses as the prefix for a mapped subcircuit name in a netlist.
<code>netNameDifferentFrom</code>	

Virtuoso ADE SKILL Reference

Core Netlist Functions

	A list that contains the name types that the net names cannot collide with. The types can be one or more of "model", "instance", and "parameter".
<code>netNamePrefix</code>	Specifies the name prefix to be used when the netlister generates a unique signal name.
<code>paramNameDifferentFrom</code>	
	A list that contains the name types that the parameter names cannot collide with. The types can be one or more of "model", "net", and "instance".
<code>paramNamePrefix</code>	Specifies the name prefix to be used when the netlister generates a unique parameter name.
<code>printFileComments (t)</code>	
	Print the standard comments at the beginning and end of the netlist file.
<code>printInstComments (nil)</code>	
	Print the standard comment before each instance line.
<code>printSubcktComments (t)</code>	
	Print the standard comments at the beginning and end of each subcircuit, including the top-level circuit.
<code>printSubcktTerminalComments (nil)</code>	
	Print the mapping of the subckt terminals to the beginning of the subcircuit.
<code>softLineLength</code>	The desirable maximum line length of a line output to the netlist file when using the <code>nlPrintString</code> function. When the maximum is reached, the line is broken at first blank space before this limit, and continued on the next line. If no space is available, the output is continued on the same line until a space is encountered, or until <code>maxLineLength</code> is reached.
<code>subcktIndentString</code>	A string that represents the indentation in a subcircuit.
<code>subcktInstPrefix</code>	If <code>useInstNamePrefix</code> is set, and no prefix is specified on the CDF, then this prefix is used for subcircuit instances.

Virtuoso ADE SKILL Reference

Core Netlister Functions

`suffixMap`

A SKILL list which specifies the suffixes that need to be mapped. Each element in the list could be a symbol or a string or another list which has either one or two elements. For example, if the value of this option is

```
' ( ' ( "M" "Meg" ) ' ( 'Y ) 'm)
```

then "M" must be mapped to "Meg", and "Y" and "m" must be mapped to the internal default.

`useInstNamePrefix (t)`

When set, the name prefix is taken into account for instance names.

nlError

```
nlError(  
    o_netlist  
    t_error  
)  
=> nil
```

Description

Issues a user error. The error is printed immediately and it is collected on the object. In this way, if netlisting is interrupted, the user is aware of any errors that occurred during netlisting. All errors are printed to the netlist log file.

Arguments

<i>o_netlist</i>	The netlist object.
<i>t_error</i>	The description of the error in the form of a format string.

Value Returned

<i>nil</i>	Returns <i>nil</i> in all cases.

Examples

Displays an error message in the specified format:

```
nlError(nl sprintf("Missing Parameter '%s' in %s" param  
    nlGetSimName(inst) ) )  
=> nil
```


nlObjError

```
nlObjError(  
    o_netlist  
    o_object  
    t_error  
)  
=> nil
```

Description

Similar to [nlError](#), but prints a description of the object along with the error message. The description includes the library name, the cell name, the view name, and the instance name in case the object is an instance.

Arguments

<i>o_netlist</i>	The netlist object.
<i>o_object</i>	The instance or the cellview object.
<i>t_error</i>	The description of the error in the form of a format string.

Value Returned

<i>nil</i>	Returns <i>nil</i> in all cases.

Examples

The following example prints a description of the object along with the error message:

```
nlObjError(nl inst sprintf("Missing Parameter '%s' in" param ) )  
=> nil
```

Related Topics

[nlError](#)

nlGetDesign

```
nlGetDesign(  
    o_netlist  
)  
=> o_design
```

Description

Returns the design object.

Arguments

<i>o_netlist</i>	The netlist object.

Value Returned

<i>o_design</i>	The design object.

Examples

The following example prints the name of the design object associated with the given netlist:

```
nlGetDesign( netlist )  
=> obj1
```

nlGetGlobalNets

```
nlGetGlobalNets(  
    o_netlist  
)  
=> l_globalNets
```

Description

Returns a list of global nets. This method should only be used in the `nlPrintHeader` method of the formatter.

Globals may be included that are not used, due to inherited connections.

Arguments

<i>o_netlist</i>	The netlist object.

Value Returned

<i>l_globalNets</i>	A list of global signals.

Examples

The following example prints a list of global nets:

```
nlGetGlobalNets( netlist )
```

nlGetNetlistDir

```
nlGetNetlistDir(  
    o_netlist  
)  
=> t_netlistDir
```

Description

Returns the netlist directory.

Arguments

<i>o_netlist</i>	The netlist object.

Value Returned

<i>t_netlistDir</i>	A netlist directory.

Examples

The following example retrieves the name of the netlist directory:

```
nlGetNetlistDir( netlist )  
=> myDirectory1
```

nlDisplayOption

```
nlDisplayOption(  
    o_netlist  
)  
=> t
```

Description

Prints the option names available on this object along with their values.

Arguments

<i>o_netlist</i>	The netlist object.

Value Returned

t	Always returns t.

Examples

The following example prints the option names and their values for the given netlist object.

```
nlDisplayOption( netlist )  
=> t
```

nlGetCurrentSwitchMaster

```
nlGetCurrentSwitchMaster(  
    o_netlister  
)  
=> s_id
```

Description

This function returns the database ID for the current switch master for the instance in hierarchical incremental netlisting. This function should only be used while printing the instance statement. Avoid using this function if possible.

Arguments

<i>o_netlister</i>	The netlister object

Value Returned

<i>s_id</i>	The database ID for the current switch master.

Examples

The following example prints the database ID of the current switch master:

```
nlGetCurrentSwitchMaster( netlister )  
=> 1234
```

nlGetNetlistedStopCellViewList

```
nlGetNetlistedStopCellViewList(  
    o_netlister  
)  
=> l_cellView
```

Description

Returns a list of cellview objects treated as stop cells for the specified netlister object.

Arguments

<i>o_netlister</i>	The netlister object.

Value Returned

<i>l_cellView</i>	A list of cellview objects.

Examples

The following example retrieves a list of stop cells for the specified netlister object:

```
obj = (nlGetNetlister formatter)  
cvList = nlGetNetlistedStopCellViewList(obj)  
=> (stdobj@0x2601d470 stdobj@0x2601d458 stdobj@0x2601d440 stdobj@0x2601d428)  
cvList~>cellName  
=> ("cap" "res" "idc" "vsin" )
```

nlGetOption

```
nlGetOption(  
    o_netlist  
    s_name  
)  
=> g_value
```

Description

Returns the value of the option.

Arguments

<i>o_netlist</i>	The netlist object.
<i>s_name</i>	The name of the netlist option.

Value Returned

<i>g_value</i>	Value of the netlist option.

Examples

The following example retrieves the value of the given netlist option:

```
nlGetOption( netlist 'begComment)  
=> "My Comment"
```


nlGetOptionNameList

```
nlGetOptionNameList(  
    o_netlist  
)  
=> l_names
```

Description

Returns a list of option names available on this object.

Arguments

<i>o_netlist</i>	The netlist object.

Value Returned

<i>l_names</i>	A list of option names.

Examples

The following example retrieves a list of option names for the given netlist object:

```
nlGetOptionNameList( netlist )
```

nlMapGlobalNet

```
nlMapGlobalNet(  
    o_netlist  
    t_net  
)  
=> t_map
```

Description

Maps a global net (signal) to the simulator name. This should only be used in the `nlPrintHeader` method of the formatter. Use at any other time is an error.

Arguments

<i>o_netlist</i>	The netlist object.
<i>t_net</i>	The schematic name of the global net.

Value Returned

<i>t_map</i>	Mapped name of the global net.

Examples

The following example returns the name of the global net that is mapped to the given schematic:

```
nlMapGlobalNet( netlist globalNet )  
=> globalNet1
```

nlInfo

```
nlInfo(  
    o_netlister  
    t_info  
    [ g_arg ... ]  
)  
=> t
```

Description

Sends an informational message to the calling application.

Arguments

<i>o_netlister</i>	The netlister object.
<i>t_info</i>	The message description, a format string.
<i>g_arg</i>	Arguments used for printing with the format string.

Value Returned

<i>t</i>	Always returns <i>t</i> .

Examples

The following example sends the given message to the calling application:

```
nlInfo(nl "My info message.\n")  
=> t
```

nlSetOption

```
nlSetOption(  
    o_netlister  
    s_option  
    g_value  
)  
=> t / nil
```

Description

Sets an option value.

This function can only be called during initialization. If it is called during netlisting, an error results.

Arguments

<i>o_netlister</i>	The netlister object.
<i>s_option</i>	The name of the option.
<i>g_value</i>	The value to which the option is set. The value type must be appropriate for the option name.

Value Returned

<i>t</i>	The option value is set.
<i>nil</i>	There is an error.

Examples

The following examples sets values for the given options:

```
nlSetOption( netlister 'maxNameLength 1024)  
nlSetOption( netlister 'hierarchyDelimiter ".")  
=> t
```

nlWarning

```
nlWarning(  
    o_netlist  
    t_warning  
    [ g_arg ... ]  
)  
=> t
```

Description

Issue a warning to the user.

Arguments

<i>o_netlist</i>	The netlist object.
<i>t_warning</i>	The description of the warning message in the form of a format string.
<i>g_arg</i>	Arguments used for printing with the format string.

Value Returned

<i>t</i>	Always returns <i>t</i> .

Examples

The following example displays a warning message as specified:

```
nlWarning(nl "My warning message.\n")  
=> t
```

nlPrintComment

```
nlPrintComment(  
    o_netlister  
    [ t_arg1 t_arg2 ... ]  
)  
=> t / nil
```

Description

Prints a comment.

Use this method to print all comments. This method uses the comment character and line wrapping. A subsequent `nlPrintf` call inserts the end comment string.

Arguments

<i>o_netlister</i>	The netlister object.
<i>t_arg1, t_arg2</i>	Arguments to formatted output.

Value Returned

<i>t</i>	The comment is printed.
<i>nil</i>	There is an error.

Examples

The following example displays a comment as specified:

```
nlPrintComment( netlister "My comment" )  
=> t
```

nlPrintIndentString

```
nlPrintIndentString(  
    o_netlist  
)  
=> t / nil
```

Description

Prints the indent string for the instance statement. When inside the top-level circuit, the empty string is printed. When inside subcircuit definitions, the value of the `subcktIndentString` netlist option is printed.

Arguments

<code>o_netlist</code>	The netlist object.

Value Returned

<code>t</code>	The indent string is printed.
<code>nil</code>	There is an error.

Examples

The following example displays an indent string for the instance statement:

```
nlPrintIndentString( netlist )  
=> t
```

nlPrintString

```
nlPrintString(  
    o_netlister,  
    [ @rest t_args ]  
)  
=> t / nil
```

Description

Prints the string arguments to the file. This function does the required line folding, prefixing, and, postfixing.

Arguments

<i>o_netlister</i>	The netlister object.
<i>@rest t_args</i>	The strings to be printed.

Value Returned

<i>t</i>	The string argument is printed.
<i>nil</i>	There is an error.

Examples

The following example displays a string argument to the file:

```
nlPrintString( netlister, "my string\n" )  
=> t
```


nlPrintStringNoFold

```
nlPrintStringNoFold(  
    o_netlist,  
    [ @rest t_args ]  
)  
=> t / nil
```

Description

Prints the string arguments to the file, like the function `nlPrintString`. This function does the required prefixing and postfixing, but does not fold the line until a newline character or `maxLength` is encountered. While in a 'no fold' print mode, calls cannot be made to `nlPrintString`.

Arguments

<i>o_netlist</i>	The netlist object.
@rest <i>t_args</i>	The strings to be printed.

Value Returned

<i>t</i>	The string argument is printed.
<i>nil</i>	There is an error.

Examples

The following example displays an indent string for the instance statement:

```
nlPrintStringNoFold( netlist, "my string\n" )  
=> t
```

Methods for Instances

During netlisting, instances are represented by objects. These objects cannot be redefined by the simulator interface. You can use the following methods to manage instances:

nlIsModelNameInherited

nlGetFormatter

nlGetSimName

nlGetSignalList

nlGetTerminalList

nlGetTerminalSignalName

nlGetNumberOfBits

nlGetModelName

nlGetParamList

nlGetParamStringValue

nlGetId

nlIncludeSrcFile

nlPrintComments

hnlGetInstanceCount



Do not redefine methods for the above functions.

nlIsModelNameInherited

```
nlIsModelNameInherited(  
    o_instance  
)  
=> t / nil
```

Description

Returns `t` if the model name for the stopping instance is passed through the hierarchy through parameters.

Arguments

<code>o_instance</code>	The instance object.

Value Returned

<code>t</code>	The model name will be passed from a higher level.
<code>nil</code>	The model name is specified at the instance.

Examples

```
nlIsModelNameInherited( inst )  
=> t
```

nlGetFormatter

```
nlGetFormatter(  
    o_instance  
)  
=> o_formatter
```

Description

Returns the formatter.

Arguments

<i>o_instance</i>	The instance object.

Value Returned

<i>o_formatter</i>	The formatter object used for netlisting.

Examples

```
nlGetFormatter( inst )  
=> obj1
```

nlGetSimName

```
nlGetSimName (
    o_instance
)
=> t_name

OR

nlGetSimName (
    o_cellView
)
=> t_simName
```

Description

If the input type is cellview object, then returns the simulator name of the subcircuit. If the input type is instance object, then returns the mapped name of the instance. The name returned depends on the `useInstNamePrefix` netlist option. When it is not set, the name prefix is not taken into account. However, for instances representing interface elements, the name prefix is always taken into account. No mapping is performed for interface elements.

For instance object, when the `useInstNamePrefix` netlist option is set, this method takes the name prefix into account. The prefix of a subcircuit does not have to be specified on the CDF of each subcircuit. If the prefix is specified on the CDF, it is used. Otherwise, the `subcktInstPrefix` netlist option is used.

Arguments

<i>o_instance</i>	The instance object.
<i>o_cellView</i>	The cellview object.

Value Returned

<i>t_name</i>	The mapped instance name if the input type is instance object.
<i>t_simName</i>	The simulator name if the input type is cellview object.

Examples

```
nlGetSimName( inst )
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

```
=> myInstance  
nlGetSimName( cv )  
=> myCellview
```

nlGetSignalList

```
nlGetSignalList(  
    o_instance  
)  
=> l_signals
```

Description

Returns a list of mapped signal names for the instance according to the terminal order specified for the cellview.

Use this method for printing instances to the netlist by the `nlPrintInst` method of the formatter. The terminal order for schematic subcircuits is determined by the `pin order` property on the schematic, or by the `termOrder` property on the CDF, or by the system, in that order.

Note: Ordering requirements, such as the Verilog[®] requirement that outputs occur before inputs, are not satisfied in this release.

Signal buses are handled in scalar form. For example, `net10<0:3>` is mapped as `net10_0`, `net10_1`, `net10_2`, and `net10_3`.

Arguments

<i>o_instance</i>	The instance object.

Value Returned

<i>l_signals</i>	The list of mapped signals for the instance.

Examples

```
nlGetSignalList( inst )
```

nlGetTerminalList

```
nlGetTerminalList(  
    o_instance  
)  
=> l_terminals
```

Description

Returns a list of terminal names in the order specified on the pin order property on the schematic, or on the `termOrder` property on the CDF, or on the cellview of the instance, in that order.

This method should not be used by the formatter. In contrast to `nlGetSignalList`, buses are not handled individually: a terminal such as `out<0:3>` is represented in its original form.

Arguments

<code>o_instance</code>	The instance object.

Value Returned

<code>l_terminals</code>	A list of terminals of the instance.

Examples

```
nlGetTerminalList( inst )
```


nlGetTerminalSignalName

```
nlGetTerminalSignalName(  
    o_instance  
    t_terminal  
    [ x_bit ]  
)  
=> t_signal
```

Description

Returns the name of the signal connected to the terminal.

Arguments

<i>o_instance</i>	The instance object.
<i>t_terminal</i>	The schematic name of the terminal.
<i>x_bit</i>	A non-negative integer value representing the bit number. The default is 0.

Value Returned

<i>t_signal</i>	The simulator name of the signal.

Examples

```
nlGetTerminalSignalName( inst "in" )
```

nlGetNumberOfBits

```
nlGetNumberOfBits(  
    o_instance  
    t_terminal  
)  
=> x_bits
```

Description

Returns the number of bits on the instance and terminal specified.

Arguments

<i>o_instance</i>	The instance object.
<i>t_terminal</i>	The schematic name of the terminal.

Value Returned

<i>x_bits</i>	A non-negative integer value representing the bit number.

Examples

```
nlGetNumberOfBits( inst "in")
```

nlGetModelName

```
nlGetModelName (
    o_instance
)
=> t_modelName
```

Description

This method must be used to obtain the model name of an instance. Use of this method assures consistency in netlisting across interfaces. For instances that represent subcircuits, this method returns a name chosen by the netlister, using the `modulePrefix` netlist option.

The method `nlGetModelName` returns the following value:

- The value of the model parameter on the instance, if this instance has CDF and if this parameter has a value.
- The value of the `modelName` parameter on the instance if this instance has no CDF and if this parameter has a value.
- The `componentName` entry on the view-specific information or on the simulator section on the `simInfo` section of the instance CDF if this has a value.
- The name of the cell.

These rules have consequences for the `simInfo` section for cells of which the views are used as stop cell-views. A number of simulators such as Spectre require a component or model name to define the type of component. An example of a component name is `resistor`, and an example of a model name is `nmos101`. This method addresses both model names and component names. For these simulators the `useInstNamePrefix` option is set to `nil`. For the "spectre" interface the `analogLib res` cell has a `componentName` entry set to "resistor". When the model name is not specified on an instance of a `res` cell, this value is used.

Virtuoso ADE SKILL Reference

Core Netlister Functions

Arguments

o_instance The instance object.

Value Returne

t_modelName Name of the model.

Examples

```
nlGetModelName( inst )
```

nlGetParamList

```
nlGetParamList(  
    o_instance  
)  
=> l_parameters
```

OR

```
nlGetParamList(  
    o_cellView  
)  
=> l_parameters
```

Description

Returns the list of parameters of the specified instance or cellview.

These parameters have been collected while the netlist was generated. In other words, all `pPar` and `atPar` expressions are collected. In addition, any `atPar` expressions from lower levels that are not resolved in the cellview are collected. Any interdependence of default values is handled by ordering the parameters so that those with independent default values are first.

Arguments

<i>o_instance</i>	The instance object.
<i>o_cellView</i>	The cellview object.

Value Returned

<i>l_parameters</i>	A list of parameters. Each element is a symbol, representing the name of the parameter.

Examples

```
nlGetParamList( inst )  
nlGetParamList( cv )
```

nlGetParamStringValue

```
nlGetParamStringValue(  
    o_instance  
    s_parameter  
)  
=> t_value / nil
```

Description

Returns a string representing the parameter value for the instance and parameter name.

The parameter name is the simulator name for the parameter. Values returned are strings irrespective of their original type. If the parameter does not exist or if the value is a blank string, `nil` is returned.

Value `none` is no longer recognized as a special keyword. Previously it was treated as being equivalent to `nil`, but now it is treated as a design variable.

The parameters are evaluated according to the common evaluator. For details, see Chapter 9 of the Compatibility Guide. This means that values of CDF parameters can reference parameters with `iPar` or `pPar` that have NLP expressions. The CDF default value can be overridden with an instance property. The property type of that property must be a string.

The following table shows a synopsis of that scheme:

This property...	Is CDF?	And evaluates to...
NLP	yes	nil
NLP	no	As NLP
string	yes	As specified on the CDF
string	no	The literal string

If the parameter is a CDF parameter, this function takes all aspects into account, including the CDF parameter type and the `parseAsCEL` attribute.

This procedure handles all look-up for `pPar`, `iPar`, and `atPar`, as well as the parameter name map specified for the component.

Virtuoso ADE SKILL Reference

Core Netlist Functions

The AEL `pPar` and the NLP “+” calls are replaced by the enclosed variable. The passed parameter found is automatically collected. The netlist tracks these parameters for appropriate invalidation purposes.

The AEL `iPar` and the NLP “~” calls are replaced by the appropriate property values.

The use of the AEL `atPar` function and the NLP “@” calls is discouraged. Like `pPar`, these are replaced with the variable specified, and the parameter specified is declared as a passed parameter in the subcircuit.

Important

The AEL `dotPar` function and the NLP “.” are not supported, and a netlist error is reported if they are used.

The suffixes are substituted according to the suffix mapping specified. Thus, a schematic value “1n” may result in “1e-9”.

Note: Only a limited set of NLP expressions is translated appropriately. The format accepted is limited to:

```
[<operator> <parameter name>],  
[<operator> <parameter name>: % ],  
[<operator> <parameter name>: % : <default value>],
```

with the operator being @, +, or ~.

Many NLP expressions are therefore not supported:

- Complex formatting such as :abc%d%e
- Formatting on the default clause
- Modifiers in front of the operator, such as time and capacitance scaling
- Nesting in the default value or value formatting section

For about `iPar`, `pPar`, `atPar` and `dotPar` functions, see [Passing Parameters in a Design](#).

Virtuoso ADE SKILL Reference

Core Netlist Functions

Arguments

<i>o_instance</i>	The instance object.
<i>s_parameter</i>	The name of the parameter.

Value Returned

<i>t_value</i>	The string form of the parameter value.
<i>nil</i>	The parameter was not found, the value was <code>nil</code> , the value was an empty string, or the value was a string consisting of only white space.

Examples

```
nlGetParamStringValue( inst 'tvpairs )
```


nlGetId

```
nlGetId(  
    o_cellView  
)  
=> g_id
```

OR

```
nlGetId(  
    o_instance  
)  
=> t_id
```

Description

Returns the database ID for the instance. If the instance represents a cdba instance, this is a database ID.

Avoid using this function if possible; use type-checking precautions.

Arguments

<i>o_cellView</i>	The cellview object.
<i>o_instance</i>	The instance object.

Value Returned

<i>t_id</i>	The database ID of the instance or the cellview object.

Examples

```
nlGetId( inst )  
nlGetId( cv )
```

nlIncludeSrcFile

```
nlIncludeSrcFile(  
    o_formatter  
    t_filename  
)  
=> t / nil
```

Description

Prints the include statement for instances bound to source files using Hierarchy Editor. This is called while printing the footer for the netlist for all the source files to which any instance in the design is bound. If your simulator does not support source file bindings, call [nlError](#) in this method.

Arguments

<i>o_formatter</i>	The formatter object.
<i>t_filename</i>	The name of the source file bound to an instance.

Value Returned

t	The include statement is printed.
nil	The operation failed.

Examples

```
(defmethod nlIncludeSrcFile ((obj <yourSimulator>Formatter) file)  
  (let ((nl (nlGetNetlist obj)))  
    (nlPrintStringNoFold nl "include '" simplifyFilename(file) "'")  
    (nlPrintStringNoFold nl "\n")  
  )  
=> t
```

Related Topics

[nlError](#)

nlPrintComments

```
nlPrintComments(  
    o_instance  
)  
=> t
```

Description

Prints the comments for the instance being netlisted.

Arguments

<i>o_instance</i>	The instance object.

Value Returned

t	The comments for the instance are printed.

Examples

```
nlPrintComments( inst )  
=> t
```

hnlGetInstanceCount

```
hnlGetInstanceCount(  
    )  
=> x_numberOfInstances
```

Description

Returns the number of instances in the design most recently netlisted in the same session. This does not include instances with the `nlAction=ignore` property that are ignored during netlisting.

Arguments

None

Value Returned

<i>x_numberOfInstances</i>	The number of instances in the design most recently netlisted in the same session.

Examples

```
(defmethod nlIncludeSrcFile ((obj <yourSimulator>Formatter) file)  
  (let ((nl (nlGetNetlister obj)))  
    (nlPrintStringNoFold nl "include '" simplifyFilename(file) "'")  
    (nlPrintStringNoFold nl "\n")  
  )  
)  
=> 5
```

Cellviews

Cellviews are represented by an object. The `nlGetCurrentCellView` method of the `nlFormatter` class returns such an object. This object is used by formatter methods such as `nlPrintSubcktName` and `nlPrintSubcktParameters`. It is primarily used in the `nlPrintSubcktHeader` and `nlPrintSubcktFooter` methods of the formatter.



Do not redefine any of these methods.

`nlGetCellName`

`nlGetLibName`

`nlGetSimTerminalNets`

`nlGetTerminalNets`

`nlGetSwitchViewList`

`nlGetViewName`

nlGetCellName

```
nlGetCellName(  
    o_inst  
)  
=> t_cellName
```

Description

Gets the cell name of the specified instance.

Arguments

<i>o_inst</i>	The instance object.

Value Returned

<i>t_cellName</i>	The cell name given as a string.

Examples

```
printf("Cell name is %s.\n" nlGetCellName(inst))  
=> myCell
```

nlGetLibName

```
nlGetLibName(  
    o_inst  
)  
=> t_libName
```

Description

Gets the library name of the specified instance.

Argument

<i>o_inst</i>	The instance object.

Value Returned

<i>t_libName</i>	The library name given as a string.

Examples

```
printf("Library name is %s.\n" nlGetLibName(inst))  
=> myLib
```

nlGetSimTerminalNets

```
nlGetSimTerminalNets(  
    o_cellView  
)  
=> l_signals
```

Description

Returns the list of mapped names of signals connecting to the terminals of the cellview based on the terminal order specified for the cellview. Several signals may come from inherited connections. Signal buses are handled in scalar form.

For example, `net10<0:3>` is mapped as `net10_0`, `net10_1`, `net10_2`, and `net10_3`.

Use this function while printing the subcircuit definition.

Arguments

<i>o_cellView</i>	The cellview object.

Value Returned

<i>l_signals</i>	A list of mapped signal names connected to the terminals of the subcircuit.

Examples

```
nlGetSimTerminalNets( cv )
```


nlGetTerminalNets

```
nlGetTerminalNets(  
    o_cellView  
)  
=> l_signals
```

Description

Returns the schematic names of the signals connected to the terminals. Many of the signals may come from inherited connections.

Arguments

<i>o_cellView</i>	The cellview object.

Value Returned

<i>l_signals</i>	A list of signals connected to the terminals of the subcircuit. Each element is a string.

Examples

```
nlGetTerminalNets( cv )
```

nlGetSwitchViewList

```
nlGetSwitchViewList(  
    o_cellView  
)  
=> l_switchViews
```

Description

Returns the switch view list for the cellView.

Arguments

<i>o_cellView</i>	The cellview object.

Value Returned

<i>l_switchViews</i>	A switch view list for the cell view. Each element is a string.

Examples

```
nlGetSwitchViewList( cv )
```

nlGetViewName

```
nlGetViewName(  
    o_cellView  
)  
=> t_cellViewName
```

Description

Returns the view name of the specified cellview object.

Arguments

<i>o_cellView</i>	The cellview object.

Value Returned

<i>t_cellViewName</i>	The view name of the specified celview object.

Examples

```
nlGetViewName( cv )  
=> "schematic"
```

Designs

Designs are represented by an object. The `nlGetDesign` method applied on the `netlister` object returns such an object.



Do not redefine any of these methods.

`nlGetTopLibName`

`nlGetTopCellName`

`nlGetTopViewName`

`nlTranslateFlatIEPathName`

nlGetTopLibName

```
nlGetTopLibName (
    o_design
)
=> t_topLibName
```

Description

Returns the library name of the design.

Arguments

<i>o_design</i>	The design object.

Value Returned

<i>t_topLibName</i>	A library name.

Examples

```
nlGetTopLibName( design )
```

nlGetTopCellName

```
nlGetTopCellName (
    o_design
)
=> t_topCellName
```

Description

Returns the cell name of the design.

Arguments

<i>o_design</i>	The design object.

Value Returned

<i>t_topCellName</i>	A cell name.

Examples

```
nlGetTopCellName ( design )
```

nlGetTopViewName

```
nlGetTopViewName (
    o_design
)
=> t_topViewName
```

Description

Returns the view name of the design.

Arguments

<i>o_design</i>	The design object.

Value Returned

<i>t_topViewName</i>	A view name.

Examples

```
nlGetTopViewName ( design )
```

nlTranslateFlatIEPathName

```
nlTranslateFlatIEPathName(  
    o_formatter  
    t_hierDelimiter  
    t_iePathName  
)  
=> t_iePathName
```

Description

Parses a hierarchical IE instance path and returns the path of the IE instance that is to be printed in the digital netlist.

Arguments

<i>o_formatter</i>	The formatter object.
<i>t_hierDelimiter</i>	The delimiter used to parse the hierarchical IE instance path.
<i>t_iePathName</i>	The hierarchical IE instance path.

Value Returned

<i>t_iePathName</i>	The path of the IE instance to be printed in the digital netlist.

Examples

```
defmethod( nlTranslateFlatIEPathName ((formatter yourFormatter) hierDelimiter  
iePathName)  
iePathComponents = parseString(iePathName hierDelimiter)  
yourFunctionToModifyIEPathName(iePathComponents)  
)
```


Other Customization Procedures

You can use the following functions to define other customization procedures:

nlSetPcellName

ansCdICompPrim

ansCdIHnIPrintInst

ansCdIPrintString

ansCdIPrintInheritedParams

ansCdIPrintInstParams

ansCdIPrintInstProps

ansCdIPrintInstName

ansCdIPrintModelName

ansCdIPrintModuleName

ansCdIPrintConnections

ansCdIGetSegmentConnections

ansCdIPrintSwitchPCellInst

ansCdIPrintSwitchPCellInstParam

ansCdIPrintSwitchPCellSubcktConn

ansCdIPrintSwitchPCellSubCircuit

ansCdIGetSegmentInfo

ansCdIGetSegmentInstParams

ansCdIGetSimPropValue

ansCdIGetMultiplicity

auCdI

nlSetPcellName

```
nlSetPcellName (
    S_cv
    t_paramNames
    g_paramValues
)
=> t / nil
```

Description

Define this function if the default generic OSS naming convention for Pcells needs to be customized.

Arguments

<i>S_cv</i>	cellView ID of the Pcell.
<i>t_paramNames</i>	Pcell parameter Names.
<i>g_paramValues</i>	Pcell parameter Values.

Value Returned

t	The default OSS name is changed.
nil	The command cannot be run.

Examples

Redefine this function to rename pcell subckt. In the example below, subckt is renamed to paramName1_paramValue1_paramName2_paramValue2_cellName:

```
procedure( nlSetPcellName( cv paramNames paramValues )
    let( ( name value (subcktName ""))
    foreach( (name value) paramNames paramValues
        subcktName = strcat( subcktName name "_" value "_" )
    )
    strcat(subcktName cv->cellName )
    )
=> t
```

ansCdlCompPrim

```
ansCdlCompPrim(  
    )
```

Description

Enables printing of device information for primitives in the `auCdl` netlist. Specify this function as a netlist procedure in the CDF for the primitive devices for which you want the device information to be printed in the `auCdl` netlist.

Arguments

None

Value Returned

None

Examples

```
ansCdlCompPrim
```

ansCdlHnlPrintInst

```
ansCdlHnlPrintInst(  
    )
```

Description

Customizes how device information is written in the auCdl netlist. Specify this function as a netlist procedure in the CDF for the devices for which you want to customize the device information in the auCdl netlist.

Arguments

None

Value Returned

None

Examples

```
ansCdlHnlPrintInst
```

ansCdlPrintString

```
ansCdlPrintString(  
    &_fp  
    S_inst  
    S_master  
    S_parent  
)
```

Description

Prints comment strings in the device information for instances in the auCdl netlist. To print comment strings in the device information, you must also use the 'string argument in the auCdlInstPrintOrder variable defined in the .simrc file. For more information about the auCdlInstPrintOrder variable, see the *Virtuoso ADE Explorer User Guide*.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the artFprintf SKILL function.
<i>S_inst</i>	Database ID of the current instance being printed in the netlist.
<i>S_master</i>	Database ID of the switch master of the current instance being printed in the netlist.
<i>S_parent</i>	Database ID of the cell for the current instance being printed in the netlist.

Value Returned

None

Examples

```
procedure(ansCdlPrintString(fp inst master parent)  
artFprintf(fp "*** CurrentInst = %s" inst~>name))
```

ansCdlPrintInheritedParams

```
ansCdlPrintInheritedParams(  
    &_fp  
    l_pairList  
)
```

Description

Customizes how inherited parameters are written in the auCdl netlist.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>l_pairList</i>	List of property name and property value pairs that should be evaluated.

Value Returned

None

Examples

```
procedure(ansCdlPrintInheritedParams(fp pairList)  
foreach( param pairList  
unless( !cadr(param)  
artFprintf( fp "%s=%s " car(param) artMakeString(cadr(param) ) )  
)  
)  
)
```

ansCdlPrintInstParams

```
ansCdlPrintInstParams(  
    &_fp  
    l_pairList  
)
```

Description

Customizes how instance parameters are written in the `auCdl` netlist.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>l_pairList</i>	List of property name and property value pairs that should be evaluated.

Value Returned

None

Examples

```
procedure( ansCdlPrintInstParams(fp pairList)  
    foreach( pair pairList  
        artFprintf( fp "%s=%s " car(pair) artMakeString(cadr(pair)) )  
    )
```

ansCdlPrintInstProps

```
ansCdlPrintInstProps (
    &_fp
    l_pairList
)
```

Description

Enables printing of user-defined instance properties and also customizes the format in which the properties are printed in the netlist.

To print user-defined properties in the netlist, you must also use the `'instProps` argument in the `auCdlInstPrintOrder` variable defined in the `.simrc` file.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>l_pairList</i>	List of property name and property value pairs that should be evaluated.

Value Returned

None

Examples

```
procedure( ansCdlPrintInstProps(fp pairList)
    foreach( pair pairList
        artFprintf( fp "%s=%s " car(pair) artMakeString(cadr(pair)) )
    )
)
```


ansCdlPrintInstName

```
ansCdlPrintInstName(  
    &_fp  
    t_prefix  
    t_name  
    t_mappedName  
    g_isPrimitive  
    S_inst  
    S_master  
)
```

Description

Customizes how instance names are written in the `auCdl` netlist.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>t_prefix</i>	Name prefix defined in the device CDF.
<i>t_name</i>	Instance name specified for the instance.
<i>t_mappedName</i>	Mapped name which <code>auCdl</code> would otherwise use to print instance name.
<i>g_isPrimitive</i>	Whether this is an instance of a leaf-level device (primitive).
<i>S_inst</i>	Database ID of the current instance being printed in the netlist.
<i>S_master</i>	Database ID of the switch master of the current instance being printed in the netlist.

Value Returned

None

Examples

```
procedure( ansCdlPrintInstName(fp prefix name mappedName isPrimitive inst master)  
    artFprintf(fp "%s " mappedName ))
```

ansCdlPrintModelName

```
ansCdlPrintModelName(  
    &_fp  
    g_isAPrimitive  
    g_definedPropVal  
    g_modelPropInstVal  
    g_componentPropInstVal  
    g_cdfModelName  
    g_cdfComponentName  
)
```

Description

Customizes the order in which `auCdl` looks for model names for primitives and the format in which the model information is written in the netlist.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>g_isAPrimitive</i>	Whether this is an instance of a leaf-level device(primitive).
<i>g_definedPropVal</i>	Value of the instance property defined using <code>auCdlHn1InstModelPropName</code> parameter in the <code>.simrc</code> file.
<i>g_modelPropInstVal</i>	Value of the 'model property on the instance.
<i>g_componentPropInstVal</i>	Value of the 'componentName property on the instance.
<i>g_cdfModelName</i>	Value of the 'modelName parameter in the device CDF.
<i>g_cdfComponentName</i>	Value of the 'componentName parameter in the device CDF.

Value Returned

None

Virtuoso ADE SKILL Reference

Core Netlister Functions

Examples

```
procedure( ansCdlPrintModelName(fp isAPrimitive definedPropVal
                                modelPropInstVal
                                componentPropInstVal
                                cdfModelName
                                cdfComponentName
                                )
  when( isAPrimitive
    let( (model)
      model = nil
      if( definedPropVal
        model = definedPropVal
      )
      if( !model && modelPropInstVal
        model = definedPropVal
      )
      if( !model && cdfModelName
        model = cdfModelName
      )
      if( !model && componentPropInstVal
        model = componentPropInstVal
      )
      if( !model && cdfComponentName
        model = cdfComponentName
      )
      artFprintf(fp "model=%s " artMakeString( model ) )
    )
  ))
```

ansCdlPrintModuleName

```
ansCdlPrintModuleName(  
    &_fp  
    g_isAPrimitive  
    S_inst  
    S_master  
    S_parent  
    t_appedModuleName  
)
```

Description

Customizes how module names are written in the auCdl netlist for subcircuits.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>g_isAPrimitive</i>	Whether this is an instance of a leaf-level device(primitive).
<i>S_inst</i>	Database ID of the current instance being printed in the netlist.
<i>S_master</i>	Database ID of the master device for the current instance being printed in the netlist.
<i>S_parent</i>	Database ID of the cell for the current instance being printed in the netlist.
<i>t_mappedModuleName</i>	Module name which auCdl would write by default for subcircuits.

Value Returned

None

Examples

```
procedure( ansCdlPrintModuleName(fp isAPrimitive inst master parent  
mappedModuleName)  
    unless( isAPrimitive  
        artFprintf( fp " / %s " mappedModuleName )  
    )
```

ansCdlPrintConnections

```
ansCdlPrintConnections(  
    &_fp  
    l_connections  
)
```

Description

Customizes how the nets connected to a device are written in the `auCdl` netlist.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.
<i>l_connections</i>	List of pairs of instance terminals and nets connected to the terminals.

Value Returned

None

Examples

```
procedure( ansCdlPrintConnections(fp connections )  
    artFprintf( fp "$PINS " )  
    foreach( cn connections  
        artFprintf( fp "%s=%s " car(cn) cadr(cn) ) ))
```

ansCdlGetSegmentConnections

```
ansCdlGetSegmentConnections (  
    S_inst  
    l_connectionPairs  
    n_iterSeg  
    n_iterMult  
    n_numSegments  
    n_multiplicityFactor  
    t_segmentConnType  
    n_netCount  
)
```

Description

This function is used to customize the `auCdl` netlist when the `ansCdlHnlPrintInst` function is specified as a netlist procedure in the CDF for the device. It controls how connectivity information is written in the netlist for instances for which a multiplicity factor is specified using the `m` or `M` property. For example, if an instance with a connection list like `((termA netA) (termB netB))` has to be converted into two segments connected in series, the modified connection list for the first segment will be `((termA netA) (termB tempnet_0))` and `((termA tempnet_0) (termB netB))` for the second segment. Define this function as a procedure in the `.simrc` file.

Virtuoso ADE SKILL Reference

Core Netlister Functions

Arguments

<i>s_inst</i>	Database ID of original instance.
<i>l_connectionPairs</i>	List of pairs of instance terminals and nets connected to the terminals of the instance.
<i>n_iterSeg</i>	If current replication needs to be converted into 3 segments connected in series or parallel, then this function is called for which iteration is from 1- 3.
<i>n_iterMult</i>	If the current instance needs to be converted into 5 instances connected in parallel, then this function is called for which iteration is from 1 - 5.
<i>n_numSegments</i>	Number of segments in each replication of the instance.
<i>n_multiplicityFactor</i>	Number of replications of the instance.
<i>t_segmentConnType</i>	Connection type of the segments. Valid values: "series" or "parallel" or any other user defined value.
<i>n_netCount</i>	A number that keeps on incrementing with calls to this function. This will help you create temporary nets to connect segments in series.

Value Returned

None

Examples

```
procedure( ansCdlGetSegmentConnections( inst connectionPairs iterSeg iterMult
numSegments multiplicityFactor segmentConnType netCount )
  let( ( firstTerm secondTerm pairList )
    if( "series" == segmentConnType
      then
        cond(
          ( 1 == iterSeg )
            secondTerm = cadr( connectionPairs )
            secondTerm = list( car(secondTerm) sprintf( nil "CdnsNet_%d"
              netCount) )
            pairList = list( car(connectionPairs) secondTerm )
            pairList = append( pairList cddr(connectionPairs) )
          )
        ( ( iterSeg == numSegments )
          firstTerm = car(connectionPairs)
          firstTerm = list( car(firstTerm) sprintf( nil "%s_%d" "CdnsNet"
            netCount-1 ) )
```

Virtuoso ADE SKILL Reference

Core Netlist Functions

```
pairList = append( list(firstTerm) cdr(connectionPairs))
)
(t
firstTerm = car(connectionPairs)
secondTerm = cadr(connectionPairs)
firstTerm = list( car(firstTerm) sprintf(nil "CdnsNet_%d"
netCount-1) )
secondTerm = list( car(secondTerm) sprintf(nil "CdnsNet_%d"
netCount) )
pairList = append1( list(firstTerm) secondTerm )
pairList = append( pairList cddr(connectionPairs))
)
)
else
pairList = connectionPairs
)
pairList
)
)
```

Related Topics

[ansCdlGetMultiplicity](#)

[ansCdlGetSegmentInfo](#)

ansCdIPrintSwitchPCellInst

```
ansCdIPrintSwitchPCellInst(  
    )
```

Description

Writes the switch Pcell instance in the `auCd1` netlist based on the size and input values of the instance.

Arguments

None

Value Returned

None

Examples

```
XI0_31 a out switch_open lay="M3"  
XI0_32 b out switch_close lay="M3"  
XI0_33 b out switch_open lay="M3"
```

Schematic instance `I0` with size value 3 and input value 2 is written in the `auCd1` netlist.

ansCdlPrintSwitchPCellInstParam

```
ansCdlPrintSwitchPCellInstParam(  
    &_fp  
    l_paramsNameValuePairList  
)
```

Description

Writes into the `auCdl` netlist the list of CDF parameters and values defined for switch Pcell instances.

Arguments

&_fp File handle used to write the string in the netlist using the `artFprintf` SKILL function.

l_paramsNameValuePairList List of pairs of parameter names and parameter values.

Value Returned

None

Examples

You can use `ansCdlPrintSwitchPCellInstParam` to print the parameter names and values as needed. The following example prints the value of the parameter `metal` in double quotes:

```
procedure(anCdlPrintSwitchPCellInstParam( fp pairList )  
    let( ( paramName )  
        foreach( pair pairList  
            paramName = car(pair)  
            if( paramName == "metal" then  
                artFprintf( fp "%s=\"%s\" " paramName artMakeString(cadr(pair)) )  
            else  
                artFprintf( fp "%s=%s " paramName artMakeString(cadr(pair)) )  
            )  
        )  
    )  
)
```

ansCdlPrintSwitchPCellSubcktConn

```
ansCdlPrintSwitchPCellSubcktConn(  
    &_fp  
    l_connections  
)
```

Description

Writes into the auCdl netlist the switch Pcell connections specified by hnlSpecialPCellLibCell in the .simrc file.

Arguments

<i>&_fp</i>	File handle used to write the string in the netlist using the artFprintf SKILL function.
<i>l_connections</i>	List of terminals specified for the switch Pcells.

Value Returned

None

Examples

Consider that hnlSpecialPCellLibCell is defined in .simrc as shown below:

```
hnlSpecialPCellLibCell = list( list("lib1" "SW") list("lib2" "cell2"))
```

By default, the value of connections is a list ("POS" "NEG") and subckt definition for the cell SW and cell2 is as below.

```
.SUBCKT SW_close POS NEG  
.ENDS
```

```
.SUBCKT SW_open POS NEG  
.ENDS
```

```
.SUBCKT cell2_close POS NEG  
.ENDS
```

```
.SUBCKT cell2_open POS NEG  
.ENDS
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

You can use the following procedure to modify connections.

```
procedure(ansCdlPrintSwitchPCellSubcktConn( fp connections )
    foreach( cn connections
        artFprintf( fp "%s " cn)
    )
)
```

ansCdIPrintSwitchPCellSubCircuit

```
ansCdIPrintSwitchPCellSubCircuit(  
    &_fp  
)
```

Description

Writes into the `auCdI` netlist the subckt definitions specified by `hnlSpecialPCellLibCell` in the `.simrc` file.

Arguments

<code>&_fp</code>	File handle used to write the string in the netlist using the <code>artFprintf</code> SKILL function.

Value Returned

None

Examples

Assume that `hnlSpecialPCellLibCell` is defined in `.simrc` as show below:

```
hnlSpecialPCellLibCell = list( list("lib1" "SW") list("lib2" "cell2"))
```

Following definitions for subckts are printed in the `auCDL` netlist:

```
.SUBCKT SW_close POS NEG  
.ENDS  
  
.SUBCKT SW_open POS NEG  
.ENDS  
  
.SUBCKT cell2_close POS NEG  
.ENDS  
  
.SUBCKT cell2_open POS NEG  
.ENDS
```

ansCdlGetSegmentInfo

```
ansCdlGetSegmentInfo(  
    S_inst  
    S_master  
    S_parent  
)  
=> l_segment
```

Description

This function is used to customize the auCdl netlist when the `ansCdlHnlPrintInst` function is specified as a netlist procedure in the CDF for the device. It returns the number of segments for an instance and the connection type (`series`, `parallel`, or user defined connection type name) between the segments. Define this function as a procedure in the `.simrc` file.

Arguments

<i>S_inst</i>	Database ID of the current instance being printed in the netlist.
<i>S_master</i>	Database ID of the switch master of the current instance being printed in the netlist.
<i>S_parent</i>	Database ID of the cell for the current instance being printed in the netlist.

Value Returned

<i>l_segment</i>	Number of segments for the instance and the connection type (<code>series</code> , <code>parallel</code> , or user defined) between segments.

Examples

```
procedure( ansCdlGetSegmentInfo( inst master parent)  
    prog( (val type )  
    val = ansCdlGetSimPropValue( 'segments )  
    when( val  
        when( stringp(val) val = atoi( val) )  
        when( numberp(val) val = fix(val) )  
    )  
    unless( val return() )  
    type = ansCdlGetSimPropValue( 'connection )  
    when( type && symbolp( type)  
        type = symbolToString( type ) )
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

```
        return( list( val type))  
    )
```

Related Topics

[ansCdlGetMultiplicity](#)

[ansCdlGetSegmentConnections](#)

ansCdlGetSegmentInstParams

```
ansCdlGetSegmentInstParams (
    S_inst
    l_propsList
    n_iterSeg
    n_iterMult
    n_numSegments
    n_multiplicityFactor
    t_segmentConnType
)
=> l_segment
```

Description

Customizes the `auCdl` netlist when the `ansCdlHnlPrintInst` function is specified as a netlist procedure in the CDF for the device.

It controls how values of parameters are printed on segments of an instance. `auCdl` calls this function with a list of property name and property value pairs for all those properties that are specified in device CDF simulation information instance parameters. Define this function as a procedure in the `.simrc` file.

Arguments

<i>S_inst</i>	Database ID of original instance.
<i>l_propsList</i>	List of property name and property value pairs that should be evaluated.
<i>n_iterSeg</i>	If current replication needs to be converted into 3 segments connected in series or parallel, then this function is called for which iteration is from 1- 3.
<i>n_iterMult</i>	If the current instance needs to be converted into 5 instances connected in parallel, then this function is called for which iteration is from 1 - 5.
<i>n_numSegments</i>	Number of segments in each replication of the instance.
<i>n_multiplicityFactor</i>	Number of replications of the instance.
<i>t_segmentConnType</i>	Connection type of the segments. Valid values: "series" or "parallel" or any other user defined value.

Virtuoso ADE SKILL Reference

Core Netlist Functions

Value Returned

l_segment

A modified list of property name and property value pairs for each segment. By default, auCdl divides the length of the instance by the number of segments in case of series connection, and divides the width of the instance by the number of segments in case of parallel connection.

Examples

```
procedure( ansCdlGetSegmentInstParams( inst propsList iterSeg iterMult
  numSegments multiplicityFactor segmentConnType )
  let( (pairList value)
  foreach( pair propsList
    case( car(pair)
      ("l"
        value = cadr(pair)
        when( "series" == segmentConnType
          when( stringp(value)
            value = atoi(value) )
            value = value/numSegments )
        pairList = cons( list( "l" value) pairList )
      )
      ("w"
        value = cadr(pair )
        when( "parallel" == segmentConnType
          when( stringp(value)
            value = atoi(value))
            value = value/numSegments )
        pairList = cons( list( "w" value ) pairList )
      )
      (t
        pairList = cons( pair pairList )
      )
    )
  pairList = reverse( pairList )
  )
)
```

Related Topics

[ansCdlGetMultiplicity](#)

[ansCdlGetSegmentInfo](#)

ansCdlGetSimPropValue

```
ansCdlGetSimPropValue (
    t_propName
)
=> t_propVal / nil
```

Description

Returns the value of a specified property on the current instance being netlisted when the `ansCdlHnlPrintInst` function is specified as a netlist procedure in the CDF for the device.

Arguments

<i>t_propName</i>	Name of the property on the current instance being netlisted for which the value is to be returned.

Value Returned

<i>t_propVal</i>	Value corresponding to a property.
<i>nil</i>	The command cannot be run.

Examples

```
ansCdlSimGetPropValue( 'vendorName '
```

ansCdlGetMultiplicity

```
ansCdlGetMultiplicity(  
    S_inst  
    S_master  
    S_parent  
)  
=> n_multiplicityFactor
```

Description

Controls how multiplicity (converting an instance into multiple instances connected in parallel) is handled for an instance in the design when the `ansCdlHnlPrintInst` function is specified as a netlist procedure in the CDF for the master of the instance.

You can specify that an instance should be treated as n instances connected in parallel by specifying n as the value of the `m` or `M` property on the instance. For example, you can specify that an instance be treated as five instances connected in parallel by specifying 5 as the value of the `m` or `M` property on the instance. By default the `ansCdlHnlPrintInst` netlist procedure does not give special treatment to the `m` or `M` property on an instance.

If you want the `ansCdlHnlPrintInst` netlist procedure to support multiplicity on instances using the `m` or `M` property, do one of the following:

- Define the `ansCdlGetMultiplicity` procedure in your `.simrc` file.
- Define `ansCdlGetMultiplicity_<LIBNAME>` in `libInit.il` file of the device library when the `libSpecificDevicePrint` flag is set in device `instParameters`.

Arguments

<i>S_inst</i>	Database ID of the current instance being printed in the netlist.
<i>S_master</i>	Database ID of the switch master of the current instance being printed in the netlist.
<i>S_parent</i>	Database ID of the cell for the current instance being printed in the netlist.

Virtuoso ADE SKILL Reference

Core Netlist Functions

Value Returned

n_multiplicityFactor Multiplicity factor.

Examples

```
procedure( ansCdlGetMultiplicity( inst master parent )
    ansCdlGetSimPropValue( 'm )
)
```

Related Topics

[ansCdlGetSegmentInfo](#)

[ansCdlGetSegmentConnections](#)

[ansCdlGetSegmentInstParams](#)

auCdl

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

Description

This function sets defaults of all the variables defined in `si.env` like `preserveRES`, `shortRES` as well as other global variables like `cdlSimViewList`, `cdlPrintComments` etc. The function also sets the list of functions and variables that must be unbound when environments(simulators) are changed.

Arguments

None

Value Returned

<code>t</code>	All the required defaults of variables as well as list of functions and variables are set.
<code>nil</code>	The command cannot be run.

Other Backend Netlister Functions

You can use the following backend netlister functions for further customizations:

[acdArtPrintIncludedNetlist](#)

[acdCurrentInstName](#)

[auLvs](#)

[auProbeAddDevsForNet](#)

[LVS](#)

acdlArtPrintIncludedNetlist

```
acdlArtPrintIncludedNetlist(  
    x_artOutfile_file_pointer  
    t_filePath  
)  
=> t_userSpecified
```

Description

Controls how an included netlist is included in the main netlist file. This function can be overridden by user settings. If user-override is not specified, the included netlist is copied without any modification. However, if both `acdlPrintIncludedNetlist` and `acdlArtPrintIncludedNetlist` are defined, `acdlPrintIncludedNetlist` is called. `acdlArtPrintIncludedNetlist` is preferred because the file pointer is not closed and re-initialized.

Arguments

x_artOutfile_file_pointer

An open file pointer to the main netlist.

You can write to this pointer using the `artFprintf` SKILL function.

t_filePath

Path to the netlist to be included.

Value Returned

t_userSpecified Value specified by the user.

Examples

```
;; (unless getd ... prevents multiple definition warning message.)  
(unless (getd 'acdlArtPrintIncludedNetlist)  
    ;; The identity function, Copy the input to the output without modification. If  
    you are overriding this function, you are likely doing some preprocessing or  
    decoration in artFprintf().  
    (define (acdlArtPrintIncludedNetlistBase fp filePath)  
        (let ((inport (infile filePath))  
              buffer)  
            while( gets( buffer inport )
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

```
        (artFprintf fp "%s" buffer )  
    )  
    t  
))
```


acdICurrentInstName

```
acdICurrentInstName(  
    )  
=> t_mappedInstName
```

Description

Maps instance names during hierarchical or flat netlisting.

You can control casing of names during hierarchical netlisting using the `instNameCase` `.cdsenv` environment variable in following way:

- Mapped names use mixed casing. This is the default setting.

```
auCdl.envOpts instNameCase string "mixed" nil
```

- Mapped names are in lower case.

```
auCdl.envOpts instNameCase string "lower" nil
```

- Mapped names are in upper case.

```
auCdl.envOpts instNameCase string "upper" nil ; give in upper case
```

Arguments

None

Value Returned

t_mappedInstName Name of the mapped instance.

Examples

```
acdICurrentInstName()
```

auLvs

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

Description

This is the primary function for LVS. It sets up all the actions needed to netlist the layout and schematics design as well as invoke LVS (for comparison) itself.

Argument

None

Value Returned.

t	The LVS function ran a comparison between the netlist from the schematic view and the layout view. Relevant messages are printed in the UI and the <code>si.log</code> file.
nil	Indicates an error. Relevant messages are printed in the UI and the <code>si.log</code> file.

Examples

```
auLvs()
```

auProbeAddDevsForNet

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

Description

This procedure enables you to select nets using the cursor or by typing the names in the CIW to add probes for all devices connected to the selected net. The function displays the prompt, `Point to net or enter net name in CIW`. On pointing to the net or typing the net name in CIW, probes would be added on all the devices connected to the selected net.

Argument

None

Value Returned

t	Probes are added for all devices connected to the selected net.
nil	Unsuccessful operation.

Examples

```
auProbeAddDevsForNet ()
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

LVS

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

Description

This is the primary function for LVS. It sets up all the actions needed to netlist the layout and schematics design as well as invoke LVS itself.

Arguments

None

Value Returned

t	All the required actions are set.
nil	Unsuccessful operation.

Examples

```
LVS ()
```

HSPICE Functions

This topic lists the SKILL functions that let you work with the HSPICE simulator.

[hnlHspicePrintInstPropVal](#)

[hnlHspiceInstPropVal](#)

[hnlHspicePrintInstPropEqVal](#)

[hnlHspicePrintMOSfetModel](#)

[hnlHspicePrintNMOSfetElement](#)

hnlHspicePrintInstPropVal

```
hnlHspicePrintInstPropVal(  
    t_propName  
)  
=> t_propVal / nil
```

Description

This procedure returns the value of the property specified by *propName* if it exists on the current instance. If the value of this property has the syntax specifying an inherited value it returns the name of the property whose value is being inherited without the surrounding syntax.

Arguments

<i>t_propName</i>	Specifies the property name.

Value Returned

<i>t_propVal</i>	The value corresponding to a property. The return value for this procedure is always a string.
<i>nil</i>	The command cannot be run.

Examples

```
hnlHspicePrintInstPropVal()  
=> prop1
```

hnlHspiceInstPropVal

```
hnlHspiceInstPropVal(  
    l_paramList  
)  
=> t / nil
```

Description

This procedure prints a list of property values to the netlist.

Arguments

<i>l_paramList</i>	List of strings containing the property names.

Value Returned

<i>t</i>	The property values are successfully printed to the netlist.
<i>nil</i>	The command cannot be run.

hnlHspicePrintInstPropEqVal

```
hnlHspicePrintInstPropEqVal(  
    l_paramList  
)  
=> t / nil
```

Description

This procedure prints a list of property values to the netlist. It is similar to the procedure `hnlHspicePrintInstPropVal` except that the property name and the symbol, = is included before the value.

Arguments

<i>l_paramList</i>	List of strings containing the property names.

Value Returned

<i>t</i>	The property values are printed to the netlist.
<i>nil</i>	The command cannot be run.

hnlHspicePrintMOSfetModel

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

Description

This function prints out the line for a MOSFET model.

Arguments

None

Value Returned

t	The MOSFET model is printed to the netlist file.
nil	The command cannot be run.

hnlHspicePrintNMOSfetElement

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

Description

This function prints out the line for a n-MOSFET model.

Arguments

None

Value Returned

t	The n-MOSFET model is printed to the netlist file.
nil	The command cannot be run.

Name Mapping Variables

Spectre

Described below are the name mapping variables for Spectre Direct netlister:

Variable	Description
<code>hnlSpectreMapInstInName</code>	List of characters that are invalid internal to an inst name.
<code>hnlSpectreMapNetInName</code>	List of characters that are invalid internal to a net name.
<code>hnlSpectreMapInstFirstChar</code>	List of characters that are invalid for the first character of a inst name.
<code>hnlSpectreMapNetFirstChar</code>	List of characters that are invalid for the first character of a net name.

Default Behavior

- Replace the invalid characters by their escaped equivalent.
- If the invalid character is a number, replace it by an underscore followed by the number.

Customization

- You can directly set any/all of the above mentioned variables to customize the character substitution and therefore the mapping.
- Each of these variables is checked separately to determine whether it is to be set or not.
- In case a variable is set, it is used to determine character substitution and hence mapping.

Flow

- The variables are used if they are set explicitly.
- Otherwise, the first time you netlist, all the variables are populated with their default values. The second time onwards the variable values will be used. If you need to modify

Virtuoso ADE SKILL Reference

Core Netlist Functions

just a part of the character substitution list, you can do it by modifying the variable in the CIW window. Otherwise the default values are used throughout.

HspiceD

Described below are the name mapping variables for HspiceD netlist:

Variable	Description
<code>hnlHspiceDMapInstInName</code>	List of characters that are invalid internal to an inst name.
<code>hnlHspiceDMapNetInName</code>	List of characters that are invalid internal to a net name.

Default Behavior

These variables do not have default values.

Customization

- You can directly set any/all of the above mentioned variables to customize the character substitution and therefore the mapping.
- Each of these variables is checked separately to determine whether it is to be set or not.
- In case a variable is set, it is used to determine character substitution and hence mapping.

Flow

- The variables are used if they are set explicitly.
- If these variables are not set explicitly, HspiceD netlist uses the following values:

- ☐ For mapping nets, it uses:

```
((">" ">") ("<" "<") "." ("#" "___") "$" "%" "^" "&" "*" "(" ")" "\\ " ("|" "___") "+" "-" "=" "{" "}"  
("[ " "___") ("]" "___") "\" " ("'" "___") (":" "___") ";" "~" ("`" "___") "," "?" "/" "@"
```

- ☐ For mapping instances, it uses:

```
((">" ">") ("<" "<") "." "!" ("#" "___") "$" "%" "^" "&" "*" "(" ")" "\\ " ("|" "___") "+" "-" "=" "{" "}"  
("[ " "___") ("]" "___") "\" " ("'" "___") (":" "___") ";" "~" " " , " "?" "/" "@"
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

Netlisting Option Functions for Socket Interfaces

This topic lists the SKILL functions that are available for direct and socket interfaces.

[asiDisplayNetlistOption](#)

[asiGetNetlistOption](#)

[asiInit<yourSimulator>NetlistOption](#)

[asiSetNetlistOption](#)

asiDisplayNetlistOption

```
asiDisplayNetlistOption(  
    o_tool  
)  
=> t / nil
```

Description

Displays the current set of netlist options and values. Use this function only to determine which netlist options you can modify.

Note: Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object.

Value Returned

<i>t</i>	Displays the set of netlist option names and values.
<i>nil</i>	Prints an error message.

Examples

Displays the Spectre netlist options.

```
asiDisplayNetlistOption( asiGetTool( 'Spectre' ) )
```

asiGetNetlistOption

```
asiGetNetlistOption(  
    { o_session | o_tool }  
    s_name  
)  
=> g_value / nil
```

Description

Gets the value of the specified netlist option.

Arguments

<i>o_session</i>	Simulation session object.
<i>o_tool</i>	Simulation tool object.
<i>s_name</i>	Netlist option for which you want the value.

Value Returned

<i>g_value</i>	The value of the netlist option.
<i>nil</i>	There is an error.

Examples

Gets the value of the `modelNamePrefix` netlist option.

```
asiGetNetlistOption( session 'modelNamePrefix )
```

Related Topics

[asiDisplayNetlistOption](#)

asiInit<yourSimulator>NetlistOption

```
asiInit<yourSimulator>NetlistOption(  
    o_tool  
)  
=> t
```

Description

Calls the procedures that modify your simulator's netlist options.

You must write `asiInit<yourSimulator>NetlistOption`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	Procedures for modifying netlist options are called. You must write <code>asiInit<yourSimulator>NetlistOption</code> to return <code>t</code> .

Examples

Creates the procedure that calls the procedures to modify the netlist options for the XYZ simulator.

```
procedure( asiInitXYZNetlistOption( tool )  
    <insert your code>  
    => t  
)
```

asiSetNetlistOption

```
asiSetNetlistOption(  
    { o_session | o_tool }  
    s_name  
    g_value  
)  
=> g_value / nil
```

Description

Sets a netlisting option.

Arguments

<i>o_session</i>	Simulation session object.
<i>o_tool</i>	Simulation tool object.
<i>s_name</i>	Name of the netlist option you want to set.
<i>g_value</i>	Value for the netlist option.

Value Returned

<i>g_value</i>	The value of the netlist option.
<i>nil</i>	Indicates an error.

Examples

Changes the maximum length of the net name to 16.

```
asiSetNetlistOption( session 'maxNameLength 16 )
```

Related Topics

[asiDisplayNetlistOption](#)

Mixed-Signal Simulation Functions

Mixed-signal simulation functions can be categorized into the following:

[Simulation Functions for Direct Interfaces](#)

[Simulation Functions for Direct and Socket Interfaces](#)

[Functions for Formatting Hierarchical Interface Elements](#)

Simulation Functions for Direct Interfaces

[asiVerilogNetlistMoreCB](#)

[asiGetDigitalNetlistFileName](#)

[asiConstructDigitalNetlist](#)

[asiInitializeNetlisterMixed](#)

[asiNetlistMixed](#)

[asiGetVerilogCommandLineOption](#)

[asiGetDigitalCommandLineOption](#)

[asiPrepareDigitalSimulation](#)

[asiCheckDigitalSimulationSuccess](#)

asiVerilogNetlistMoreCB

```
asiVerilogNetlistMoreCB(  
    )  
=> t
```

Description

Displays the Verilog HNL Netlisting Option form.

Arguments

None

Value Returned

t	Always returns t.

Examples

```
asiVerilogNetlistMoreCB()  
=> t
```

asiGetDigitalNetlistFileName

```
asiGetDigitalNetlistFileName(  
    o_session  
)  
=> t_digitalNetlistFileName
```

Description

Returns the digital netlist file name.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_digitalNetlistFileName</i>	The digital netlist file name.

asiConstructDigitalNetlist

```
asiConstructDigitalNetlist(  
    o_session  
)  
=> t / nil
```

Description

Constructs the digital netlist file for viewing.

This file is a concatenated form of the following files:

- The template stimulus file (`testfixture.template`)
- The verimix stimulus file (`testfixture.verimix`)
- The verimix specific defines (`simOptions.verimix`)
- The part of the netlist file that defines Interface Elements and Verimix Synchronization task statements (`IE.verimix`)
- Parasitic simulation annotate definitions (`annotate_msb`)
- The `$save_waveform` definitions (`saveDefs`)
- The hierarchical netlist blocks concatenated into one file (`netlist`).

Arguments

<code>o_session</code>	Simulation session object.

Value Returned

<code>t</code>	The netlist file is created.
<code>nil</code>	Indicates an error.

asiInitializeNetlistMixed

```
asiInitializeNetlistMixed(  
    o_session  
)  
=> t_mixedSignalDesignObject / nil
```

Description

Initializes the mixed-signal netlist. Partitions the design and then calls `nlCreateDesign` to get the design object. Does not re-partition if the design has not changed since last partition.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>t_mixedSignalDesignObject</i>	The object representing the mixed-signal design if the netlist is initialized.
<i>nil</i>	Indicates an error.

asiNetlistMixed

```
asiNetlistMixed(  
    o_session  
)  
=> g_status / nil
```

Description

This method performs mixed netlisting. It creates a formatter object with `nlCreateFormatter`, after which the netlister is run and a netlist is generated with `nlNetlist`.

This function is called by `asiNetlist`. You should not make a direct call to this function.

Arguments

<code>o_session</code>	Simulation session object.

Value Returned

<code>g_status</code>	A DPL of file names created by the netlister.
<code>nil</code>	Indicates an error.

asiGetVerilogCommandLineOption

```
asiGetVerilogCommandLineOption(  
    o_session  
)  
=> t_verilogCommandLineOption
```

Description

Returns the Verilog simulator command line options.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

t_verilogCommandLineOption

The command line options for the Verilog simulator with mixed mode capability.

asiGetDigitalCommandLineOption

```
asiGetDigitalCommandLineOption(  
    o_session  
)  
=> t_digitalCommandLineOption
```

Description

Returns the digital part of the mixed simulation command line options.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_digitalCommandLineOption</i>	The digital part of the mixed simulation command line options.

asiPrepareDigitalSimulation

```
asiPrepareDigitalSimulation(  
    o_session  
)  
=> t / nil
```

Description

Performs digital run directory clean up. It removes the file that stores the exit code for any previous digital simulation and cleans up the digital simulator old log file.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t</i>	The old log files are deleted.
<i>nil</i>	Indicates an error.

asiCheckDigitalSimulationSuccess

```
asiCheckDigitalSimulationSuccess(  
    o_session  
)  
=> t / nil
```

Description

Reports the success or failure of the simulation by looking at the digital simulator status file. Further, if no digital signals are saved by the user, this function updates the `logFileVerilog` file to indicate that no digital waveforms are available for viewing.

Arguments

<code>o_session</code>	Simulation session object.

Value Returned

<code>t</code>	The simulation is successful.
<code>nil</code>	Indicates an error.

Simulation Functions for Direct and Socket Interfaces

You can use the following simulation functions for direct and socket interfaces:

[asiGetNetworkId](#)

[asiGetDigitalStimulusFileName](#)

[asiEditDigitalStimulus](#)

[asiPartitionDesign](#)

[asiGetDigitalSimulatorLogFileName](#)

[asiGetDigitalSimExecName](#)

[asiSetVerilogHost](#)

[asiSetVerilogHostMode](#)

[asiGetVerilogHost](#)

[asiGetVerilogHostMode](#)

[asiGetAnalogRunDir](#)

[asiGetDigitalRunDir](#)

[asiGetAnalogKeepList](#)

[asiGetDigitalKeepList](#)

[asiInitMixedKeepOption](#)

[asiInitVerilog](#)

[asiInitVerilogEnvOption](#)

[asiInitVerilogFNLEnvOption](#)

[asiInitVerilogHNLEnvOption](#)

[asiInitVerilogSimOption](#)

[asiSetAnalogSimulator](#)

Virtuoso ADE SKILL Reference

Core Netlister Functions

asiSetDigitalSimulator

mspDisplaySetPartSetupForm

mspEditIEProps

asiGetNetworkId

```
asiGetNetworkId(  
    o_session  
)  
=> o_networkId
```

Description

Returns the network ID of a mixed-signal design.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>o_networkId</i>	The network ID of the mixed-signal design

asiGetDigitalStimulusFileName

```
asiGetDigitalStimulusFileName(  
    o_session  
)  
=> t_digitalStimulusFileName
```

Description

Returns the digital stimulus file name. This file name is the concatenated path string of the digital netlist run directory followed by `testfixture.verimix`.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_digitalStimulusFileName</i>	The full path of the digital stimulus file name.

asiEditDigitalStimulus

```
asiEditDigitalStimulus(  
    o_session  
)  
=> o_childId / nil
```

Description

This method edits the digital stimulus file. It gets the full path to the stimulus file by calling `asiGetDigitalStimulusFileName`. If the stimulus file does not exist, the `asiNetlist` flowchart step is executed to run the netlister and create a digital stimulus file. Finally the stimulus file is opened in an editor.

Arguments

<code>o_session</code>	The OASIS session object.

Value Returned

<code>o_childId</code>	Handle on the child editor process.
<code>nil</code>	Indicates an error.

Examples

```
asiInitializeNetlisterMixed( session )
```

asiPartitionDesign

```
asiPartitionDesign(  
    o_session  
)  
=> t / nil
```

Description

This method partitions a mixed-signal design into analog and digital parts.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t</i>	Returns <i>t</i> if the design is successfully partitioned.
<i>nil</i>	Indicates an error.

Examples

```
asiNetlistMixed( session )  
=> t
```

asiGetDigitalSimulatorLogFileName

```
asiGetDigitalSimulatorLogFileName(  
    o_session  
)  
=> t_verilogLogFileName
```

Description

Retrieves the digital simulator log file name for the given design.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_verilogLogFileName</i>	The name of the log file for digital simulator.

asiGetDigitalSimExecName

```
asiGetDigitalSimExecName(  
    o_session  
)  
=> t_digitalSimExecName
```

Description

Displays the verimix executable name.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_digitalSimExecName</i>	Name of the digital simulator executable.

asiSetVerilogHost

```
asiSetVerilogHost(  
    o_session  
    t_host  
)  
=> t_digitalSimulatorHostName
```

Description

Sets the digital simulator host name for mixed-signal simulation.

Arguments

<i>o_session</i>	Simulation session object.
<i>t_host</i>	Name of the host on which you want to run the digital simulator.

Value Returned

<i>t_digitalSimulatorHostName</i>	Name of the host on which you want to run the digital simulator.

asiSetVerilogHostMode

```
asiSetVerilogHostMode(  
    o_session  
    t_hostMode  
)  
=> t_digitalSimulatorHostMode
```

Description

Specifies whether the digital simulator will run locally or on a remote host. If you specify `remote`, you must specify the host name by using the `asiSetVerilogHost` command.

Arguments

<i>o_session</i>	Simulation session object.
<i>t_hostMode</i>	Either of the following modes: <code>local</code> or <code>remote</code> .

Value Returned

<i>t_digitalSimulatorHostMode</i>	The digital simulator host mode for mixed-signal simulation.

asiGetVerilogHost

```
asiGetVerilogHost(  
    o_session  
)  
=> t_digitalSimulatorHostName
```

Description

Returns the digital simulator host name for mixed-signal simulation. This method first verifies if host name was set using *asiSetVerilogHost*. If yes, it returns that host name. Otherwise, it returns the default value of digitalHost in the *.cdsenv* file for tool *asimenv* and partition startup.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_digitalSimulatorHostName</i>	Name of the host on which the digital simulator will be run.

asiGetVerilogHostMode

```
asiGetVerilogHostMode(  
    o_session  
)  
=> t_digitalSimulatorHostMode
```

Description

Returns the digital simulator host mode for mixed-signal simulation. This method first verifies if host mode was set using `asiSetVerilogHostMode`. If yes, it returns that mode. Otherwise, it returns the default value of `digitalHostMode` in the `.cdsenv` file for tool `asimenv` and partition startup.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_digitalSimulatorHostMode</i>	The digital simulator host mode for mixed-signal simulation.

asiGetAnalogRunDir

```
asiGetAnalogRunDir(  
    o_session  
)  
=> t_AnalogSimulatorRunDir
```

Description

Returns the analog simulator run directory.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_AnalogSimulatorRunDir</i>	Analog simulator run directory.

asiGetDigitalRunDir

```
asiGetDigitalRunDir(  
    o_session  
)  
=> t_DigitalSimulatorRunDir
```

Description

Returns the digital simulator run directory.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_AnalogSimulatorRunDir</i>	The digital simulator run directory.

asiGetAnalogKeepList

```
asiGetAnalogKeepList(  
    o_session  
)  
=> l_AnalogSignalDescriptionList
```

Description

Returns a list of user-selected output signals that are analog. This method first checks if the current simulation session is mixed-signal. If yes, then it returns all the user-selected output signals within the analog partition of the mixed-signal design. Otherwise, it calls `asiGetKeepList` to get the list of signals and currents to be saved during simulation.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>l_AnalogSignalDescriptionList</i>	A list of analog signals to be saved during simulation.
<i>nil</i>	No signals are specified to be kept during simulation.

asiGetDigitalKeepList

```
asiGetDigitalKeepList(  
    o_session  
)  
=> l_DigitalSignalDescriptionList
```

Description

Returns a list of user-selected output signals that are digital.

Arguments

<i>o_session</i>	Simulation session object.

Value Returned

<i>l_DigitalSignalDescriptionList</i>	A list of digital signals to be saved during simulation.
<i>nil</i>	No signals are specified to be kept during simulation.

asiInitMixedKeepOption

```
asiInitMixedKeepOption(  
    o_tool  
)  
=> t
```

Description

Initializes the mixed-signal keep options variables.

Arguments

<i>o_tool</i>	Name of the tool.

Value Returned

t	The mixed-signal keep options variables are initialized.

Examples

```
asiInitMixedKeepOption(asiGetTool('spectre'))  
=> t
```

asiInitVerilog

```
asiInitVerilog(  
    o_tool  
)  
=> t
```

Description

Initializes the Verilog tool.

Arguments

<i>o_tool</i>	The Verilog tool.

Value Returned

t	The Verilog tool is initialized.

Examples

```
asiInitVerilog(asiGetTool('spectre'))  
=> t
```

asiInitVerilogEnvOption

```
asiInitVerilogEnvOption(  
    o_tool  
)  
=> t
```

Description

Initializes the base Verilog environment options.

Arguments

<i>o_tool</i>	Name of the Verilog tool.

Value Returned

t	The base Verilog environment options are initialized.

Examples

```
asiInitVerilogEnvOption(asiGetTool('spectre'))  
=> t
```

asiInitVerilogFNLEnvOption

```
asiInitVerilogFNLEnvOption(  
    o_tool  
)  
=> t
```

Description

Initializes the Verilog FNL netlisting options.

Arguments

<i>o_tool</i>	Name of the tool.

Value Returned

t	The Verilog FNL netlisting options are initialized.

Examples

```
asiInitVerilogFNLEnvOption(asiGetTool('spectre'))  
=> t
```


asiInitVerilogHNLEnvOption

```
asiInitVerilogHNLEnvOption(  
    o_tool  
)  
=> t
```

Description

Initializes the Verilog HNL netlisting options.

Arguments

<i>o_tool</i>	Tool

Value Returned

t	The Verilog HNL netlisting option are initialized.

Examples

```
asiInitVerilogHNLEnvOption(asiGetTool('spectre'))  
=> t
```

asiInitVerilogSimOption

```
asiInitVerilogSimOption(  
    o_tool  
)  
=> t
```

Description

Initializes the simulation options for Verilog.

Arguments

<i>o_tool</i>	Name of the tool.

Value Returned

t	The simulation options for Verilog are initializes.

Examples

```
asiInitVerilogSimOption(asiGetTool('spectre))
```

asiSetAnalogSimulator

```
asiSetAnalogSimulator(  
    o_tool  
    s_analogSimulator  
)  
=> s_analogSimulator
```

Description

Sets the analog simulator.

Arguments

<i>o_tool</i>	Name of the tool.
<i>s_analogSimulator</i>	Analog simulator name.

Value Returned

s_analogSimulator Analog simulator name that has been set.

Examples

```
asiSetAnalogSimulator( asiGetTool('ams') 'spectre )  
=> anaSimulator1
```

asiSetDigitalSimulator

```
asiSetDigitalSimulator(  
    o_tool  
    s_digitalSimulator  
)  
=> s_digitalSimulator
```

Description

Sets the digital simulator.

Arguments

<i>o_tool</i>	Name of the tool.
<i>s_digitalSimulator</i>	Digital simulator name.

Value Returned

<i>s_digitalSimulator</i>	Digital simulator name that has been set.

Examples

```
asiSetDigitalSimulator( asiGetTool('Spectre) 'ams )
```

mSPDisplaySetPartSetupForm

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

Description

This function displays the Partitions Options form, which can be used to edit and set the values for the Analog Stop View Set and Digital Stop View Set.

Arguments

None

Value Returned

t	The Partitions Options form is displayed.
nil	The Partitions Options form cannot be displayed.

Examples

```
mSPDisplaySetPartSetupForm()  
=> t
```

mspEditIEProps

```
mspEditIEProps(  
    t_objectType  
)  
=> t / nil
```

Description

This is the main entry procedure for the Interface Element property editor. It takes an object type. The object type can be one of instTerm, instance, terminal, cellView, lib, or default.

This function first calls a selection function appropriate for the object type, which prompts the user to select an object. When the object has been selected, the selection function will execute its callback. An exception to this is that for lib and default objectType values, no selection functions are called. Next, the IE model property editor is called to create a parameter entry form for the current ieModel. The callbacks to this form will attach properties to the object selected earlier.

Arguments

<i>t_objectType</i>	The object type.

Value Returned

<i>t</i>	The IE properties are edited.

Note: This value is returned immediately after the function is called denoting that the function has completed execution. The function does not wait for the actions mentioned in the description to complete. It is only responsible for invoking the selection function which generates the callback action and subsequent updation of properties is needed.

<i>nil</i>	The command cannot be run.

Examples

```
mspEditIEProps("cellView")  
=> t
```

Functions for Formatting Hierarchical Interface Elements

Use the following functions to format pmos, rpmos, nmos, and rnmos gates:

hnlVerilogPrintNmosPmos

hnlVerilogPrintCmos

nlGetCdf

It is not necessary to write these functions unless the default definition suffices.

hnlVerilogPrintNmosPmos

```
hnlVerilogPrintNmosPmos(  
    t_name  
)  
=> t
```

Description

Used by netlister to print `pmos`, `rpms`, `nmos`, or `rnmos` gates.

It is not necessary to write these functions unless the default definition suffices.

Arguments

<i>t_name</i>	The name of the gate. Valid values are <code>pmos</code> <code>rpms</code> <code>nmos</code> and <code>rnmos</code> .

Value Returned

<i>t</i>	Name of the gates to be printed

hnlVerilogPrintCmos

```
hnlVerilogPrintCmos(  
    t_name  
)  
=> t
```

Description

Used by netlister to print `cmos`, `rcmos` gates.

Arguments

<i>t_name</i>	The name of the gate. Valid values are <code>cmos</code> and <code>rcmos</code> .

Value Returned

<i>t</i>	Name of the gates to be printed

nlGetCdf

```
nlGetCdf(  
    o_inst  
)  
=> o_cdfId
```

Description

Obtains the CDF information of the IE cell. CDF information contains the default IE parameter, default IE macro model file name and parameter mapping from CDF to simulator specific names.

Arguments

<i>o_inst</i>	The IE object (nlIEInstance)

Value Returned

<i>o_cdfId</i>	CDF data of the IE cell

Examples

```
SpectreimInfo = nlGetCdf(ie_inst)->simInfo->spectre
```

Virtuoso ADE SKILL Reference

Core Netlister Functions

Functions to Customize Simulator Integration

This topic lists the SKILL functions that let you customize simulator integration.

Environment Variable Functions

[asiAddEnvOption](#)

[asiChangeEnvOption](#)

[asiChangeEnvOptionFormProperties](#)

[asiDeleteEnvOption](#)

[asiDisplayEnvOption](#)

[asiDisplayEnvOptionFormProperties](#)

[asiGetEnvOptionChoices](#)

[asiGetEnvOptionVal](#)

OASIS Functions

[asiExtractQRCTechPathFromDSPF](#)

[asiExtractQRCTechPathFromSession](#)

[asiGetAnalogSimulator](#)

[asiGetAdvAnalysis](#)

[asiGetEMIROptionVal](#)

[asiGetNetlistFileListToSymLink](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiGetDigitalSimulator

asiAnalogAutoloadProc

ansAnalogRegCDFsimInfo

asiCheckAcEnabledWhenNoiseEnabled

asiCheckAnalysis

asiCheckBlank

asiCreateIncludeStatementFile

asiGetAnalysisField

asiGetHighPerformanceOptionVal

asiSetHighPerformanceOptionVal

asiDisplayHighPerformanceOption

asiGetDesignCellName

asiGetDesignLibName

asiGetDesignViewName

asiGetDrlData

asiGetId

asiGetIterationUpdateFile

asiGetResultsPsfDir

asiGetResultsNetlistDir

asiGetSimulatorList

asiGetSimCommandLineOrder

asiGetStimulusGlobals

asiGetStimulusInputs

asilsConfigDesign

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiSetValid

asiCheckBlankNumericLeq

asiCheckBlankNumericGeq

asiFormatGraphicalStimuli

asiFormatGraphicalStimulusFileList

asiAddOceanAlias

asiGetAvailableMCOptions

asiGetSupportedMCOptions

asiSetEMIROptionVal

OASIS Print Functions

artOutfile

artFprintf

artClose

artCloseAllFiles

artFlush

artListOpenFiles

Simulator Option Functions

asiAddSimOption

asiChangeSimOption

asiChangeSimOptionFormProperties

asiDeleteSimOption

asiDisplaySimOption

asiDisplaySimOptionFormProperties

asiGetReservedWordList

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

[asilsCaseSensitive](#)

[asiGetSimOptionChoices](#)

[asiGetSimOptionNameList](#)

[asiGetSimOptionSendMethod](#)

[asiGetSimOptionVal](#)

[asiGetSimulationRunCommand](#)

[asiInit<yourSimulator>SimOption](#)

[asilsCLPSupported](#)

[asiSetHostOptions](#)

[asiSetSimOptionChoices](#)

[asiSetSimOptionVal](#)

[asiGetSimulatorSrcList](#)

Analysis Functions

[asiAddAnalysis](#)

[asiAddAnalysisField](#)

[asiAddAnalysisOption](#)

[asiChangeAnalysis](#)

[asiChangeAnalysis](#)

[asiChangeAnalysisField](#)

[asiChangeAnalysisOption](#)

[asiChangeAnalysisOptionFormProperties](#)

[asiCreateAnalysisField](#)

[asiCreateAnalysisOption](#)

[asiDeleteAnalysis](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

[asiDeleteAnalysisField](#)
[asiDeleteAnalysisOption](#)
[asiDisableAnalysis](#)
[asiDisplayAnalysis](#)
[asiDisplayAnalysisField](#)
[asiDisplayAnalysisOption](#)
[asiDisplayAnalysisOptionFormProperties](#)
[asiEnableAnalysis](#)
[asiFormatAnalysis](#)
[asiGetAnalysis](#)
[asiGetAnalysisFieldChoices](#)
[asiGetAnalysisFieldList](#)
[asiGetAnalysisFieldVal](#)
[asiGetAnalysisFormFieldChoices](#)
[asiGetAnalysisFormObj](#)
[asiGetAnalysisFormFieldVal](#)
[asiGetAnalysisName](#)
[asiGetAnalysisNameList](#)
[asiGetAnalysisOptionChoices](#)
[asiGetAnalysisOptionList](#)
[asiGetAnalysisOptionSendMethod](#)
[asiGetAnalysisOptionVal](#)
[asiGetAnalysisParamNameList](#)
[asiGetEnabledAnalysisList](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiInit<yourSimulator>Analysis

asiAnalysisEnabled

asiSetAnalysisFieldChoices

asiSetAnalysisFieldVal

asiSetAnalysisFormFieldChoices

asiSetAnalysisFormFieldVal

asiSetAnalysisFormWidth

asiSetAnalysisOptionFormProperties

asiSetAnalysisOptionChoices

asiSetAnalysisOptionVal

Simulation Control Functions for Direct Interfaces

asiInitialize

asiNetlist

asiInterruptSim

asiSetProjectDirChangeSetup

asiQuitSimulator

Integrator Overloadable Methods for Invoking Simulation

asiRunSimulation

asiGetPredefinedCommandLineOption

asiGetCommandFooter

asiFormatControlStmts

asiFormatDesignVarList

asiFormatInitCond

asiFormatNodeSet

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiFormatKeepList

asiFormatSimulatorOptions

asiFormatAnalysisList

asiFormatAnalysis

asiFormatModelLibSelectionList

asiFormatDefinitionFileList

asiFormatTextStimulusFileList

asiNeedSuffixEvaluation

asiInvalidateControlStmts

asiGetSimExecName

asiGetCommandLineOption

asiGetAnalysisSigList

asiGetAnalysisType

asiGetAnalysisSimFieldList

asiGetModelLibSelectionList

asiGetModelLibFile

asiGetModelLibSection

asiGetDefinitionFileList

asiGetTextStimulusFileList

asiGetFormattedVal

asiGetSelObjName

asiGetSelObjType

asiGetSelObjValue

asiMapOutputName

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

[asiGetSimInputFileList](#)

[artInvalidateAmap](#)

Flowchart Functions

[asiAddFlowchartLink](#)

[asiAddFlowchartStep](#)

[asiChangeFlowchartStep](#)

[asiCreateFlowchart](#)

[asiDeleteFlowchartLink](#)

[asiDeleteFlowchartStep](#)

[asiDisplayFlowchart](#)

[asiExecuteFlowchart](#)

[asiFinalNetlist](#)

[asiGetFlowchart](#)

[asiInit<yourSimulator>Flowchart](#)

[asiInvalidateFlowchartStep](#)

[asiInvalidateFlowchartStep](#)

[asiRawNetlist](#)

[asiSendAnalysis](#)

[asiSendControlStmts](#)

[asiSendDesignVars](#)

[asiSendInitCond](#)

[asiSendInitFile](#)

[asiSendKeepList](#)

[asiSendModelPath](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

[asiSendNetlist](#)

[asiSendNodeSets](#)

[asiSendOptions](#)

[asiSendRestore](#)

[asiSendUpdateFile](#)

Keep Option Functions

[asiAddKeepOption](#)

[asiChangeKeepOption](#)

[asiChangeKeepOptionFormProperties](#)

[asiDeleteKeepOption](#)

[asiDisplayKeepOption](#)

[asiDisplayKeepOptionFormProperties](#)

[asiGetKeepOptionChoices](#)

[asiGetKeepOptionVal](#)

[asiInit<yourSimulator>KeepOption](#)

[asiSetKeepOptionChoices](#)

[asiSetKeepOptionVal](#)

Reliability Functions

[calcValForRel](#)

[maeGetStressFile](#)

[maeAddRelxSetup](#)

[maeCopyRelxSetup](#)

[maeCreateNetlistForRel](#)

[maeDeleteRelxSetup](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

maeExportScenarioSetup

maeGetReliabilityAnalysisEnabled

maeGetReliabilityModel

maeGetReliabilityOptionVal

maeGetReliabilityVarValue

maeGetRelxSetups

maeGetRelxSetupEnabled

maeGetRelxStressFile

maeGetRelxTest

maeGetRelxTestEnabled

maeImportScenarioSetup

maeRenameRelxSetup

maeSetReliabilityAnalysisEnabled

maeSetReliabilityModel

maeSetReliabilityOptionVal

maeSetReliabilityVarValue

maeSetRelxSetupEnabled

maeSetRelxStressFile

maeSetRelxTest

maeSetRelxTestEnabled

relxEnableFormTab

relxDisplayDiscField

relxEnableDiscField

relxHideAgeCalculationApproachField

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

relxGetCustomTabName

relxCustomizeDisplayOrEnableStatus

relxCreateCustomizedTab

relxAddSetupRelxOption

asiFormatSpecialParameterForRel

relxInitOptionsInCdsenv

relxAddReliabilityInStateComponent

relxAddSetupRelxOption

asiFormatSpecialParameterForRel

relxInitOptionsInCdsenv

relxAddReliabilityInStateComponent

relxAddReliabilityOption

relxCreateRunObjectFile

relxDisplayResult

relxFormatRXControlFile

relxGetModifyNetlistVal

relxGetMosAgingTimeUnitVal

relxGetMosAgingTimeVal

relxGetReliabilityOptionChoices

relxGetReliabilityOptionVal

relxGetRelxStage

relxGetStressFileDir

relxGetRXControlFileName

relxGetSimulationRunCommand

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

relxGetSpecifiedReliabilityStateFileName

relxGetUserCmdLine

relxHighLightDevices

relxInitAdapterReliabilityOption

relxGetUserCmdLine

relxHighLightDevices

relxInitAdapterReliabilityOption

relxIsAgingOn

relxIsReliabilityEnabled

relxIsStressOn

relxPostSimulation

relxRunSimulation

relxSetAgingVal

relxSetReliabilityOptionFormProperties

relxSetReliabilityOptionVal

relxSetReliabilityVal

Stimulus Generation Functions

asiStmSupportWaveformGeneration

asiStmGenerateNetlist

asiStmRunSimulation

asiStmGenerateWaveform

Miscellaneous Functions

ahdlUpdateViewInfo

amseGeneralSetupForm

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

amseQuickSetupForm

amsUpdateTextviews

vmsUpdateCellViews

annRetrieveFromEffectiveCDF

artEnableAnnotationBalloon

artGenerateHierSymbolCDF

artGetCdfTargetCV

artGetCellViewDesignVarList

artCurrentInstSimName

artListToWaveform

artBlankString

artMakeString

artMakeStringPrec15

asiAddDesignVarList

asiAddVerilogArgs

asiGetDataReaderVersion

asiLoadState

asiSaveState

asiCheck

asiCheckDesignVariable

asiCheckExpression

asiCheckExpressionGreater

asiCheckBlankNumeric

asiCheckBlankNumericGreater

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

[asiCheckBlankNumericNequal](#)

[asiCheckBlankNetExists](#)

[asiCheckMultipleGreater](#)

[asiCheckSimulationSuccess](#)

[asiCreateLogFileVerilog](#)

[asiDcStore](#)

[asiGetCurrentSession](#)

[asiGetDesignVarList](#)

[asiGetFormFieldChoices](#)

[asiGetFormFieldVal](#)

[asiGetKeepList](#)

[asiGetLogFileList](#)

[asiGetMarchList](#)

[asiGetNetlistDir](#)

[asiGetMarchList](#)

[asiGetNetlistDir](#)

[asiGetOutputList](#)

[asiGetPlotList](#)

[asiGetPsfDir](#)

[asiGetSession](#)

[asiGetSimName](#)

[asiGetTool](#)

[asiGetTopCellView](#)

[asiSendSim](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiSetDesignVarList

asiSetFormFieldChoices

asiSetFormFieldVal

asiSetKeepList

asiSetMarchList

asiSetPlotList

asiSetSyncFlag

asiTransientStore

asiMapNetName

asiMapTerminalName

asiMapInstanceName

asiRegCallBackOnSimComp

asiUnRegCallBackOnSimComp

asiRegCallBackOnSimCompForDist

asiUnRegCallBackOnSimCompForDist

almDefineParam_accuracyMode

almDefineParam_additionalParam

almDefineParam_fq

almDefineParam_noiseParaLabel

almDefineParam_nportFileB

almDefineParam_noiseParaLabel

almDefineParam_tranAdvanParaLabel

almDefineParam_tranParaLabel

almGetModuleName

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

almGetNamePrefix

almGetParameterList

almGetTerminalList

almGetTerminalMap

almSetTerminalMap

almGetOpPointParamMap

almSetOpPointParamMap

almGetNetlistProcedure

almGetViewInfoNameList

almGetNetlistType

almHasViewInformation

almSetNamePrefix

almSetModuleName

almSetNetlistProcedure

almSetParameterList

almSetTerminalList

almSetPropMappingList

almGetPropMappingList

almSetOtherParameterList

almGetStringParameterList

almSetStringParameterList

ancGetSimInstName

ancAdjustNameCase

drbBrowseFormCB

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

msgHelp

addCheck

deleteChecks

densityEstimateWaveform

disableAllChecks

disableChecks

disableDeviceChecking

displayChecks

enableAllChecks

enableChecks

enableDeviceChecking

setDevCheckOptions

printViolations

captabSummary

evmOFDM

relxOption

asiAddModelLibSelection

asiRemoveAllModelLibSelection

Environment Variable Functions

This topic lists the SKILL functions that let you work with environment variables.

[asiAddEnvOption](#)

[asiChangeEnvOption](#)

[asiChangeEnvOptionFormProperties](#)

[asiDeleteEnvOption](#)

[asiDisplayEnvOption](#)

[asiDisplayEnvOptionFormProperties](#)

[asiGetEnvOptionChoices](#)

[asiGetEnvOptionVal](#)

[asiInit<yourSimulator>EnvOption](#)

[asiSetEnvOptionChoices](#)

[asiSetEnvOptionVal](#)

asiAddEnvOption

```
asiAddEnvOption(  
    o_tool  
    [ ?name s_name ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?browse g_browse ]  
    [ ?mode t_browseMode ]  
    [ ?invalidateFunc s_invalidateFunc ]  
    [ ?defaultSubcircuitCall s_defaultSubcircuitCall ]  
)  
=> o_envVar / nil
```

Description

Adds a new simulation environment option.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<code>o_tool</code>	Simulation tool object
<code>?name s_name</code>	Name of the environment option you want to add.
<code>?prompt t_prompt</code>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <code>s_name</code>
<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code> , <code>integer</code> , <code>float</code> , <code>toggle</code> , <code>cyclic</code> , <code>radio</code> , <code>Boolean</code> , <code>list</code> , <code>radioToggle</code> , <code>listBox</code> , <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> See the <code>asiAddSimOption</code> function for an example that shows how to use the <code>separator</code> argument.
<code>?choices l_choices</code>	List of choices if <code>s_type</code> is <code>cyclic</code> , <code>radio</code> , <code>radioToggle</code> or <code>listBox</code> , or the list of switches if <code>s_type</code> is <code>toggle</code> . Note: This argument is valid only if <code>s_type</code> is <code>cyclic</code> , <code>toggle</code> , <code>radio</code> , <code>radioToggle</code> , or <code>listBox</code> .
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code> , <code>cyclic</code> , <code>toggle</code> , and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <code>l_choices</code>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code> , <code>float</code> , or <code>scale</code> option. Default Value: <code>nil</code> , which means <code>-infinity</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?max <i>g_max</i></code>	Specifies the maximum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr <i>s_allowExpr</i></code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>
<code>?row <i>x_row</i></code>	Row in the form where the field appears. Note: This argument is valid only for <i>'twoD'</i> type forms.
<code>?column <i>x_column</i></code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. Note: This argument is valid only for <i>'twoD'</i> type forms.
<code>?width <i>x_width</i></code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. Note: This argument is valid only for <i>'twoD'</i> type forms. Default Value: 1
<code>?coordinates <i>l_coordinates</i></code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) Note: This argument is valid only for <i>'custom'</i> type forms.
<code>?displayOrder <i>x_displayOrder</i></code>	

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first. The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form.

Note: This argument is valid only for *'oneD* type forms.

Valid Values: Any integer

?labelText *t_labelText*

Optional label displayed with frame type fields.

Default Value: *nil*

?private *s_private* Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software.
Valid Values: *t* (option does not appear in the UI), *nil* (option appears in the UI)

Default Value: *nil*

?display *s_display* Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: *t*

?editable *s_editable*

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: *t*

This argument only applies to type-in fields.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?appCB s_appCB` Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: (*o_session*)

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. You can use this function to do error checking on the value entered by the designer.

`?formApplyCB s_formApplyCB`

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: (*o_session r_form r_field*)

`?changeCB st_changeCB`

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows` Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?browse g_browse`

Adds a browse button (...) for the specified option. This button can be used to open a file selection form to browse and select a file. The mode of file selection is specified by the `t_mode` argument.

Valid values: `t`, `nil`

Default value: `nil`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?mode t_mode</code>	Specifies mode for the file selection form that is displayed when the <code>g_browse</code> argument is set to <code>t</code>. Valid values: <code>anyFile</code> specifies that you can open any file type; <code>existingFile</code> specifies that you can open any existing file; and <code>existingFiles</code> specifies that you can select multiple files. Default value: <code>anyFile</code>
<code>?invalidateFunc s_invalidateFunc</code>	Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart. Callback parameter list: <code>(o_session)</code>
<code>?defaultSubcircuitCall s_defaultSubcircuitCall</code>	Specifies the name of the netlist procedure added to the <code>cdf</code> <code>simInfo</code> for sub-circuits that is added by the cell-view to cell-view utility. For interfaces derived from the <code>asiAnalog</code> class this is an empty string, and no netlist procedure is added. For interfaces derived from the <code>asiSocket</code> class this is <code>"ansSpiceSubcktCall"</code>. To change the value, use <code>asiChangeEnvOption</code> in your <code>envOption.il</code> file.

Value Returned

<code>o_envVar</code>	Returns the environment option object.
<code>nil</code>	Indicates an error.

Examples

Adds a new environment option called XYZ file in the fourth position in the Environment Options form.

```
asiAddEnvOption( tool ?name 'XYZfile ?prompt "XYZ file" ?value  
"~/XYZfile/XYZfile" ?displayOrder 4 )
```

Related Topics

[asiAddSimOption](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

[asiDisplayEnvOption](#)

[asiChangeEnvOptionFormProperties](#)

asiChangeEnvOption

```
asiChangeEnvOption(  
    o_tool  
    [ ?name s_name ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
)  
=> o_envVar / nil
```

Description

Changes a simulation environment option.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<code>o_tool</code>	Simulation tool object.
<code>?name s_name</code>	Name of the environment option you want to change.
<code>?prompt t_prompt</code>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <code>s_name</code>
<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code> , <code>integer</code> , <code>float</code> , <code>toggle</code> , <code>cyclic</code> , <code>radio</code> , <code>Boolean</code> , <code>list</code> , <code>radioToggle</code> , <code>listBox</code> , <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> See the <code>asiAddSimOption</code> function for an example that shows how to use the <code>separator</code> argument.
<code>?choices l_choices</code>	List of choices if <code>s_type</code> is <code>cyclic</code> , <code>radio</code> , <code>radioToggle</code> or <code>listBox</code> , or the list of switches if <code>s_type</code> is <code>toggle</code> . Note: This argument is valid only if <code>s_type</code> is <code>cyclic</code> , <code>toggle</code> , <code>radio</code> , <code>radioToggle</code> , or <code>listBox</code> .
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code> , <code>cyclic</code> , <code>toggle</code> , and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <code>l_choices</code>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code> , <code>float</code> , or <code>scale</code> option. Default Value: <code>nil</code> , which means <code>-infinity</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?max <i>g_max</i></code>	Specifies the maximum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr <i>s_allowExpr</i></code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>
<code>?row <i>x_row</i></code>	Row in the form where the field appears. Note: This argument is valid only for <i>'twoD</i> type forms.
<code>?column <i>x_column</i></code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. Note: This argument is valid only for <i>'twoD</i> type forms.
<code>?width <i>x_width</i></code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. Note: This argument is valid only for <i>'twoD</i> type forms. Default Value: 1
<code>?coordinates <i>l_coordinates</i></code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) Note: This argument is valid only for <i>'custom</i> type forms.
<code>?displayOrder <i>x_displayOrder</i></code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first.

Note: The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for *'oneD'* type forms.

Valid Values: Any integer

?labelText *t_labelText*

Optional label displayed with frame type fields.

Default Value: *nil*

?private *s_private*

Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software.

Valid Values: *t* (option does not appear in the UI), *nil* (option appears in the UI)

Default Value: *nil*

?display *s_display*

Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: *t*

?editable *s_editable*

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

Note: This argument only applies to type-in fields.

`?appCB s_appCB`

Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: `(o_session)`

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

`?formApplyCB s_formApplyCB`

Specifies a callback function that is executed when the designer clicks *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

`?changeCB st_changeCB`

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows`

Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values:

- `t`: Specifies that multiple items can be selected.
- `nil`: Specifies that only one item can be selected at a time.

`?invalidateFunc` *s_invalidateFunc*

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: (*o_session*)

Value Returned

o_envVar

Returns the changed environment option object.

`nil`

Returns an error message and `nil` if unsuccessful.

Examples

Changes the default value of the switch view list.

```
asiChangeEnvOption( tool ?name 'switchViewList
?value '("XYZ" "spice" "cmos.sch" "schematic" "symbol"))
```

Related Topics

[asiDisplayEnvOption](#)

[asiChangeEnvOptionFormProperties](#)

asiChangeEnvOptionFormProperties

```
asiChangeEnvOptionFormProperties(  
    o_tool  
    [ ?type s_type ]  
    [ ?width x_width ]  
    [ ?columns x_columns ]  
)  
=> o_formObj / nil
```

Description

Changes the display characteristics of the Environment Options form.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<code>o_tool</code>	Simulation tool object
<code>?type s_type</code>	Specifies the form type. Valid Values: ■ <code>'oneD</code>: Specifies a sequential display of fields in one column. ■ <code>'twoD</code>: Specifies a two dimensional display of fields based on row and column positions. (Use the <code>asiAddEnvOption</code> or <code>asiChangeEnvOption</code> function to specify values for the rows and columns.) ■ <code>'custom</code>: Lets you specify the exact coordinate locations for each field.)Use the <code>asiAddEnvOption</code> or <code>asiChangeEnvOption</code> function to specify the coordinates.) ■ <code>'matrix</code>: Specifies a matrix of equally sized fields. Default Value: <code>'oneD</code>
<code>?width x_width</code>	Width of the form, in pixels. Default Value: The minimum default width of the form is 400 pixels. If the fields require more space than this, the form defaults to the smallest width that can accommodate the fields.
<code>?columns x_columns</code>	Number of columns. Use this argument only with matrix type forms. Default Value: 2

Value Returned

<code>o_formObj</code>	The environment option form object
<code>nil</code>	Indicates an error.

Examples

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Changes the width of the Environment Options form for the Spectre simulator.

```
asiChangeEnvOptionFormProperties( asiGetTool('Spectre')?width 450)
```

Related Topics

[Setting up Simulator Environment Options](#)

asiDeleteEnvOption

```
asiDeleteEnvOption(  
    o_tool  
    s_name  
)  
=> t / nil
```

Description

Deletes a simulation environment option.

Arguments

<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the environment option you want to remove from the form.

Value Returned

<i>t</i>	The option is deleted.
<i>nil</i>	Returns an error message and <i>nil</i> if the option does not exist.

Examples

Removes the *stimulusFile* option from the Environment Options form.

```
asiDeleteEnvOption( tool 'stimulusFile )
```

Related Topics

[asiDisplayEnvOption](#)

asiDisplayEnvOption

```
asiDisplayEnvOption(  
    o_tool  
)  
=> t / nil
```

Description

Displays the current set of simulation environment option names and values. Use this function only to determine which environment options you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object

Value Returned

<i>t</i>	Displays the list of environment option names and values.
<i>nil</i>	Indicates an error.

Examples

Displays the environment options for Spectre.

```
asiDisplayEnvOption( asiGetTool( 'Spectre' ) )
```

asiDisplayEnvOptionFormProperties

```
asiDisplayEnvOptionFormProperties(  
    o_tool  
)  
=> t / nil
```

Description

Displays the form characteristics for the Environment Options form. Use this function only to determine which form characteristics you want to modify.

Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object

Value Returned

<i>t</i>	Displays a list of Environment Options form characteristics and returns <i>t</i> .
<i>nil</i>	Indicates an error.

Examples

Displays the form characteristics for the Spectre Environment Options form and returns *t*.

```
asiDisplayEnvOptionFormProperties( asiGetTool('Spectre))
```


asiGetEnvOptionChoices

```
asiGetEnvOptionChoices(  
    { o_session | o_tool }  
    s_name  
)  
=> l_choices / nil
```

Description

Gets the list of choices for an environment option that is set up as a list box.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the option for which you want the list of choices.

Value Returned

<i>l_choices</i>	A list of choices for the specified simulation environment option.
<i>nil</i>	The option does not exist.

Related Topics

[asiDisplayEnvOption](#)

asiGetEnvOptionVal

```
asiGetEnvOptionVal(  
    { o_session | o_tool }  
    s_name  
)  
=> g_value / nil
```

Description

Gets the value for the specified simulation environment option.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the option for which you want the value.

Value Returned

<i>g_value</i>	Returns the value for the specified simulation environment option.
<i>nil</i>	Returns <i>nil</i> if the option does not exist.

Examples

Gets the value for the *modelPath* simulation environment option.

```
asiGetEnvOptionVal( session 'modelPath )
```

Related Topics

[asiDisplayEnvOption](#)

asiInit<yourSimulator>EnvOption

```
asiInit<yourSimulator>EnvOption(  
    o_tool  
)  
=> t
```

Description

Calls your procedures to modify the simulation options.

You must write `asiInit<yourSimulator>EnvOption`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object

Value Returned

<code>t</code>	Returns <code>t</code> when the procedures are called.

Examples

Creates the procedure that calls the procedures to add environment options for the XYZ simulator.

```
procedure( asiInitXYZEnvOption( tool )  
    <insert your code>  
    t  
)
```

asiSetEnvOptionChoices

```
asiSetEnvOptionChoices(  
    { o_session | o_tool }  
    s_name  
    l_choices  
)  
=> l_choices / nil
```

Description

Specifies the list of choices to appear in the list box field for the specified environment option.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name a simulation environment option of the type <code>listBox</code> .
<i>l_choices</i>	List of choices to appear in the list box field.

Value Returned

<i>l_choices</i>	Returns the new list of choices to appear in the list box field.
<i>nil</i>	Returns <code>nil</code> if the option does not exist.

asiSetEnvOptionVal

```
asiSetEnvOptionVal(  
    { o_session | o_tool }  
    s_name  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value of the specified simulation environment option.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the simulation environment option to which you want to assign a value.
<i>g_value</i>	Value for the simulation option.

Value Returned

<i>g_value</i>	Returns the new value for the simulation environment option.
<i>nil</i>	Returns an error message and <i>nil</i> if the option does not exist.

Examples

Sets the value of *modelFiles* to `/mypath/mySpectreModels.scs` for Spectre Direct.

```
asiSetEnvOptionVal(session "modelFiles" '("/mypath/mySpectreModels.scs"))
```

Sets the value of *modelFiles* to `/mypath/mySpectreModels.scs` and the section name to `cmos` for Spectre Direct.

```
asiSetEnvOptionVal(session "modelFiles" '("/mypath/mySpectreModels.scs"  
"cmos"))
```

Sets the value of the *modelPath* simulation environment option to `~models/nmos` for Spectre.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
asiSetEnvOptionVal( session 'modelPath "~models/nmos" )
```

Related Topics

[asiDisplayEnvOption](#)

OASIS Functions

This topic lists the SKILL functions that let you print and manage OASIS files.

[asiExtractQRCTechPathFromDSPF](#)

[asiExtractQRCTechPathFromSession](#)

[asiGetAnalogSimulator](#)

[asiGetAdvAnalysis](#)

[asiGetEMIROptionVal](#)

[asiGetNetlistFileListToSymLink](#)

[asiGetDigitalSimulator](#)

[asiAnalogAutoloadProc](#)

[ansAnalogRegCDFsimInfo](#)

[asiCheckAcEnabledWhenNoiseEnabled](#)

[asiCheckAnalysis](#)

[asiCheckBlank](#)

[asiCreateIncludeStatementFile](#)

[asiGetAnalysisField](#)

[asiGetHighPerformanceOptionVal](#)

[asiSetHighPerformanceOptionVal](#)

[asiDisplayHighPerformanceOption](#)

[asiGetDesignCellName](#)

[asiGetDesignLibName](#)

[asiGetDesignViewName](#)

[asiGetDrIData](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiGetId

asiGetIterationUpdateFile

asiGetResultsPsfDir

asiGetResultsNetlistDir

asiGetSimulatorList

asiGetSimCommandLineOrder

asiGetStimulusGlobals

asiGetStimulusInputs

asilsConfigDesign

asiSetValid

asiCheckBlankNumericLeq

asiCheckBlankNumericGeq

asiFormatGraphicalStimuli

asiFormatGraphicalStimulusFileList

asiAddOceanAlias

asiGetAvailableMCOptions

asiGetSupportedMCOptions

asiSetEMIROptionVal

asiExtractQRCTechPathFromDSPF

```
asiExtractQRCTechPathFromDSPF(  
    t_dspfFile  
)  
=> t_QRCTechDir / nil
```

Description

Extracts the path to the QRC technology directory from the specified DSPF file.

Arguments

<i>t_dspfFile</i>	Path to the DSPF file.

Value Returned

<i>t_QRCTechDir</i>	Path to the QRC technology directory.
nil	If no QRC technology directory is found for the specified DSPF file.

Examples

The following examples display how this SKILL function returns the path to the QRC technology directory extracted from the specified DSPF file.

```
asiExtractQRCTechPathFromDSPF("../..../adc_sample_hold.dspf")  
=> "/ADE1/backup/ADE2/users/jinduo/DAVSource/devLnx86_64bit_IC618_MAIN/  
tools.lnx86/dfII/test/maestro/newEMIRUI/supportN16/test/"  
asiExtractQRCTechPathFromDSPF("../..../adc_sample_holdNone.dspf")  
=> nil
```

asiExtractQRCTechPathFromSession

```
asiExtractQRCTechPathFromSession(  
    t_session  
)  
=> t_QRCTechDir / nil
```

Description

Extracts the path to the QRC technology directory from the DSPF file defined in the EMIR setup for the specified session.

Arguments

<i>t_session</i>	Name of the OASIS session.

Value Returned

<i>t_QRCTechDir</i>	Path to the QRC technology directory.
<i>nil</i>	■ No QRC technology directory is found for the DSPF file specified in the EMIR setup for the specified session. ■ No DSPF file is defined in the EMIR setup for the specified session.

Examples

The following example displays how this SKILL function returns the path to the QRC technology directory extracted from the DSPF file defined in the EMIR setup for the specified session.

```
asiExtractQRCTechPathFromSession(asiGetCurrentSession())  
=> "/ADE1/backup/ADE2/users/jinduo/DAVSource/devLnx86_64bit_IC618_MAIN/  
tools.lnx86/dfII/test/maestro/newEMIRUI/supportN16/test/"
```

asiGetAnalogSimulator

```
asiGetAnalogSimulator(  
    { o_session | o_tool }  
)  
=> s_simulatorName
```

Description

Gets the value of the analog simulator for a tool or session object.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object

Value Returned

<i>s_simulatorName</i>	Name of the analog simulator.

Examples

Displays the analog simulator for the tool Spectre as Spectre.

```
asiGetAnalogSimulator(asiGetTool('Spectre'))
```

asiGetAdvAnalysis

```
asiGetAdvAnalysis(  
    { o_session | o_tool }  
    s_analysisName  
)  
=> o_analysis / nil
```

Description

Returns the object of advanced analyses, such as Monte Carlo. This function is similar to the `asiGetAnalysis` function and is normally used by third party integrator to implement the simulator interface for their simulators.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_analysisName</i>	Name of the analysis you want to retrieve.

Value Returned

<i>s_analysisName</i>	The analysis object for the specified analysis.
<i>nil</i>	Returns <i>nil</i> if the specified analysis does not exist or there is an error.

Examples

The following example returns the object of the analysis `mc`.

```
asiGetAdvAnalysis(session 'mc)
```

asiGetDesignVarListWithOriginalExpressions

```
asiGetDesignVarListWithOriginalExpressions(  
    o_session  
)  
=> l_designVars / nil
```

Description

Returns a list of name-value pairs for design variables specified in the given session.

The value represents the unevaluated value of the design variable.

Arguments

<i>o_session</i>	Simulation session object

Value Returned

<i>l_designVars</i>	Returns the list of name-value pairs for design variables specified in the given session.
<i>nil</i>	Returns <i>nil</i> if the specified session object is invalid.

Examples

Returns the list of name-value pairs for design variables specified in the session `stdobj@0x30dd91b8`.

```
asiGetDesignVarListWithOriginalExpressions(stdobj@0x30dd91b8)  
=> (("a" "4") ("var2" "VAR(\"var1\") +2") ("var1" "2"))
```

asiGetEMIROptionVal

```
asiGetEMIROptionVal(  
    o_session  
    s_name  
)  
=> g_value / nil
```

Description

Returns the value of the specified EMIR option from the EMIR Analysis Setup form.

Arguments

<i>o_session</i>	Simulation session object
<i>s_name</i>	Name of the EMIR option of which you want the value.

Value Returned

<i>g_value</i>	Returns the value of the specified EMIR option.
<i>nil</i>	Returns <i>nil</i> if the specified option does not exist.

Examples

The following example shows how this function returns the name of the QRC technology file specified in the *qrcTechFile* field of the EMIR Analysis Setup form.

```
session = asiGetCurrentSession()  
=> stdobj@0x30dd91b8  
asiGetEMIROptionVal(stdobj@0x30dd91b8 'qrcTechFile)  
=> "$WORKDIR/em.ict"
```

asiGetNetlistFileListToSymLink

```
asiGetNetlistFileListToSymLink(  
    o_session  
)  
=> l_files / nil
```

Description

Returns the list of files present in the netlist directory and created by the `asiNetlist` procedure.

Arguments

<code>o_session</code>	Simulation session object

Value Returned

<code>l_files</code>	Returns the list of files.
<code>nil</code>	Indicates an error.

Examples

```
(defmethod asiGetNetlistFileListToSymLink ( (session asiAnalog_session) )  
    list("netlistHeader" "netlistFooter" ".__master.termorder"  
    ".__master.optionFile" ".control" ".includedModels" "netlist" "si.env"  
    "si.foregnd.log" "designInfo" "ihnl" "amap" "map")
```

asiGetDigitalSimulator

```
asiGetDigitalSimulator(  
    { o_session | o_tool }  
)  
=> s_simulatorName / nil
```

Description

Gets the value of the digital simulator for a tool or session object.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object

Value Returned

<i>s_simulatorName</i>	Name of the digital simulator.
<i>nil</i>	Returns <i>nil</i> if there is no digital simulator.

Examples

Displays the digital simulator for the tool Spectre as *nil* as there is no digital simulator for Spectre.

```
asiGetDigitalSimulator(asiGetTool('Spectre'))
```


asiAnalogAutoloadProc

```
asiAnalogAutoloadProc(  
    )  
=> t
```

Description

Called by OASIS for the purpose of autoloading the context. This is done so that the classes are defined before the tool is created and initialization is started.

Arguments

None

Value Returned

t	The context is autoloaded successfully.

ansAnalogRegCDFsimInfo

```
ansAnalogRegCDFsimInfo(  
    )  
=> t
```

Description

This is a utility function used to create a CDF for the <yourSimulator> simulator. The `ansAnalogRegCDFsimInfo` functions are called by the CDF editor. These functions are used to provide data type information for all the `simInfo` attributes.

Arguments

None

Value Returned

t Function is defined successfully.

Examples

The following example illustrates the declaration of `ansAnalogRegCDFsimInfo`:

```
procedure( ansAnalogRegCDFsimInfo()  
    '( (nil name netlistProcedure type symbol)  
      (nil name otherParameters type list)  
      (nil name instParameters type list)  
      (nil name componentName type symbol)  
      (nil name namePrefix type string)  
      (nil name termOrder type list)  
      (nil name termMapping type list)  
      (nil name propMapping type list)  
    )  
)
```

The following is a sample <yourSimulator>.ini file for <yourSimulator> simulator:

```
asiRegisterTool( '<yourSimulator> ?class 'asiAnalog ?initFunc  
'asiInitialize)  
procedure( ansRegCDFsimInfo_<yourSimulator>()  
    ansAnalogRegCDFsimInfo()  
)
```

Note: To assure consistency in netlisting across simulators, the parameters returned by `ansAnalogRegCDFsimInfo` must not be redefined. If the formatter of your simulator has the `useInstNamePrefix netlist` option set to `nil`, the `namePrefix` parameter must be removed.

asiCheckAcEnabledWhenNoiseEnabled

```
asiCheckAcEnabledWhenNoiseEnabled(  
    o_session  
    r_form  
)  
=> t / nil
```

Description

This method verifies that an AC analysis is enabled when a noise analysis is selected. If this is not the case, it displays an error message in the error dialog box of the analysis form and sets the error status of the form. It is called during the form apply callback.

Arguments

<i>o_session</i>	Simulation session object
<i>r_form</i>	Form created by a call to <code>hiCreateAppForm</code> or <code>hiCreateForm</code> .

Value Returned

<i>t</i>	Returns <i>t</i> when check is successful.
<i>nil</i>	Returns an error message and <i>nil</i> if unsuccessful.

asiCheckAnalysis

```
asiCheckAnalysis(  
    o_analog  
    r_form  
)  
=> t / nil
```

Description

Checking function for the analysis class. e.g. for Spectre simulator object it checks each field value in the environment, highlights any errors, returns `t` or `nil`. Can be used for different analog simulator analysis objects.

The only difference between this and the check method in the analog class is that this one does not give an error if 'step' is not set.

Arguments

<i>o_analog</i>	Derived object of the analog simulator class.
<i>r_form</i>	Form created by a call to <code>hiCreateAppForm</code> or <code>hiCreateForm</code> .

Value Returned

<code>t</code>	Returns <code>t</code> if check is successful.
<code>nil</code>	Indicates an error.

asiCheckBlank

```
asiCheckBlank(  
    o_obj  
    r_form  
    s_fieldName  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is non-blank.

Arguments

<i>o_obj</i>	Can be one of the following objects: o_analysis, o_anaOption, o_envOption, o_simOption, o_keepOption.
<i>r_form</i>	Form created by a call to hiCreateAppForm or hiCreateForm.
<i>s_fieldName</i>	Handle to this field within a form. It may be referenced as formHandle->hiFieldSym

Value Returned

t	Returns t if the check is successful.
nil	Otherwise, returns nil.

asiCreateIncludeStatementFile

```
asiCreateIncludeStatementFile(  
    o_simulatorSession  
    t_netlistFile  
    t_includeNetlistFile  
)  
=> t_includeNetlistFile / nil
```

Description

Creates a file that contains a statement to include netlist for simulation. Override this function to create a customized include statement according to the the simulator netlisting procedure and syntax.

The default include statement is as per Spectre format, as shown below.

```
include "netlist"
```

Arguments

<i>o_simulatorSession</i>	An object of the simulator session class
<i>t_netlistFile</i>	Name of the netlist file
<i>t_includeNetlistFile</i>	Name of the temporary file that contains the statement to include netlist

Value Returned

<i>t_includeNetlistFile</i>	Name of the temporary file that contains the statement to include netlist
<i>nil</i>	Otherwise

Examples

```
session = asiGetCurrentSession()  
=> stdobj@0x3157a0b0  
; Session for other simulators can be fetched accordingly.  
  
asiCreateIncludeStatementFile with (stdobj@0x3157a0b0 "netlist" "tmpNetlistFile")  
=> "tmpNetlistFile"
```

asiGetAnalysisField

```
asiGetAnalysisField(  
    o_analysis  
    s_fieldName  
)  
=> o_fieldEnvVar / nil
```

Description

Returns the specified analysis field object.

Arguments

<i>o_analysis</i>	The analysis object for which you want the field object.
<i>s_fieldName</i>	Handle to this field within a form. It may be referenced as <code>formHandle->hiFieldSym</code> .

Value Returned

<i>o_fieldEnvVar</i>	Object of field environment variable.
<i>nil</i>	Returns <code>nil</code> if no analysis field is present.

Examples

Gets the tran analysis, stop field object.

```
analysis = asiGetAnalysis( session 'tran )  
asiGetAnalysisField( analysis 'stop )
```

asiGetHighPerformanceOptionVal

```
asiGetHighPerformanceOptionVal(  
    o_session  
    s_varName  
)  
=> t_varNameVal / nil
```

Description

Returns the field value of the High Performance form with the passed session object.

This method returns the field value of the High Performance form for spectre class (`spectre_session`) or AMS class (`ams_session`) only. For other classes, this method returns `nil`.

Arguments

<code>o_session</code>	Simulation session object
<code>s_varName</code>	The field name of the High Performance form.

Value Returned

<code>t_varNameVal</code>	The value of the specified field.
<code>nil</code>	Returns <code>nil</code> when function call is unsuccessful.

Examples

Returns the value of the variable `uniMode` for spectre class or AMS class.

```
session = asiGetCurrentSession()  
asiGetHighPerformanceOptionVal( session 'uniMode )
```


asiSetHighPerformanceOptionVal

```
asiSetHighPerformanceOptionVal(  
    s_sessionName  
    s_varName  
    s_varValue  
)  
=> t_Value / nil
```

Description

Sets the value for the specified High Performance option variable.

Arguments

<i>S_sessionName</i>	The session for the simulator.
<i>s_varName</i>	Name of the variable in the High Performance Option form that you want to set.
<i>s_varValue</i>	The value of the variable field.

Value Returned

<i>t_Value</i>	Returns the value of the specified field.
<i>nil</i>	Returns <i>nil</i> if unsuccessful.

Examples

Sets the value of the variable `uniMode` to `APS` for spectre class or AMS class.

```
Session = asiGetCurrentSession()  
asiSetHighPerformanceOptionVal(Session 'uniMode "APS")  
=> "APS"
```

asiDisplayHighPerformanceOption

```
asiDisplayHighPerformanceOption(  
    o_toolName  
)  
=> t / nil
```

Description

Displays a list of variable and values of the High Performance Simulation Option form.

Arguments

<i>o_toolName</i>	The tool object according to the simulator.

Value Returned

<i>t</i>	Displays the list of options and values of the High Performance Simulation Option form and returns <i>t</i> .
<i>nil</i>	Returns <i>nil</i> if unsuccessful.

Examples

```
asiDisplayHighPerformanceOption(asiGetTool('spectre'))  
uniMode:      "Spectre"  
turboSwitch:  nil  
envSwitch:    nil  
uniSeparate:  
errorLevel:   "Do not override"  
mtOption:     "Auto"  
numThreads:   ""  
apspplus:     nil  
cktpreset:    "None"  
proc_affinity: ""  
pseparate:  
srSwitch:     nil  
psrOption:    "Default"  
psrFmax:      ""  
preserveOption: "None"  
preserveInst:  nil  
preserveSelect:  
preserveClear:  
rfseparate:  
rfmtOption:   "Disable"  
numRFThreads: ""  
  
asiDisplayHighPerformanceOption(asiGetTool('ams'))  
=> ERROR (ADE-5032): asiEnvDisplayVar: No variables defined in partition
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
'turboOpts' of tool 'ams'.  
=> nil
```

asiGetDesignCellName

```
asiGetDesignCellName(  
    o_session  
)  
=> t_cellName / nil
```

Description

Returns the cell name of the design associated with the passed session object.

Arguments

<i>o_session</i>	Simulation session object

Value Returned

<i>t_cellName</i>	Name of the cell associated with the session object.
<i>nil</i>	The function failed to give cell name.

Examples

```
session = asiGetCurrentSession()  
asiGetDesignCellName(session)
```

asiGetDesignLibName

```
asiGetDesignLibName(  
    o_session  
)  
=> t_libName / nil
```

Description

Returns the library name of the design associated with the passed object.

Arguments

<i>o_session</i>	Simulation session object

Value Returned

<i>t_libName</i>	Name of the library containing the cellview.
<i>nil</i>	Returns <code>nil</code> when function call fails.

Examples

```
session = asiGetCurrentSession()  
asiGetDesignLibName(session)
```

asiGetDesignViewName

```
asiGetDesignViewName(  
    o_session  
)  
=> t_viewName / nil
```

Description

Returns the view name of the design associated with the passed object.

Arguments

<i>o_session</i>	Simulation session object

Value Returned

<i>t_viewName</i>	Name of the view related with the passed session object
<i>nil</i>	Returns <i>nil</i> when function call fails.

Examples

```
session = asiGetCurrentSession()  
asiGetDesignViewName(session)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiGetDrIData

```
asiGetDrIData(  
    t_anaType  
    l_specifier  
    t_dataDir  
)  
=> g_familyOrWaveform / nil
```

Description

Returns the results data for the given specifier from the given data directory.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<i>t_anaType</i>	Type of analysis. This can be any type of basic analysis. For example, <code>tran</code> , <code>ac</code> , <code>dc</code> , <code>noise</code> , and so on.
<i>l_specifier</i>	The specifier name list. Depending on the value of the <i>t_anaType</i> argument or the data you need to return, this argument can contain the following: ■ For basic analyses, a list containing a single node name or element name for which you want to return the results data. For example, <code>list("in_m")</code>, where <code>in_m</code> is a net name. ■ For <code>info</code> analysis, a list containing the node name and the parameter name for which you want to return the results data. For example, <code>list("R0" "v")</code>, where <code>R0</code> is an instance name here and <code>v</code> is an instance parameter. ■ ? – This is a wildcard to be used to get the names of all the nodes available in the results data. ■ ?? – This is a wildcard to be used to get the names of all the nodes available in the results data along with their values.
<i>t_dataDir</i>	Data directory name

Value Returned

<i>g_familyOrWaveform</i>	Family data or waveform data
<i>nil</i>	Indicates an error.

Note: This function is required for 3rd party simulator integrations. For related information, you can also refer to the `asiDefineDataAccess` function.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Examples

In the following code example, the function returns a list of all the signal names available for the specified analysis type, `tran`.

```
asiGetDr1Data(list('tran) list("?") dir)
=>("in_m" "in_p" "net10" "net35" "out"
   "v0:p" "v1:p" "v2:p" "vdd!" "vss!"
   "I1.gnode" "I1.net6" "I1.net26" "I1.net52" "I1.Q3"
   "I1.Q2" "I1.Q4" "I1.Q0" "I1.M2" "I1.M5"
   "I1.M3" "I1.M1" "I1.C0" "I1.C1" "I0"
   "R0" "R2" "R1" "I1.R0" "E0"
   "v0" "v1" "v2" "I1.Q3.kpnp_mod" "I1.Q2.kpnp_mod"
   "I1.Q4.kpnp_mod" "I1.Q0.kpnp_mod" "I1.M2.nch_mod" "I1.M5.nch_mod"
   "I1.M3.pch_mod"
   "I1.M1.pch_mod" "capacitor" "resistor"
)
```

In the following code example, the function returns the operating point value of the instance parameter, `v`, for instance `R0`.

```
asiGetDr1Data("info" list("R0" "v") dir)
=>0.004584486
```

In the following code example, the function returns the waveform read from the specified analysis type, `tran`, for the given specifier, `in_m`.

```
asiGetDr1Data(list('tran) list("in_m") dir)
=>srrWave:0x150e4020
```

asiGetId

```
asiGetId(  
    o_session  
)  
=> x_id
```

Description

Returns the name of the session ID associated with the given OASIS session.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>x_id</i>	OASIS session ID

Examples

Gets the ID of the current OASIS session.

```
asiGetId( asiGetCurrentSession() )  
=> 1
```

asiGetIterationUpdateFile

```
asiGetIterationUpdateFile(  
    o_session  
)  
=> t_fileName / nil
```

Description

Returns the name and path to the simulator-specific file in which the iteration number and the corresponding PSF data location is written after each Monte Carlo iteration completes. This file is used to monitor the progress of the Monte Carlo run and to get the directory path from where the PSF data corresponding to an iteration is read.

Note the following:

- The file should be in the following format:

1	../psf/mc1_separate/mc1-001/
2	../psf/mc1_separate/mc1-002/
3	../psf/mc1_separate/mc1-003/
4	../psf/mc1_separate/mc1-004/
- The file should be updated with the iteration number and the corresponding PSF data location only after the completion of each iteration.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

o_session Specifies the session object.

Value Returned

t_fileName Returns the name and path to the simulator-specific file in which the iteration number and the corresponding PSF data location is written after each Monte Carlo iteration completes.

nil Returns *nil* if no such file is associated with the current session.

Examples

```
defmethod( asiGetIterationUpdateFile (( session XYZ_session )
    "mcddata.iter"
)
```

asiGetResultsPsfDir

```
asiGetResultsPsfDir(  
    o_session  
)  
=> t_PsfDir / nil
```

Description

Returns the name of the PSF directory for the current or last-run simulation.

Arguments

<i>o_session</i>	The OASIS session.

Value Returned

<i>t_PsfDir</i>	Returns the path of the PSF directory for the last simulation results.
<i>nil</i>	Returns <i>nil</i> if the directory is not present.

Examples

The following example returns the PSF directory for the current simulation session.

```
asiGetResultsNetlistDir(asiGetCurrentSession())  
=> "/servers/scratch02/aakhil/testcases/newtest/simulation/ampTest/spectre/  
schematic/distributed/job026/psf"
```

asiGetResultsNetlistDir

```
asiGetResultsNetlistDir(  
    o_session  
)  
=> t_netlistDir / nil
```

Description

Returns the name of the netlist directory for the current or last-run simulation.

Arguments

<i>o_session</i>	The OASISsession.

Value Returned

<i>t_netlistDir</i>	Returns the path of the netlist directory for the last simulation results.
<i>nil</i>	Returns <i>nil</i> if the directory is not present.

Examples

The following example returns the netlist directory for the current simulation session.

```
asiGetResultsNetlistDir(asiGetCurrentSession())  
=> "/servers/scratch02/aakhil/testcases/newtest/simulation/ampTest/spectre/  
schematic/distributed/job026/netlist"
```

asiGetSimulatorList

```
asiGetSimulatorList(  
    [ s_subclass ]  
)  
=> l_simulatorNameList / nil
```

Description

The function returns a list of all simulation interfaces within the specified simulator subclass.

Arguments

<i>s_subclass</i>	Name of the sub-class containing the OASIS simulation interfaces.

Value Returned

<i>l_simulatorNameList</i>	List containing simulation interfaces associated with the simulator subclass.
<i>nil</i>	There are no entries.

Examples

```
asiGetSimulatorList()  
("ams" "hspiceD" "spectre" "UltraSim")
```

For a specific subclass like `asiAnalog` this function returns the corresponding `simulatorNameList`.

```
asiGetSimulatorList('asiAnalog)  
("ams" "hspiceD" "spectre" "UltraSim")
```

asiGetSimCommandLineOrder

```
asiGetSimCommandLineOrder(  
    o_session  
)  
=> s_optionList / nil
```

Description

Returns the order of options used in the simulator run command. By default, it returns the options in the following order:

```
simulatorName inputFile simOptions scriptOptions
```

To get a different order of the options in the list returned, overload the function to specify the desired order. For example:

```
defmethod( asiGetSimCommandLineOrder( o_session )  
'(simulatorName simOptions scriptOptions inputFile)  
)
```

Argument

<i>o_session</i>	Simulation session object

Value Returned

<i>s_optionList</i>	List of the options in the default or specified order.
<i>nil</i>	Indicates an error.

Examples

Displays the options used to run the spectre simulator.

```
asiGetSimCommandLineOrder( spectre_session )
```


asiGetStimulusGlobals

```
asiGetStimulusGlobals(  
    o_session  
)  
=> l_globals / nil
```

Description

This method retrieves the list of global stimuli from the session.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>l_globals</i>	List of global stimuli.
<i>nil</i>	Returns <i>nil</i> if there is no global stimuli associated with the current session.

asiGetStimulusInputs

```
asiGetStimulusInputs(  
    o_session  
)  
=> l_inputs / nil
```

Description

This method retrieves the list of input stimuli from the session.

Arguments

<i>o_session</i>	Specifies the session object

Value Returned

<i>l_inputs</i>	List of input stimuli.
<i>nil</i>	Returns <i>nil</i> if there is no input stimuli associated with the current session.

asIsConfigDesign

```
asIsConfigDesign(  
    o_session  
)  
=> t / nil
```

Description

The function returns `t` if the design associated with the session is a Cadence 5.x configuration.

Arguments

<code>o_session</code>	Simulation session object

Value Returned

<code>t</code>	When the design belongs to 5.x config.
<code>nil</code>	Indicates an error.

Examples

```
asIsConfigDesign( session )
```

asiSetValid

```
asiSetValid(  
    o_analysis  
    g_value  
)  
=> t / nil
```

Description

The function sets valid analysis in the current simulation environment. The call to `asiSetValid` should be used with `asiCheck/asiCheckAnalysis` method for the analysis.

Arguments

<i>o_analysis</i>	The analysis object used
<i>g_value</i>	The value you want to choose either <code>t</code> or <code>nil</code> .

Value Returned

<code>t</code>	Returns <code>t</code> when the function sets value as true.
<code>nil</code>	Indicates an error.

asiCheckBlankNumericLeq

```
asiCheckBlankNumericLeq(  
    o_obj  
    r_form  
    s_fieldName  
    g_value  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a numeric value less than or equal to *g_value*.

Arguments

<i>o_obj</i>	Can be one of the following objects: <i>o_analysis</i> , <i>o_anaOption</i> , <i>o_envOption</i> , <i>o_simOption</i> , <i>o_keepOption</i> .
<i>r_form</i>	Form created by a call to <i>hiCreateAppForm</i> or <i>hiCreateForm</i> .
<i>s_fieldName</i>	Handle to this field within a form. It may be referenced as <i>formHandle->hiFieldSym</i>
<i>g_value</i>	The value you want to choose.

Value Returned

<i>t</i>	When the check is successful.
<i>nil</i>	Indicates an error.

asiCheckBlankNumericGeq

```
asiCheckBlankNumericGeq(  
    o_obj  
    r_form  
    s_fieldName  
    g_value  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a numeric value greater than or equal to *g_value*.

Arguments

<i>o_obj</i>	Can be one of the following objects: <i>o_analysis</i> , <i>o_anaOption</i> , <i>o_envOption</i> , <i>o_simOption</i> , <i>o_keepOption</i> .
<i>r_form</i>	Form created by a call to <i>hiCreateAppForm</i> or <i>hiCreateForm</i> .
<i>s_fieldName</i>	Handle to this field within a form. It may be referenced as <i>formHandle->hiFieldSym</i>
<i>g_value</i>	The value you want to choose.

Value Returned

<i>t</i>	When the check is successful.
<i>nil</i>	Indicates an error.

asiFormatGraphicalStimuli

```
asiFormatGraphicalStimuli(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats the graphical stimuli statements to send to Cadence SPICE.

For integrators, the function `asiFormatGraphicalStimuli` needs to be overloaded for the simulator session to create a file which contains the graphical stimuli. If you define a graphical stimuli, it is mandatory for this to be overloaded, else an error stating that the function `asiFormatGraphicalStimuli` is not defined for the appropriate class, is displayed:

```
... *Error* asiFormatGraphicalStimuli: no applicable method for the class -  
<yoursimulator>_session
```

Also, the function `asiPrintSource` needs to be written by the integrator.

Arguments

<i>o_session</i>	Simulation session object. For example, an XYZ session is <i>XYZ_session</i> for a third party.
<i>p_fp</i>	Specifies a file pointer to the file containing the graphical stimuli statements to send to the simulator (for simulators that are not in the Cadence SPICE socket).

Value Returned

<i>t</i>	Returns <i>t</i> if the stimuli statements are generated.
<i>nil</i>	Indicates an error.

Examples

The function prototype is as follows:

```
asiFormatGraphicalStimuli ((session XYZ_session) fp)
```

where, *fp* is the pointer to the file where graphical stimuli will be written.

asiFormatGraphicalStimulusFileList

```
asiFormatGraphicalStimulusFileList (
    o_session
    &_fp
)
=> t_string
```

Description

Formats the statement that includes the graphical stimulus files.

Arguments

<i>o_session</i>	Simulation session object.
<i>&_fp</i>	Pointer to the control statement file.

Value Returned

<i>t_string</i>	Formatted string that includes the graphical stimulus file.

Examples

```
defmethod ( asiFormatGraphicalStimulusFileList ( (session xyz_session) fp )
    (artFprintf fp "include \"%s\"\n" "./_graphical_stimuli.scs")
)
```


asiAddOceanAlias

```
asiAddOceanAlias(  
    s_simulatorName  
    s_alias  
    s_analysisName  
)  
=> t / nil
```

Description

Adds an ocean alias to the current simulator. This is useful for defining ocean related data access aliases for third party simulator integration.

Arguments

<i>s_simulatorName</i>	Simulator or tool's class name
<i>s_alias</i>	Data access alias for specified analysis name
<i>s_analysisName</i>	Equivalent analysis name recognized by third party simulator

Value Returned

t	When the alias is added and defined.
nil	Indicates an error.

Examples

For a third-party simulator integration if the simulator PSF maps transient analysis name as analysisTran1-tran then following OCEAN aliasing should be used:

```
asiAddOceanAlias( 'spice3 'tran "analysisTran1-tran")
```

This would make `selectResults('tran)` work in the auto generated OCEAN script file

asiGetAvailableMCOptions

```
asiGetAvailableMCOptions(  
    )  
    => l_list / nil
```

Description

Returns a list of Monte Carlo analysis options along with their description. The supported Monte Carlo analysis options are displayed in the `<OptionName, Description>` format.

Arguments

None

Value Returned

<code>l_list</code>	Returns a list of Monte Carlo analysis options supported by the application.
<code>nil</code>	Returns <code>nil</code> if there are no Monte Carlo analysis options associated with the current session.

Examples

```
asiGetAvailableMCOptions()  
    <mcmethode, "Level of statistical variation to apply. Possible values are  
    process, mismatch or all">  
    <mcnumpoints, "Number of Monte Carlo iterations to perform">  
    <samplingmode, "Method of statistical sampling to apply. Possible values are  
    standard or lhs">  
    <mcnumbins, "Number of bins for lhs (latin-hypercube) method. The number is  
    checked against numruns + firstrun - 1, and Max(numbins, numruns + firstrun -  
    1 ) is used for the lhs.">  
    <saveprocess, "Whether or not to save scalar data for statistically varying  
    process parameters which are subject to process variation. Possible values are  
    no or yes">  
    <savemismatch, "Whether or not to save scalar data for statistically varying  
    mismatch parameters which are subject to mismatch variation. Possible values  
    are no or yes">  
    <donominal, "This parameter controls whether or not simulator runs a nominal  
    run before starting the main Monte Carlo loop of iterations">  
    <saveallplots, " Whether or not to save data for family plots">  
    <montecarloseed, "Optional starting seed for random number generator">
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
<mcstartingrunnumber, "Starting iteration number">  
<dutsummary, "Subcircuit instances to which mismatch variations must be  
applied. Mismatch variations will also be applied to all subcircuits  
instantiated under the selected instances.">  
<ignoreflag, "If set, mismatch variations will not be applied to subcircuit  
instances selected in dut. Mismatch variations will also not be applied to all  
subcircuits instantiated under the selected instances.">
```

asiGetSupportedMCOptions

```
asiGetSupportedMCOptions(  
    o_session  
)  
=> l_list / nil
```

Description

Returns a list of Monte Carlo analysis options supported by your simulator. Ensure that the option names returned by this method match with the option names returned by `asiGetAvailableMCOptions`.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>l_list</code>	Returns a list of Monte Carlo options supported by your simulator.
<code>nil</code>	Returns <code>nil</code> if your simulator does not support Monte Carlo analysis options.

Examples

```
defmethod( asiGetSupportedMCOptions ( session XYZ_session )  
    list("mcstartingrunnumber" "mcnumpoints" "saveallplots" "mcmethod"  
"samplingmode" "saveprocess" "savemismatch" "donominal" "montecarloseed"  
"dutsummary" "ignoreflag")  
)
```

Related Topics

[asiGetAvailableMCOptions](#)

asiSetEMIROptionVal

```
asiSetEMIROptionVal(  
    o_session  
    s_name  
    g_value  
)  
=> t / nil
```

Description

Sets the given value for the specified EMIR option.

Arguments

<i>o_session</i>	Simulation session object
<i>s_name</i>	Name of the EMIR option to which you want to assign a value.
<i>g_value</i>	The value to be set for the specified EMIR option.

Value Returned

<i>t</i>	Successful operation
<i>nil</i>	Returns <i>nil</i> , along with an error message if the option does not exist.

Examples

Sets the specified QRC technology file in the *qrcTechFile* field of the EMIR Analysis Setup form.

```
session = asiGetCurrentSession()  
=> stdobj@0x30dd91b8  
asiSetEMIROptionVal(stdobj@0x30dd91b8 'qrcTechFile "techfile.txt")  
=> t
```

Modifies the customized options table in the *Summary* tab of the EMIR Analysis Setup form.

```
asiSetEMIROptionVal(asiGetCurrentSession() 'customizedOptionsListBox list(  
list(t "net" "name=[*] analysis=[vmax vavg iavg irms iacpeak]")  
list(t "net" "name=[*] analysis=[sigvavg] reftype=[avg] findsrc=[true]")  
list(t "net" "name=[*] analysis=[sigvmax] reftype=[max] findsrc=[true]")
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
list(t "net" "name=[*] analysis=[ipwc]")
list(t "emirutil" "autorun=[true] report=[text]")
list(t "emirutil" "dynamicACPeak=[multiplePeak]")
list(t "emirutil" "enableSHHeatSink=[true]")
list(t "emirutil" "res_pdata_sw=on")
list(t "emirutil" "extendHeatSinkRange=[0.4]")
list(t "emirutil" "connectingODLayers=[\"MD_OD MD_OD_N MD_OD_P MD_STI MD_OD_N_SRM
MD_OD_P_SRM MD_STI_SRM VDR MO\"]")
list(t "emirutil" "technology=\"tsmcN3Generic\"")
list(t "emirutil" "extendedreport=[true]")
list(t "emirutil" "idirn=[true]")
list(t "net" "design_res_models=[name=rhim l=lr w=wr unit=1] analysis=[iacpeak
iavgabs irms imax iavg ipwc vmax vavg]")
list(t "emirutil" "deltaTLayers=[(layer=rhim hi_em=1 deltaT=25)]")
list(t "emirutil" "auto_detect_geounit=[false]")
list(t "emirutil" "disableODSH=[false]")
list(t "emirutil" "disableHRSH=[false]")
list(t "emirutil" "extendedRMSReport=[true]")
list(t "emirutil" "shXMLFile=../netlist/input.tmideg0.xml")
list(t "emirutil" "sebMode=1")
list(t "emirutil" "sebParamFile=/projects/analog/bin/template/SHE/
N3E_2P02M_SEB.ircx")
list(t "emirutil" "print_em_report=[net=all_nets type=fit filename=fit.rpt]")
list(t "emirutil" "Tambient=105")
list(t "emirutil" "deltaT=5")
list(t "emirutil" "extendMetalRange=0.0")
list(t "emirutil" "extendHRRRange=0.5")
list(t "emirutil" "extendODRange=1.8")
list(t "<Click_To_Add>" "")
))
=> t
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The *Summary Information* table gets updated as follows.

	Options	Value
☒	net	name=[*] analysis=[vmax vavg iavg irms iacpeak]
☒	net	name=[*] analysis=[sigvavg] reftype=[avg] findsrc=[true]
☒	net	name=[*] analysis=[sigvmax] reftype=[max] findsrc=[true]
☒	net	name=[*] analysis=[ipwc]
☒	emirutil	autorun=[true] report=[text]
☒	emirutil	dynamicACPeak=[multiplePeak]
☒	emirutil	enableSHHeatSink=[true]
☒	emirutil	res_pdata_sw=on
☒	emirutil	extendHeatSinkDamage=[0.4]

OASIS Print Functions

This topic lists the available OASIS print functions.

[artOutfile](#)

[artFprintf](#)

[artClose](#)

[artCloseAllFiles](#)

[artFlush](#)

[artListOpenFiles](#)

artOutfile

```
artOutfile(  
    t_name  
    t_mode  
    x_len  
    t_break  
    t_cont  
    t_begCom  
    t_endCom  
    t_tab  
    t_comments  
)  
=> x_handle
```

Description

Opens the named file. The first argument is mandatory. The other arguments keep their default values if they are not set.

Arguments

<i>t_name</i>	Name of the file.
<i>t_mode</i>	Mode in which to open the file, either <i>w</i> or <i>a</i> . The default value is <i>w</i> .
<i>x_len</i>	Maximum number of chars before each new line. The default value is <i>internal default</i> .
<i>t_break</i>	Break characters, where breaking the line is legal. For example, <i>"\t"</i> . The default value is <i>null</i> .
<i>t_cont</i>	Continuation convention. For example, <i>"\\n"</i> or <i>"\n+"</i> . The default value is <i>null</i> .
<i>t_begCom</i>	Comment convention. For example, <i> ";"</i> or <i>"slash-asterisk"</i> . The default value is <i>null</i> .
<i>t_endCom</i>	Complement of <i>beginComment</i> . For example, <i>"\n"</i> or <i>asterisk-slash</i> .
<i>t_tab</i>	Indicates if a tab should be replaced with a blank. The default value is <i>1</i> .
<i>t_comments</i>	Indicates if comments can be ignored. The default value is <i>1</i> .

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

x_handle The named file

Examples

```
fHandle = artOutfile( "aFile" "a" )
```

artFprintf

```
artFprintf(  
    x_handle  
    t_text  
    g_args  
    )  
=> t / nil
```

Description

Prints out data like the standard C library `fprintf`, with the handle returned from `artOutfile()` as the first argument.

Arguments

<i>x_handle</i>	File handle returned from <code>artOutfile</code> .
<i>t_text</i>	Formatting string
<i>g_args</i>	Data to be printed.

Value Returned

<i>t</i>	Returns <i>t</i> if the data is printed.
<i>nil</i>	Indicates an error.

Examples

```
artFprintf( fHandle "%s" "test" )
```

artClose

```
artClose(  
    x_handle  
)  
=> t / nil
```

Description

Closes the file associated with the given handle.

Arguments

<i>x_handle</i>	File handle returned from <code>artOutfile</code>

Value Returned

<code>t</code>	The file is closed.
<code>nil</code>	Indicates an error.

Examples

```
artClose( fHandle )
```

artCloseAllFiles

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

Description

Closes all files opened with `artOutfile()`.

Arguments

None

Value Returned

<code>t</code>	The files are closed.
<code>nil</code>	Indicates an error.

Examples

```
artCloseAllFiles()  
=> t
```

artFlush

```
artFlush(  
    x_handle  
)  
=> t / nil
```

Description

Flushes the file associated with the given handle.

Argument

<i>x_handle</i>	File handle returned from <code>artOutfile</code> .

Value Returned

<i>t</i>	Returns <i>t</i> if the file is flushed.
<i>nil</i>	Indicates an error.

Examples

```
artFlush( fHandle )
```

artListOpenFiles

```
artListOpenFiles(  
    )  
=> l_names / nil
```

Description

Lists names of all files opened with `artOutfile`.

Arguments

None

Value Returned

<code>t</code>	Returns <code>t</code> if the file names are listed.
<code>nil</code>	Indicates an error.

Examples

```
fNameList = artListOpenFiles()
```

Simulator Option Functions

This topic lists all SKILL functions that let you modify simulator-specific options.

[asiAddSimOption](#)

[asiChangeSimOption](#)

[asiChangeSimOptionFormProperties](#)

[asiDeleteSimOption](#)

[asiDisplaySimOption](#)

[asiDisplaySimOptionFormProperties](#)

[asiGetReservedWordList](#)

[asilsCaseSensitive](#)

[asiGetSimOptionChoices](#)

[asiGetSimOptionNameList](#)

[asiGetSimOptionSendMethod](#)

[asiGetSimOptionVal](#)

[asiGetSimulationRunCommand](#)

[asiInit<yourSimulator>SimOption](#)

[asiSetHostOptions](#)

[asiSetSimOptionChoices](#)

[asiSetSimOptionVal](#)

[asiGetSimulatorSrcList](#)

asiAddSimOption

```
asiAddSimOption(  
    o_tool  
    [ ?name s_name ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?browse g_browse ]  
    [ ?mode t_browseMode ]  
    [ ?invalidateFunc s_invalidateFunc ]  
    [ ?genericName s_genericName ]  
    [ ?sendMethod s_sendMethod ]  
)  
=> o_envVar / nil
```

Description

Adds a new simulator option.

Arguments

<i>o_tool</i>	Simulation tool object
?name <i>s_name</i>	Name of the simulator option you want to define.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?prompt t_prompt</code>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <code>s_name</code>
<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> Note: The example that follows shows how to use the separator argument.
<code>?choices l_choices</code>	List of choices if <code>s_type</code> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <code>s_type</code> is <code>toggle</code>. This argument is valid only if <code>s_type</code> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <code>l_choices</code>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies whether *g_value* can contain expressions.

Valid Values: *t* (value can contain expressions), *nil* (value cannot contain expressions)

Default Value: *nil*

?row *x_row*

Row in the form where the field appears. This argument is valid only for *'twoD* type forms.

?column *x_column*

Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for *'twoD* type forms.

?width *x_width*

Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available.

For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an *x_width* of 1, and the last field has an *x_width* of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for *'twoD* type forms.

Default Value: 1

?coordinates *l_coordinates*

List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding *hi* field.) This argument is valid only for *'custom* type forms.

?displayOrder *x_displayOrder*

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first.

The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for *'oneD* type forms.

Valid Values: Any integer

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

?labelText *t_labelText*

Optional label displayed with frame type fields.

Default Value: *nil*

?private *s_private* Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software.

Valid Values: *t* (option does not appear in the UI), *nil* (option appears in the UI)

Default Value: *nil*

?display *s_display* Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: *t*

?editable *s_editable*

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: *t*

Note: This argument only applies to type-in fields.

?appCB *s_appCB* Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: (*o_session*)

?callback *t_callback*

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

?formApplyCB *s_formApplyCB*

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

`?changeCB st_changeCB`

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows` Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?browse g_browse`

Adds a browse button (...) for the specified option. This button can be used to open a file selection form to browse and select a file. The mode of file selection is specified by the `t_mode` argument.

Valid values: `t`, `nil`

Default value: `nil`

`?mode t_browseMode` Specifies mode for the file selection form that is displayed when the `g_browse` argument is set to `t`.

Valid values: `anyFile` specifies that you can open any file type; `existingFile` specifies that you can open any existing file; and `existingFiles` specifies that you can select multiple files.

Default value: `anyFile`

`?invalidateFunc s_invalidateFunc`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: (*o_session*)

?genericName *s_genericName*

Specifies the name of the generic variable that the option represents. This argument applies to temperature options only.

Valid Values: *temperature* specifies that the option represents the temperature for the simulator, *nominalTemp* specifies that the option represents the nominal temperature for the simulator

?sendMethod *s_sendMethod*

Specifies how simulator options are sent to Cadence SPICE. Use this argument only if your simulator is in the SPICE Socket.

Valid Values:

- *set* — Specifies options known by Cadence SPICE, for example, *tempdc* and *dcoppt*
- *ptprop* — Specifies numbers
- *psprop* — Specifies strings

Default Value: *ptprop*

Note: You can specify other values for the *sendMethod* argument. However, options must be sent to Cadence SPICE with the appropriate *set*, *ptprop*, or *psprop* statements.

Value Returned

o_envVar

Returns the simulator option object.

nil

Indicates an error.

Examples

```
asiAddSimOption( tool ?name 'gmin ?value "1e-12" )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Adds the `gmin` simulator option, which is set to the value of `1e-12`.

```
procedure( asiInit<yourSimulator>SimOption( tool )
;*****
; TOLERANCE OPTIONS
;*****
asiAddSimOption( tool
    ?name          'separator1
    ?type          'separator
)
asiAddSimOption( tool
    ?name          'label1
    ?type          'label
    ?prompt        "TOLERANCE OPTIONS"
)
asiAddSimOption( tool
    ?name          'reltol
    ?type          'string
    ?value         "1e-3"
)
asiAddSimOption( tool
    ?name          'conv
    ?type          'cyclic
    ?value         "off"
    ?choices       '( "off" "on" )
    ?sendMethod    'psprop
)
;*****
; TEMPERATURE OPTIONS
;*****

asiAddSimOption( tool
    ?name          'separator2
    ?type          'separator
)
asiAddSimOption( tool
    ?name          'label2
    ?type          'label
    ?prompt        "TEMPERATURE OPTIONS"
)
asiAddSimOption( tool
    ?name          'temp
    ?type          'string
    ?value         "27"
    ?genericName   'temperature
)
```

Adds the `reltol`, `conv`, and `temp` options. Separator lines are drawn between these sections on the form.

Note: This example is for a *"oneD"* form.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Related Topics

[asiDisplaySimOption](#)

[asiChangeSimOptionFormProperties](#)

asiChangeSimOption

```
asiChangeSimOption(  
    o_tool  
    [ ?name s_name ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
    [ ?genericName s_genericName ]  
    [ ?sendMethod s_sendMethod ]  
)  
=> o_envVar / nil
```

Description

Changes a simulator option.

Arguments

<i>o_tool</i>	Simulation tool object
?name <i>s_name</i>	Name of the simulator option you want to change.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?prompt t_prompt</code>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <code>s_name</code>
<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the form), <code>button</code> (for a button on the form), <code>scale</code> (for a slider field on the form) Default Value: <code>string</code> Note: See the asiAddSimOption function for an example that uses the <code>separator</code> argument.
<code>?choices l_choices</code>	List of choices if <code>s_type</code> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <code>s_type</code> is <code>toggle</code>. This argument is valid only if <code>s_type</code> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRows x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <code>l_choices</code>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies whether *g_value* can contain expressions.

Valid Values: *t* (value can contain expressions), *nil* (value cannot contain expressions)

Default Value: *nil*

?row *x_row*

Row in the form where the field appears. This argument is valid only for *twoD* type forms.

?column *x_column*

Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for *twoD* type forms.

?width *x_width*

Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an *x_width* of 1, and the last field has an *x_width* of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for *twoD* type forms.

Default Value: 1

?coordinates *l_coordinates*

List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding *hi* field.) This argument is valid only for *custom* type forms.

?displayOrder *x_displayOrder*

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first.

The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for *oneD* type forms.

Valid Values: Any integer

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?labelText t_labelText`

Optional label displayed with frame type fields.

Default Value: `nil`

`?private s_private` Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software.

Valid Values: `t` (option does not appear in the UI), `nil` (option appears in the UI)

Default Value: `nil`

`?display s_display` Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

`?editable s_editable`

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

Note: This argument only applies to type-in fields.

`?appCB s_appCB` Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: `(o_session)`

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

`?formApplyCB s_formApplyCB`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

`?changeCB st_changeCB`

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows` Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the *listBox* type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: `(o_session)`

`?genericName s_genericName`

Specifies the name of the generic variable that the option represents. This argument applies to temperature options only.

Valid Values: `temperature` specifies that the option represents the temperature for the simulator, `nominalTemp` specifies that the option represents the nominal temperature for the simulator

`?sendMethod s_sendMethod`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies how simulator options are sent to Cadence SPICE. (Use this argument only if your simulator is in the SPICE Socket.)

Valid Values:

- `set`: Specifies options known by Cadence SPICE (for example, `tempdc` and `dcoppt`).
- `ptprop`: Specifies numbers.
- `psprop`: Specifies strings.

Default Value: `ptprop`

Note: You can specify other values for the `sendMethod` argument. However, options must be sent to Cadence SPICE with the appropriate `set`, `ptprop`, or `psprop` statements.

Value Returned

`o_envVar`

Returns the changed simulator option object.

`nil`

Returns an error message and `nil` if unsuccessful.

Related Topics

[asiDisplaySimOption](#)

[asiChangeSimOptionFormProperties](#)

asiChangeSimOptionFormProperties

```
asiChangeSimOptionFormProperties(  
    o_tool  
    [ ?type s_type ]  
    [ ?width x_width ]  
    [ ?columns x_columns ]  
)  
=> o_formObj / nil
```

Description

Changes the display characteristics for the Simulator Options form.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<code>o_tool</code>	Simulation tool object
<code>?type s_type</code>	Specifies the form type. Valid Values: ■ <code>'oneD</code>: Specifies a sequential display of fields in one column. ■ <code>'twoD</code>: Specifies a two dimensional display of fields based on row and column positions. Use the <code>asiAddSimOption</code> or <code>asiChangeSimOption</code> function to specify values for the rows and columns. ■ <code>'custom</code>: Lets you specify the exact coordinate locations for each field. Use the <code>asiAddSimOption</code> or <code>asiChangeSimOption</code> function to specify the coordinates. ■ <code>'matrix</code>: Specifies a matrix of equally sized fields. Default Value: <code>'oneD</code>
<code>?width x_width</code>	Width of the form, in pixels. Default Value: The minimum default width of the form is 400 pixels. If the fields require more space than this, the form defaults to the smallest width that can accommodate the fields.
<code>?columns x_columns</code>	Number of columns. Use this argument only with matrix type forms. Default Value: 2

Value Returned

<code>o_formObj</code>	Returns the simulator option form object if successful.
<code>nil</code>	Indicates an error.

Examples

Changes the width of the Simulator Options form for the Spectre simulator.

```
asiChangeSimOptionFormProperties( asiGetTool('Spectre')?width 450 )
```


asiDeleteSimOption

```
asiDeleteSimOption(  
    o_tool  
    s_name  
)  
=> t / nil
```

Description

Deletes a simulator option.

Arguments

<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the simulator option you want to delete.

Value Returned

<i>t</i>	Returns <i>t</i> when the option is deleted.
<i>nil</i>	Returns an error message and <i>nil</i> if the option does not exist.

Examples

Removes the `delmax` simulator option.

```
asiDeleteSimOption( tool 'delmax )
```

Related Topics

[asiDisplaySimOption](#)

asiDisplaySimOption

```
asiDisplaySimOption(  
    { o_tool | o_session }  
)  
=> t / nil
```

Description

Displays the current set of simulator option names and values. Use this function only to determine which simulator options you want to modify. Do not use this function as part of another procedure.

Arguments

o_tool | *o_session* Simulation tool object

Value Returned

<i>t</i>	Displays the set of simulator option names and values and returns <i>t</i> .
<i>nil</i>	Indicates an error.

Examples

Displays the Spectre simulator options.

```
asiDisplaySimOption( asiGetTool( 'Spectre' ) )
```

asiDisplaySimOptionFormProperties

```
asiDisplaySimOptionFormProperties(  
    o_tool  
)  
=> t / nil
```

Description

Displays the characteristics of the Simulator Options form. Use this function only to determine which form characteristics you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object

Value Returned

<i>t</i>	Displays a list of Simulator Option form characteristics and returns <i>t</i> .
<i>nil</i>	Indicates an error.

Examples

Displays the form characteristics of the Spectre Simulator Options form and returns *t*.

```
asiDisplaySimOptionFormProperties( asiGetTool('Spectre') )
```

asiGetReservedWordList

```
asiGetReservedWordList(  
    o_session  
)  
=> l_list / nil
```

Description

Returns the simulator specific reserved keyword list. All the keywords specified by `asiGetReservedWordList` will be taken as keywords and not design variables. This process will ensure that the design parameters are not printed in the simulation control file under 'parameters' statement. By default, this function returns `nil`.

Note: While adding a simulator in Analog Design environment, you need to overload the function `asiGetReservedWordList`. Otherwise it will return the default value.

Arguments

<code>o_session</code>	Simulation tool object

Value Returned

<code>l_list</code>	Returns the simulator specific keyword list.
<code>nil</code>	By default, this function returns <code>nil</code> .

asIsCaseSensitive

```
asIsCaseSensitive(  
    o_session  
)  
=> t / nil
```

Description

Determines whether the simulator is case sensitive or not. Returns `t`, if the simulator is case sensitive, else returns `nil`. The default value for this function is `t`.

While adding a simulator in Analog Design environment, you need to overload the function `asIsCaseSensitive`. Otherwise it will return the default value.

Arguments

<code>o_session</code>	Simulation tool object

Value Returned

<code>t</code>	Determines whether the simulator is case sensitive or not and returns <code>t</code> .
<code>nil</code>	Indicates an error.

asiGetSimOptionChoices

```
asiGetSimOptionChoices(  
    { o_session | o_tool }  
    s_name  
)  
=> l_choices / nil
```

Description

Gets the list of choices for a simulator option that is set up as a list box.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the option for which you want the list of choices.

Value Returned

<i>l_choices</i>	Returns the list of values for the specified simulation option.
<i>nil</i>	Returns <i>nil</i> if the option does not exist.

Related Topics

[asiDisplaySimOption](#)

asiGetSimOptionNameList

```
asiGetSimOptionNameList(  
    o_tool  
)  
=> l_nameList
```

Description

Returns the list of simulator option names.

Arguments

<i>o_tool</i>	Simulation tool object

Value Returned

<i>l_nameList</i>	Returns a list of the names of the simulator options.

Examples

Loops through the simulator options and gets the corresponding values and `sendMethods` for those options.

```
foreach( name asiGetSimOptionNameList(session)  
    value = asiGetSimOptionVal( session name)  
    sendMethod = asiGetSimOptionSendMethod( session name)  
    ; Refer to asiSendOptions for a more complete example.  
)
```

asiGetSimOptionSendMethod

```
asiGetSimOptionSendMethod(  
    { o_session | o_tool }  
    s_name  
)  
=> s_sendMethod / nil
```

Description

Gets the `sendMethod` for the specified simulator option. The `sendMethod` indicates how the simulator option is sent to Cadence SPICE.

Arguments

<code>o_session</code>	Simulation session object
<code>o_tool</code>	Simulation tool object
<code>s_name</code>	Name of the simulator option.

Value Returned

<code>s_sendMethod</code>	Returns the <code>sendMethod</code> , which indicates how the simulator option is sent to Cadence SPICE.
<code>nil</code>	Returns <code>nil</code> if the option does not exist.

Examples

Gets the `sendMethod` for the *abstol* simulator option.

```
asiGetSimOptionSendMethod( session 'abstol )
```


asiGetSimOptionVal

```
asiGetSimOptionVal(  
    { o_session | o_tool }  
    s_name  
)  
=> g_value / nil
```

Description

Gets the value for the specified simulator option.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the simulator option.

Value Returned

<i>g_value</i>	Returns the value for the simulator option you specify.
<i>nil</i>	Returns <code>nil</code> if the option does not exist.

Examples

Returns the value for the *reltol* simulator option.

```
asiGetSimOptionVal( session 'reltol )
```

asiGetSimulationRunCommand

```
asiGetSimulationRunCommand(  
    o_session  
)  
=> g_command / nil
```

Description

Gets the simulation run command. For direct simulators, it also creates the `runSimulation` file.

Arguments

<i>o_session</i>	Simulation session object

Value Returned

<i>g_command</i>	Returns the simulation run command.
<code>nil</code>	Indicates an error.

Examples

For the spectre session `sess`, this command creates the `runSimulation` file in the netlist directory and returns `./runSimulation` as the simulation run command.

```
asiGetSimulationRunCommand( sess )
```

asiInit<yourSimulator>SimOption

```
asiInit<yourSimulator>SimOption(  
    o_tool  
)  
=> t
```

Description

Calls the procedures to add your simulator options.

Note: You must write `asiInit<yourSimulator>SimOption`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object

Value Returned

<code>t</code>	Returns <code>t</code> when your procedures are called.

Examples

Creates the procedure that calls the procedures to add the simulator options for the XYZ simulator.

```
procedure( asiInitXYZSimOption( tool )  
    <insert your code>  
    t  
)
```

asiIsICLPSupported

```
asiIsICLPSupported(  
    o_session  
)  
=> t / nil
```

Description

Specifies whether the simulator supports ICLP simulations that are enabled by the *Optimize Single Job Run* option on the Job Policy Setup form.

Arguments

<code>o_session</code>	Simulation session object.

Value Returned

<code>t</code>	When ICLP simulations are supported by the current simulator.
<code>nil</code>	When ICLP simulations are not supported by the current simulator.

Examples

The following function checks for the support for ICLP simulations in the given session:

```
asiIsICLPSupported( sess )  
=> t
```

asiSetHostOptions

```
asiSetHostOptions(  
    o_session  
    t_hostMode  
    [ t_host ]  
    [ t_remoteDir ]  
)  
=> t / nil
```

Description

Changes the host mode, host and remote directory for simulation.

Arguments

<i>o_session</i>	Simulation session object
<i>t_hostMode</i>	The type of simulation you want to select. Valid Values: <code>local</code> , <code>remote</code> and <code>distributed</code> . Default Value: <code>local</code>
<i>t_host</i>	Name of the host on which you want to run the digital simulator.
<i>t_remoteDir</i>	Specifies the project directory on the remote host to be used for remote simulation.

Value Returned

<i>t</i>	Returns <i>t</i> when successfully changes the host options.
<i>nil</i>	Indicates an error.

Examples

```
asiSetHostOptions( stdobj@0x18d7caa4 "local")  
=> t  
asiSetHostOptions( stdobj@0x18d7caa4 "distributed")  
=> t  
asiSetHostOptions(stdobj@0x18d7caa4 "remote" "ciclinux71" "/servers/scratch02/  
aakhil/testcase/simulation")  
=> t  
asiSetHostOptions(stdobj@0x18d7caa4 "remote")
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
WARNING (ADE-3042): Unable to contact host (machine)
=> nil
```

asiSetSimOptionChoices

```
asiSetSimOptionChoices(  
    { o_session | o_tool }  
    s_name  
    l_choices  
)  
=> l_choices / nil
```

Description

Specifies the list of choices to appear in the list box field for the specified simulator option.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name a simulator option of the type <code>listBox</code> .
<i>l_choices</i>	List of choices to appear in the list box field.

Value Returned

<i>l_choices</i>	Returns the new list of choices to appear in the list box field.
<i>nil</i>	Returns <code>nil</code> if the option does not exist.

asiSetSimOptionVal

```
asiSetSimOptionVal(  
    { o_session | o_tool }  
    s_name  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value of the specified simulator option.

Arguments

<i>o_session</i>	Simulation session object
<i>o_tool</i>	Simulation tool object
<i>s_name</i>	Name of the simulator option.
<i>g_value</i>	Value for the simulator option.

Value Returned

<i>g_value</i>	Returns the new value for the simulator option.
<i>nil</i>	Returns an error message and <code>nil</code> if the option does not exist.

Examples

Sets the *tempdc* simulator option to 35.

```
asiSetSimOptionVal( session 'tempdc "35" )
```


asiGetSimulatorSrcList

```
asiGetSimulatorSrcList(  
    o_session  
)  
=> l_result
```

Description

Customizes the values in the *Function* drop-down list of the Setup Analog Stimuli form.

Arguments

<i>o_session</i>	Simulation session object

Value Returned

<i>l_result</i>	List of values in the <i>Function</i> drop-down list of the Setup Analog Stimuli form.

Examples

```
defmethod ( asiGetSimulatorSrcList ( (session xyz_session) )  
'("dc" "pulse" "sin" "exp")  
)
```

Analysis Function

This section describes the functions that let you declare and manage simulator analyses.

If you want to use the `asiCheck` function to check values in your analyses or your analysis options, refer to [Miscellaneous Functions](#).

[`asiAddAnalysisField`](#)

[`asiAddAnalysisOption`](#)

[`asiAddAnalysisOption`](#)

[`asiAddSimOption`](#)

[`asiChangeAnalysis`](#)

[`asiChangeAnalysisField`](#)

[`asiChangeAnalysisOption`](#)

[`asiChangeAnalysisOptionFormProperties`](#)

[`asiCreateAnalysisField`](#)

[`asiCreateAnalysisOption`](#)

[`asiDeleteAnalysis`](#)

[`asiDeleteAnalysisField`](#)

[`asiDeleteAnalysisOption`](#)

[`asiDisableAnalysis`](#)

[`asiDisplayAnalysis`](#)

[`asiDisplayAnalysisField`](#)

[`asiDisplayAnalysisOption`](#)

[`asiDisplayAnalysisOptionFormProperties`](#)

[`asiEnableAnalysis`](#)

[`asiFormatAnalysis`](#)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiGetAnalysis

asiGetAnalysisFieldChoices

asiGetAnalysisFieldList

asiGetAnalysisFieldVal

asiGetAnalysisFormFieldChoices

asiGetAnalysisFormObj

asiGetAnalysisFormFieldVal

asiGetAnalysisName

asiGetAnalysisNameList

asiGetAnalysisOptionChoices

asiGetAnalysisOptionList

asiGetAnalysisOptionSendMethod

asiGetAnalysisOptionVal

asiGetAnalysisParamNameList

asiGetEnabledAnalysisList

asiInit<yourSimulator>Analysis

asilsAnalysisEnabled

asiSetAnalysisFieldChoices

asiSetAnalysisFieldVal

asiSetAnalysisFormFieldChoices

asiSetAnalysisFormFieldVal

asiSetAnalysisFormWidth

asiSetAnalysisOptionFormProperties

asiSetAnalysisOptionChoices

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

asiSetAnalysisOptionVal

asiAddAnalysis

```
asiAddAnalysis(  
    o_tool  
    [ ?name s_analysisName ]  
    [ ?prompt t_prompt ]  
    [ ?fieldList l_analysisFields ]  
    [ ?optionList l_analysisOptions ]  
    [ ?formType s_formType ]  
    [ ?enable s_enable ]  
)  
=> o_analysis / nil
```

Description

Adds a new analysis.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>?name s_analysisName</i>	Name of the analysis.
<i>?prompt t_prompt</i>	Prompt for the analysis on the form.
<i>?fieldList l_analysisFields</i>	List of analysis fields.
<i>?optionList l_analysisOptions</i>	List of analysis options.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?formType</code> <code>s_formType</code>	Specifies how the form is displayed. Valid values: ■ <code>'oneD</code> – Specifies a sequential display of fields in one column. ■ <code>'twoD</code> – Specifies a two dimensional display of fields based on row and column positions. The row and column positions are specified with <code>asiAddAnalysisField</code>, <code>asiChangeAnalysisField</code>, or <code>asiCreateAnalysisField</code>. ■ <code>'custom</code> – Lets you specify exact coordinate locations for each field. The coordinate locations are specified with <code>asiAddAnalysisField</code>, <code>asiChangeAnalysisField</code>, or <code>asiCreateAnalysisField</code>. Default Value: <code>'oneD</code>
<code>?enable s_enable</code>	Boolean flag that specifies whether the analysis is enabled by default. Valid Values: <code>t</code> specifies that the analysis is enabled by default, <code>nil</code> specifies that the analysis is not enabled by default. Default Value: <code>nil</code>

Value Returned

<code>o_analysis</code>	Returns the new analysis object if successful.
<code>nil</code>	Unsuccessful operation.

Examples

The following example adds a new analysis called *XYZanalysis*, which has a *from*, *to*, and *by* field, and *XYZ1* and *XYZ2* analysis options.

```
tool = asiGetTool('spectre)
defclass( spectre_XYZ_analysis (analysis) nil)
anaObj = asiAddAnalysis( tool
  ?name 'XYZ
  ?prompt "XYZ"
  ?fieldList list(
    asiCreateAnalysisField(
      ?name 'from
      ?prompt "from"
      ?value "0"
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
        ?row 1
        ?column 1
    )
    asiCreateAnalysisField(
        ?name 'to
        ?prompt "to"
        ?row 1
        ?column 2
    )
    asiCreateAnalysisField(
        ?name 'by
        ?prompt "by"
        ?row 1
        ?column 3
    )
)
?optionList list(
    asiCreateAnalysisOption(
        ?name 'XYZ1
        ?value "1e-4"
    )
    asiCreateAnalysisOption(
        ?name 'XYZ2
        ?value "1e-6"
    )
)
)
```

Related Topics

[asiDisplayAnalysis](#)

asiAddAnalysisField

```
asiAddAnalysisField(  
    o_analysis  
    [ ?name s_fieldName ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
)  
=> o_envVar / nil
```

Description

Adds an analysis field to an existing analysis.

Arguments

<i>o_analysis</i>	Analysis object to which you want to add a new field.
<i>?name s_fieldName</i>	Name of the analysis field to add.
<i>?prompt t_prompt</i>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <i>s_name</i>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> Note: See the function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>
<code>?row x_row</code>	Row in the form where the field appears. This argument is valid only for <i>twoD</i> type forms.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

?column *x_column* Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for *twoD* type forms.

?width *x_width* Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an *x_width* of 1, and the last field has an *x_width* of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for *twoD* type forms.

Default Value: 1

?coordinates *l_coordinates*

List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding *hi* field.) This argument is valid only for *custom* type forms.

?displayOrder *x_displayOrder*

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first.

The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for *oneD* type forms.

Valid Values: Any integer

?labelText *t_labelText*

Optional label displayed with frame type fields.

Default Value: *nil*

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?private s_private` Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software.

Valid Values: `t` (option does not appear in the UI), `nil` (option appears in the UI)

Default Value: `nil`

`?display s_display` Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

`?editable s_editable`

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

Note: This argument only applies to type-in fields.

`?appCB s_appCB` Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: `(o_session)`

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

`?formApplyCB s_formApplyCB`

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

`?changeCB st_changeCB`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows` Number of rows shown on the form for a `listBox` type field.

`?s_multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: `(o_session)`

Value Returned

`o_envVar`

Returns the environment variable object representing the field.

`nil`

Unsuccessful operation.

Examples

The following example adds a sweep field to an AC analysis:

```
o_analysisAC = asiGetAnalysis(tool 'ac)
asiAddAnalysisField( o_analysisAC
    ?name 'sweep
    ?prompt "Sweep"
    ?type 'cyclic
    ?choices '("Temperature" "Device Parameter"
               "Model Parameter")
    ?value "Model Parameter"
)
```

Related Topics

[asiDisplayAnalysisField](#)

asiAddAnalysisOption

```
asiAddAnalysisOption(  
    o_analysis  
    [ ?name s_optionName ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?browse g_browse ]  
    [ ?mode t_browseMode ]  
    [ ?invalidateFunc s_invalidateFunc ]  
    [ ?sendMethod s_sendMethod ]  
)  
=> o_envVar / nil
```

Description

Adds an option to an existing analysis.

Arguments

o_analysis Analysis object to which you want to add an option.
?name s_optionName Name of the analysis option to add.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?prompt t_prompt</code>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <code>s_name</code>
<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code> , <code>integer</code> , <code>float</code> , <code>toggle</code> , <code>cyclic</code> , <code>radio</code> , <code>Boolean</code> , <code>list</code> , <code>radioToggle</code> , <code>listBox</code> , <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> Note: See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <code>s_type</code> is <code>cyclic</code> , <code>radio</code> , <code>radioToggle</code> or <code>listBox</code> , or the list of switches if <code>s_type</code> is <code>toggle</code> . This argument is valid only if <code>s_type</code> is <code>cyclic</code> , <code>toggle</code> , <code>radio</code> , <code>radioToggle</code> , or <code>listBox</code> .
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code> , <code>cyclic</code> , <code>toggle</code> , and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <code>l_choices</code>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code> , <code>float</code> , or <code>scale</code> option. Default Value: <code>nil</code> , which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code> , <code>float</code> , or <code>scale</code> option. Default Value: <code>nil</code> , which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies whether *g_value* can contain expressions.

Valid Values: *t* (value can contain expressions), *nil* (value cannot contain expressions)

Default Value: *nil*

?row *x_row*

Row in the form where the field appears. This argument is valid only for *'twoD* type forms.

?column *x_column*

Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for *'twoD* type forms.

?width *x_width*

Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an *x_width* of 1, and the last field has an *x_width* of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for *'twoD* type forms.

Default Value: 1

?coordinates *l_coordinates*

List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding *hi* field.) This argument is valid only for *'custom* type forms.

?displayOrder *x_displayOrder*

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first.

The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for *'oneD* type forms.

Valid Values: Any integer

?labelText *t_labelText*

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Optional label displayed with frame type fields.

Default Value: `nil`

`?private s_private` Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software. Valid Values: `t` (option does not appear in the UI), `nil` (option appears in the UI)

Default Value: `nil`

`?display s_display` Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

`?editable s_editable`

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

Note: This argument only applies to type-in fields.

`?appCB s_appCB` Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: `(o_session)`

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

`?formApplyCB s_formApplyCB`

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?changeCB st_changeCB`

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a *listBox* type field.

`x_numRows`

Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?browse g_browse`

Adds a browse button (...) for the specified option. This button can be used to open a file selection form to browse and select a file. The mode of file selection is specified by the `t_mode` argument.

Valid values: `t`, `nil`

Default value: `nil`

`?mode t_browseMode` Specifies mode for the file selection form that is displayed when the `g_browse` argument is set to `t`.

Valid values: `anyFile` specifies that you can open any file type; `existingFile` specifies that you can open any existing file; and `existingFiles` specifies that you can select multiple files.

Default value: `anyFile`

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: (`o_session`)

`?sendMethod s_sendMethod`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies how simulator options are sent to Cadence SPICE. (Use this argument only if your simulator is in the SPICE Socket.)

Valid Values:

- `set` — Specifies options known by Cadence SPICE (for example, `tempdc` and `dcoppt`).
- `ptprop` — Specifies numbers.
- `psprop` — Specifies strings.

Default Value: `ptprop`

Note: You can specify other values for the *sendMethod* argument. However, options must be sent to Cadence SPICE with the appropriate `set`, `ptprop`, or `psprop` statements.

Value Returned

<code>o_envVar</code>	Returns the environment variable object representing the analysis option.
<code>nil</code>	Unsuccessful operation.

Examples

The following example adds the *XYZ2* analysis option to the *XYZ* analysis:

```
asiAddAnalysisOption( o_XYZ
?name 'XYZ2
?value "1e-6"
)
```

Related Topics

[asiDisplayAnalysisOption](#)

[asiChangeAnalysisOptionFormProperties](#)

asiChangeAnalysis

```
asiChangeAnalysis(  
    o_tool  
    [ ?name s_analysisName ]  
    [ ?prompt t_prompt ]  
    [ ?fieldList l_analysisFields ]  
    [ ?optionList l_analysisOptions ]  
    [ ?formType s_formType ]  
)  
=> o_analysis / nil
```

Description

Changes an existing analysis.

Arguments

<code>o_tool</code>	Simulation tool object.
<code>?name s_analysisName</code>	Name of the analysis you want to change.
<code>?prompt t_prompt</code>	Prompt for the analysis on the form.
<code>?fieldList l_analysisFields</code>	List of analysis fields.
<code>?optionList l_analysisOptions</code>	List of analysis options.
<code>?formType s_formType</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies how the form is displayed.

Valid Values:

- `'oneD`—Specifies a sequential display of fields in one column.
- `'twoD`—Specifies a two dimensional display of fields based on row and column positions. The row and column positions are specified with `asiAddAnalysisField`, `asiChangeAnalysisField`, or `asiCreateAnalysisField`.
- `'custom`—Lets you specify exact coordinate locations for each field. The coordinate locations are specified with `asiAddAnalysisField`, `asiChangeAnalysisField`, or `asiCreateAnalysisField`.

Default Value: `'oneD`

Value Returned

<code>o_analysis</code>	The analysis object.
<code>nil</code>	Unsuccessful operation.

Examples

The following example changes the form prompt for the AC analysis and adds two fields:

```
asiChangeAnalysis( tool
    ?name 'ac
    ?prompt "Modified AC Analysis"
    ?fieldList list(

        asiCreateAnalysisField(
            ?name 'model
            ?prompt "Model Name"
            ?row 4
            ?column 1
        )
        asiCreateAnalysisField(
            ?name 'modelParam
            ?prompt "Model Parameter"
            ?row 4
            ?column 2
        )
    )
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

)
)
)

Related Topics

[asiDisplayAnalysis](#)

asiChangeAnalysisField

```
asiChangeAnalysisField(  
    o_analysis  
    [ ?name s_fieldName ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
)  
=> o_envVar / nil
```

Description

Changes a field in an existing analysis.

Arguments

<i>o_analysis</i>	Analysis object with the field you want to change.
<i>?name s_fieldName</i>	Name of the analysis field to change.
<i>?prompt t_prompt</i>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <i>s_name</i>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> Note: See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. Note: This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?row x_row</code>	Row in the form where the field appears. Note: This argument is valid only for <i>'twoD'</i> type forms.
<code>?column x_column</code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. Note: This argument is valid only for <i>'twoD'</i> type forms.
<code>?width x_width</code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. Default Value: 1 Note: This argument is valid only for <i>'twoD'</i> type forms.
<code>?coordinates l_coordinates</code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) Note: This argument is valid only for <i>'custom'</i> type forms.
<code>?displayOrder x_displayOrder</code>	Position (from the top) of the option in the form. Use <i>x_displayOrder</i> to reposition your options in an <i>inherited</i> form. By default, the options appear in the order defined; therefore, inherited options appear first. The value of <i>x_displayOrder</i> must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. Valid Values: Any integer Note: This argument is valid only for <i>'oneD'</i> type forms.
<code>?labelText t_labelText</code>	Optional label displayed with frame type fields. Default Value: <code>nil</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?private s_private` Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software.

Valid Values: `t` (option does not appear in the UI), `nil` (option appears in the UI)

Default Value: `nil`

`?display s_display` Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

`?editable s_editable`

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

Note: This argument only applies to type-in fields.

`?appCB s_appCB` Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: `(o_session)`

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

`?formApplyCB s_formApplyCB`

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

`?changeCB st_changeCB`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows` Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: `(o_session)`

Value Returned

`o_envVar` A new environment variable object representing the analysis field.

`nil` Unsuccessful operation.

Examples

The following example changes the *from prompt* to *start*:

```
analysis = asiGetAnalysis(tool 'tran)
asiChangeAnalysisField(analysis
  ?name      'from
  ?prompt    "start"
)
```

Related Topics

[asiDisplayAnalysisField](#)

asiChangeAnalysisOption

```
asiChangeAnalysisOption(  
    o_analysis  
    [ ?name s_optionName ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
    [ ?sendMethod s_sendMethod ]  
)  
=> o_envVar / nil
```

Description

Changes an analysis option for an existing analysis.

Arguments

<i>o_analysis</i>	Analysis object with the option you want to change.
<i>?name s_optionName</i>	Name of the analysis option to change.
<i>?prompt t_prompt</i>	Optional argument that specifies the prompt (on the UI form) for the given option.
Default Value: <i>s_name</i>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code>. Note: See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. Note: This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?row x_row</code>	Row in the form where the field appears. Note: This argument is valid only for <i>twoD</i> type forms.
<code>?column x_column</code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. Note: This argument is valid only for <i>twoD</i> type forms.
<code>?width x_width</code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. Default Value: 1 Note: This argument is valid only for <i>twoD</i> type forms.
<code>?coordinates l_coordinates</code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) Note: This argument is valid only for 'custom type forms.
<code>?displayOrder x_displayOrder</code>	Position (from the top) of the option in the form. Use <i>x_displayOrder</i> to reposition your options in an <i>inherited</i> form. By default, the options appear in the order defined; therefore, inherited options appear first. The value of <i>x_displayOrder</i> must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. Valid Values: Any integer Note: This argument is valid only for <i>oneD</i> type forms.
<code>?labelText t_labelText</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

	Optional label displayed with frame type fields. Default Value: <code>nil</code>
<code>?private s_private</code>	Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software. Valid Values: <code>t</code> (option does not appear in the UI), <code>nil</code> (option appears in the UI) Default Value: <code>nil</code>
<code>?display s_display</code>	Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code>
<code>?editable s_editable</code>	Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code> Note: This argument only applies to type-in fields.
<code>?appCB s_appCB</code>	Specifies a callback function that is executed when the value of the option is changed. Callback parameter list: <code>(o_session)</code>
<code>?callback t_callback</code>	Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)
<code>?formApplyCB s_formApplyCB</code>	Specifies a callback function that is executed when the designer clicks on <i>Apply</i> or <i>OK</i> on the form containing the associated field. Callback parameter list: <code>(o_session r_form r_field)</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?changeCB` *st_changeCB* Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB` *st_doubleClickCB*

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows` *x_numRows* Number of rows shown on the form for a `listBox` type field.

`?multipleSelect` *s_multipleSelect*

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc` *s_invalidateFunc*

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: (*o_session*)

`?sendMethod` *s_sendMethod*

Specifies how simulator options are sent to Cadence SPICE. (Use this argument only if your simulator is in the SPICE Socket.)

Valid Values:

- `set`—Specifies options known by Cadence SPICE (for example, `tempdc` and `dcoppt`).
- `ptprop`—Specifies numbers.
- `psprop`—Specifies strings.

Default Value: `ptprop`

Note: You can specify other values for the *sendMethod* argument. However, your options will be automatically translated into the appropriate `set`, `ptprop`, or `psprop` statements to be sent to Cadence SPICE.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>o_envVar</code>	A new environment variable object that represents the analysis option.
<code>nil</code>	Unsuccessful operation.

Examples

The following example code modifies the *lvltim* transient option by adding an extra choice ("0") to the *choices* list:

```
asiChangeAnalysisOption( analysis
    ?name      'lvltim
    ?choices ' ( "0" "1" "2" )
)
```

Related Topics

[asiDisplayAnalysisOption](#)

[asiChangeAnalysisOptionFormProperties](#)

asiChangeAnalysisOptionFormProperties

```
asiChangeAnalysisOptionFormProperties(  
    o_analysis  
    [ ?type s_type ]  
    [ ?width x_width ]  
    [ ?columns x_columns ]  
)  
=> o_formObj / nil
```

Description

Changes the display characteristics for one of the analysis options forms.

Arguments

o_analysis Analysis object.

?type s_type Specifies the form type.

Valid Values:

- *'oneD*—Specifies a sequential display of fields in one column.
- *'twoD*—Specifies a two dimensional display of fields based on row and column positions. (Use the `asiAddAnalysisField`, `asiChangeAnalysisField`, or `asiCreateAnalysisField` functions to specify values for the rows and columns.)
- *'custom*—Lets you specify the exact coordinate locations for each field. (Use the `asiAddAnalysisField`, `asiChangeAnalysisField`, or `asiCreateAnalysisField` functions to specify the coordinates.)
- *'matrix*—Specifies a matrix of equally sized fields.

Default Value: *'oneD*

?width x_width Width of the form, in pixels.

Default Value: The minimum default width of the form is 400 pixels. If the fields require more space than this, the form defaults to the smallest width that can accommodate the fields.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?columns` *x_columns* Number of columns. Use this argument only with matrix type forms.

Default Value: 2

Value Returned

o_formObj Returns the analysis option form object if successful.

nil Unsuccessful operation.

Examples

```
asiChangeAnalysisOptionFormProperties( asiGetAnalysis( asiGetTool( 'Spectre )  
'tran ) ?width 500)
```

For the Spectre simulator, changes the width of the Transient Options form to 500 pixels.

Related Topics

[asiDisplayAnalysisOption](#)

[asiChangeAnalysisOptionFormProperties](#)

asiCreateAnalysisField

```
asiCreateAnalysisField(  
    [ ?name s_fieldName ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
)  
=> o_envVar / nil
```

Description

Creates a new analysis field, such as *from* or *to*, for a new or changed analysis.

You can call this function from within `asiAddAnalysis` or `asiChangeAnalysis` as an argument to *?fieldList*.

Arguments

<i>?names_s_fieldName</i>	Name of the analysis field to define.
<i>?prompt t_prompt</i>	Optional argument that specifies the prompt (on the UI form) for the given option.
	Default Value: <i>s_name</i>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> Note: See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>
<code>?row x_row</code>	Row in the form where the field appears. This argument is valid only for <i>twoD</i> type forms.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

?column *x_column* Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for *twoD* type forms.

?width *x_width* Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an *x_width* of 1, and the last field has an *x_width* of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for *twoD* type forms.

Default Value: 1

?coordinates *l_coordinates*

List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding *hi* field.) This argument is valid only for *custom* type forms.

?displayOrder *x_displayOrder*

Position (from the top) of the option in the form. Use *x_displayOrder* to reposition your options in an *inherited* form. By default, the options appear in the order defined; therefore, inherited options appear first.

The value of *x_displayOrder* must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for *oneD* type forms.

Valid Values: Any integer

?labelText *t_labelText*

Optional label displayed with frame type fields.

Default Value: *nil*

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?private s_private` Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software. Valid Values: `t` (option does not appear in the UI), `nil` (option appears in the UI)

Default Value: `nil`

`?display s_display` Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

`?editable s_editable`

Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on *any* field in the form.

Default Value: `t`

Note: This argument only applies to type-in fields.

`?appCB s_appCB` Specifies a callback function that is executed when the value of the option is changed.

Callback parameter list: `(o_session)`

`?callback t_callback`

Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)

`?formApplyCB s_formApplyCB`

Specifies a callback function that is executed when the designer clicks on *Apply* or *OK* on the form containing the associated field.

Callback parameter list: `(o_session r_form r_field)`

`?changeCB st_changeCB`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Specifies a callback function that is executed when the value of a `listBox` type field is changed on the form.

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows` Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the *listBox* type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: (*o_session*)

Value Returned

`o_envVar` Returns the environment variable object that represents the field.

`nil` Unsuccessful operation.

Examples

The following example adds a new analysis called XYZ analysis. This analysis has a *from*, *to*, and *by* field.

```
asiAddAnalysis( tool
  ?name 'XYZ
  ?fieldList list(
    asiCreateAnalysisField(
      ?name 'from
      ?prompt "from"
      ?value "0"
    )
    asiCreateAnalysisField(
      ?name 'to
      ?prompt "to"
    )
  )
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
        asiCreateAnalysisField(  
            ?name 'by  
            ?prompt "by"  
        )  
    )  
)
```

The following example shows another way to add a new XYZ analysis with a *from*, *to*, and *by* field.

```
fromField = asiCreateAnalysisField(  
    ?name 'from  
    ?prompt "from"  
    ?value "0"  
)  
toField = asiCreateAnalysisField(  
    ?name 'to  
    ?prompt "to"  
)  
byField = asiCreateAnalysisField(  
    ?name 'by  
    ?prompt "by"  
)  
asiAddAnalysis( tool  
    ?name 'XYZ  
    ?prompt "XYZ"  
    ?fieldList list( fromField toField byField )  
)
```

Related Topics

[asiDisplayAnalysisField](#)

asiCreateAnalysisOption

```
asiCreateAnalysisOption(  
    [ ?name s_optionName ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
    [ ?sendMethod s_sendMethod ]  
)  
=> o_envVar / nil
```

Description

Creates a new analysis option for a new or changed analysis. You can call this function from within `asiAddAnalysis` or `asiChangeAnalysis` as an argument to *?optionList*.

Arguments

<code>?name <i>s_optionName</i></code>	Name of the analysis option to define.
<code>?prompt <i>t_prompt</i></code>	Optional argument that specifies the prompt (on the UI form) for the given option.
Default Value: <i>s_name</i>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow</code> <code>x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>
<code>?row x_row</code>	Row in the form where the field appears. This argument is valid only for <i>twoD</i> type forms.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?column <i>x_column</i></code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for <i>'twoD</i> type forms.
<code>?width <i>x_width</i></code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for <i>'twoD</i> type forms. Default Value: 1
<code>?coordinates <i>l_coordinates</i></code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) This argument is valid only for <i>'custom</i> type forms.
<code>?displayOrder <i>x_displayOrder</i></code>	Position (from the top) of the option in the form. Use <i>x_displayOrder</i> to reposition your options in an <i>inherited</i> form. By default, the options appear in the order defined; therefore, inherited options appear first. The value of <i>x_displayOrder</i> must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for <i>'oneD</i> type forms. Valid Values: Any integer
<code>?labelText <i>t_labelText</i></code>	Optional label displayed with frame type fields. Default Value: <code>nil</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?private s_private</code>	Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software. Valid Values: <code>t</code> (option does not appear in the UI), <code>nil</code> (option appears in the UI) Default Value: <code>nil</code>
<code>?display s_display</code>	Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code>
<code>?editable s_editable</code>	Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code> This argument only applies to type-in fields.
<code>?appCB s_appCB</code>	Specifies a callback function that is executed when the value of the option is changed. Callback parameter list: (<code>o_session</code>)
<code>?callback t_callback</code>	Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)
<code>?formApplyCB s_formApplyCB</code>	Specifies a callback function that is executed when the designer clicks on <i>Apply</i> or <i>OK</i> on the form containing the associated field. Callback parameter list: (<code>o_session r_form r_field</code>)
<code>?changeCB st_changeCB</code>	Specifies a callback function that is executed when the value of a <code>listBox</code> type field is changed on the form.
<code>?doubleClickCB st_doubleClickCB</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

	Specifies a callback function that is executed when a designer double clicks on a <code>listBox</code> type field.
<code>?numRows x_numRows</code>	Number of rows shown on the form for a <code>listBox</code> type field.
<code>?multipleSelect s_multipleSelect</code>	Boolean flag that specifies whether multiple items can be selected from the <code>listBox</code> type field. Valid Values: <code>t</code> (multiple items can be selected), <code>nil</code> (only one item can be selected at a time)
<code>?invalidateFunc s_invalidateFunc</code>	Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart. Callback parameter list: (<code>o_session</code>)
<code>?sendMethod s_sendMethod</code>	Specifies how simulator options are sent to Cadence SPICE. (Use this argument only if your simulator is in the SPICE Socket.) Valid Values: ■ <code>set</code>—Specifies options known by Cadence SPICE (for example, <code>tempdc</code> and <code>dcoppt</code>). ■ <code>ptprop</code>—Specifies numbers. ■ <code>psprop</code>—Specifies strings. Default Value: <code>ptprop</code> You can specify other values for the <code>sendMethod</code> argument. However, options must be sent to Cadence SPICE with the appropriate <code>set</code>, <code>ptprop</code>, or <code>psprop</code> statements.

Value Returned

<code>o_envVar</code>	Returns the environment variable object representing the analysis option.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

nil Unsuccessful operation.

Examples

The following example adds a new analysis called *XYZanalysis* with *XYZ1* and *XYZ2* analysis options.

```
asiAddAnalysis( tool
    ?name 'XYZ
    ?prompt "XYZ"
    ?optionList list(
        asiCreateAnalysisOption(
            ?name 'XYZ1
            ?value "1e-4"
        )
        asiCreateAnalysisOption(
            ?name 'XYZ2
            ?value "1e-6"
        )
    )
)
```

The following example shows another way to add new XYZ analysis with *XYZ1* and *XYZ2* analysis options.

```
option1 = asiCreateAnalysisOption(
    ?name 'XYZ1
    ?value "1e-4"
)
option2 = asiCreateAnalysisOption(
    ?name 'XYZ2
    ?value "1e-6"
)
asiAddAnalysis(tool
    ?name 'XYZ
    ?prompt "XYZ"
    ?optionList list( option1 option2 )
)
```

Related Topics

[asiDisplayAnalysisOption](#)

[asiChangeAnalysisOptionFormProperties](#)

asiDeleteAnalysis

```
asiDeleteAnalysis(  
    o_tool  
    s_analysisName  
)  
=> t / nil
```

Description

Deletes an analysis from the given simulation tool.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>s_analysisName</i>	Name of the analysis you want to delete.

Value Returned

<i>t</i>	The analysis is deleted.
<i>nil</i>	Returns an error message and <i>nil</i> if the analysis does not exist.

Examples

The following example deletes an AC analysis from the Spectre simulator:

```
tool = asiGetTool('spectre')  
asiDeleteAnalysis(tool 'ac')
```

Related Topics

[asiDisplayAnalysis](#)

asiDeleteAnalysisField

```
asiDeleteAnalysisField(  
    o_analysis  
    s_fieldName  
)  
=> t / nil
```

Description

Deletes an analysis field from an existing analysis.

Arguments

<i>o_analysis</i>	Existing analysis object.
<i>s_fieldName</i>	Name of the analysis field to delete.

Value Returned

t	The analysis field is deleted.
nil	The field does not exist.

Examples

The following example deletes the *by* field from the existing transient analysis:

```
analysis = asiGetAnalysis(tool 'tran)  
asiDeleteAnalysisField(analysis 'by)
```

Related Topics

[asiDisplayAnalysisField](#)

asiDeleteAnalysisOption

```
asiDeleteAnalysisOption(  
    o_analysis  
    s_optionName  
)  
=> t / nil
```

Description

Deletes an analysis option.

Arguments

<i>o_analysis</i>	Analysis object with the option you want to delete.
<i>s_optionName</i>	Name of the analysis option to delete.

Value Returned

t	The option is deleted.
nil	The option does not exist.

Examples

The following example deletes the *tranOpt1* analysis option from the *tran* analysis:

```
analysis = asiGetAnalysis(tool 'tran)  
asiDeleteAnalysisOption(analysis 'tranOpt1)
```

Related Topics

[asiDisplayAnalysisOption](#)

asiDisableAnalysis

```
asiDisableAnalysis(  
    o_analysis  
)  
=> t / nil
```

Description

Disables an analysis while keeping it in the analysis list. The analysis remains in the UI, but it is not sent to the simulator.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

<i>t</i>	The analysis is disabled.
<i>nil</i>	The analysis does not exist.

Examples

The following example disables the *tran* analysis:

```
analysis = asiGetAnalysis(session 'tran)  
asiDisableAnalysis(analysis)
```

Related Topics

[asiEnableAnalysis](#)

asiDisplayAnalysis

```
asiDisplayAnalysis(  
    o_tool  
)  
=> t / nil
```

Description

Displays the analyses for a tool. Use this function to determine which analyses you need to add or modify. Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object.

Value Returned

<i>t</i>	Returns <i>t</i> and displays the available analyses.
<i>nil</i>	Unsuccessful operation.

Examples

The following example displays the analyses registered for the Analog class:

```
tool = asiGetTool('analog')  
asiDisplayAnalysis(tool)
```

Related Topics

[asiDisplayAnalysisOption](#)

asiDisplayAnalysisField

```
asiDisplayAnalysisField(  
    o_analysis  
)  
=> t / nil
```

Description

Displays the analysis field names for an analysis. Use this function to determine which analysis field you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_analysis</i>	Analysis object.

Value Returned

<i>t</i>	Returns <i>t</i> and displays the current analysis field names.
<i>nil</i>	Unsuccessful operation.

Examples

The following example displays the names of the analysis fields for the *tran* analysis:

```
analysis = asiGetAnalysis(tool 'tran)  
asiDisplayAnalysisField(analysis)
```

Related Topics

[asiDisplayAnalysis](#)

[asiDisplayAnalysisOption](#)

asiDisplayAnalysisOption

```
asiDisplayAnalysisOption(  
    o_analysis  
)  
=> t / nil
```

Description

Displays the analysis option names for an analysis. Use this function to determine which analysis option you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_analysis</i>	Analysis object.

Value Returned

<i>t</i>	Returns <i>t</i> and displays the current analysis option names.
<i>nil</i>	Unsuccessful operation.

Examples

The following example displays the analysis option names for the *tran* analysis:

```
analysis = asiGetAnalysis(tool 'tran)  
asiDisplayAnalysisOption(analysis)
```

Related Topics

[asiDisplayAnalysis](#)

asiDisplayAnalysisOptionFormProperties

```
asiDisplayAnalysisOptionFormProperties(  
    o_analysis  
)  
=> t / nil
```

Description

Displays the characteristics for one of the analysis options forms. Use this function only to determine which form characteristics you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_analysis</i>	Analysis object.

Value Returned

<i>t</i>	Displays the analysis options form characteristics.
<i>nil</i>	Unsuccessful operation.

Examples

The following example displays the form characteristics for the Transient Options form:

```
asiDisplayAnalysisOptionFormProperties(  
asiGetAnalysis( asiGetTool( 'Spectre ) 'tran))
```

Related Topics

[asiDisplayAnalysis](#)

[asiDisplayAnalysisOption](#)

asiEnableAnalysis

```
asiEnableAnalysis(  
    o_analysis  
)  
=> t / nil
```

Description

Enables an analysis, which means the analysis is selected and sent to the simulator.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

<i>t</i>	The analysis instance is enabled.
<i>nil</i>	The analysis does not exist.

Examples

The following example enables the *tran* analysis:

```
analysis = asiGetAnalysis(session 'tran)  
asiEnableAnalysis(analysis)
```

Related Topics

[asiDisableAnalysis](#)

asiFormatAnalysis

```
asiFormatAnalysis(  
    o_analysis  
    p_fp  
)  
=> t / nil
```

Description

Formats the analysis statements to send to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_analysis</i>	Specifies the analysis object.
<i>p_fp</i>	Specifies a file pointer to the file containing the analysis statements to send to the simulator (for simulators that are not in the Cadence SPICE Socket).

Value Returned

<i>t</i>	The statements are generated.
<i>nil</i>	Unsuccessful operation.

Examples

In the following example, the integrator of the XYZ simulator sends the trial analysis to Cadence SPICE, which formats the analysis to send to the XYZ simulator:

```
defmethod( asiFormatAnalysis ( (analysis XYZ_trial_analysis) fp )  
    let(( str ( aelEnv aelEnvCreate( 's ) ) (session asiGetSession( analysis ))  
    )  
  
        str = sprintf( nil "* Trial Analysis\n")  
        str = sprintf( nil "%sptprop XYZ_trial DoIt 1\n" str )  
  
        str = sprintf( nil "%sptprop XYZ_trial from %s\n" str  
            aelEnvInterpret( aelEnv asiGetAnalysisFieldVal  
                (analysis 'from ) ) )  
        str = sprintf( nil "%sptprop XYZ_trial to %s\n" str  
            aelEnvInterpret( aelEnv asiGetAnalysisFieldVal
```


Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
(analysis 'to ))))
str = sprintf( nil "%sptprop XYZ_trial by %s\n" str
  aelEnvInterpret( aelEnv asiGetAnalysisFieldVal
    (analysis 'by ))))

asiSendSim( session str nil nil nil )

asiFormatAnalysisOption( analysis fp )

t
)
)
```

asiGetAnalysis

```
asiGetAnalysis(  
    { o_session | o_tool }  
    s_analysisName  
)  
=> o_analysis / nil
```

Description

Gets an analysis object.

Arguments

<i>o_session</i>	Simulation session object.
<i>o_tool</i>	Simulation tool object.
<i>s_analysisName</i>	Name of the analysis you want to get.

Value Returned

<i>o_analysis</i>	The analysis object.
<i>nil</i>	The analysis does not exist.

Examples

The following example returns the *noise* analysis object:

```
analysis = asiGetAnalysis( session 'noise )
```

Related Topics

[asiDisplayAnalysis](#)

asiGetAnalysisFieldChoices

```
asiGetAnalysisFieldChoices(  
    o_analysis  
    s_fieldName  
)  
=> l_choices / nil
```

Description

Gets the list of choices for an analysis field that is set up as a list box.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_fieldName</i>	Name of the analysis field for which you want to get the list of choices.

Value Returned

<i>l_choices</i>	Returns the list of choices for the specified analysis field.
<i>nil</i>	The field does not exist.

Related Topics

[asiDisplayAnalysisField](#)

asiGetAnalysisFieldList

```
asiGetAnalysisFieldList(  
    o_analysis  
)  
=> l_fieldList / nil
```

Description

Returns a list of analysis field objects defined for a particular analysis object.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

<i>l_fieldList</i>	Returns the list of field objects for the specified analysis object.
<i>nil</i>	The analysis has no fields associated with it.

Examples

The following example returns the list of DC analysis fields:

```
dcAnal = asiGetAnalysis(session 'dc')  
asiGetAnalysisFieldList( dcAnal )
```

Related Topics

[asiDisplayAnalysisField](#)

asiGetAnalysisFieldVal

```
asiGetAnalysisFieldVal(  
    o_analysis  
    s_fieldName  
)  
=> g_value / nil
```

Description

Gets the value of an analysis field from the environment.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_fieldName</i>	Name of the analysis field for which you want to get the value.

Value Returned

<i>g_value</i>	Returns the value for the given field.
<i>nil</i>	The field does not exist.

Examples

The following example gets the value of the `from` field for an analysis:

```
analysis = asiGetAnalysis( session 'tran )  
asiGetAnalysisFieldVal( analysis 'from )
```

Related Topics

[asiDisplayAnalysisField](#)

asiGetAnalysisFormFieldChoices

```
asiGetAnalysisFormFieldChoices(  
    r_form  
    s_analysisName  
    s_fieldName  
)  
=> l_choices / nil
```

Description

Returns the list of choices for a field in the Choosing Analyses form. This procedure can be used within the *asiCheck* method to get the list of choices for an analysis field from the form for subsequent value checking. You can also use this procedure in an expression that controls whether a field is displayed in the form.

Arguments

<i>r_form</i>	Form containing the analysis field.
<i>s_analysisName</i>	Name of the analysis.
<i>s_fieldName</i>	Name of an analysis field of the type <code>listBox</code> .

Value Returned

<i>l_choices</i>	Returns the list of choices for the specified field on the Choosing Analyses form.
<code>nil</code>	The field does not exist.

Related Topics

[asiDisplayAnalysisField](#)

asiGetAnalysisFormObj

```
asiGetAnalysisFormObj(  
    o_session  
)  
=> o_analysis / nil
```

Description

Returns the current analysis selected in the *Choosing Analysis* form for the simulation session.

Arguments

<i>o_session</i>	Specifies a simulation session object.

Value Returned

<i>o_analysis</i>	Returns the current analysis.
<i>nil</i>	The session object is invalid.

Examples

The following example returns the name of the current analysis object selected in the cellview:

```
currentAnalysisObject = asiGetAnalysisFormObj(asiGetCurrentSession())  
=> stdobj@0x2da80938  
asiGetAnalysisName(currentAnalysisObject)  
=> "AC"
```

Related Topics

[asiGetAnalysis](#)

[asiGetAnalysisName](#)

asiGetAnalysisFormFieldVal

```
asiGetAnalysisFormFieldVal(  
    r_form  
    s_analysisName  
    s_fieldName  
)  
=> g_fieldValue / nil
```

Description

Returns the value of a field in the Choosing Analyses form. This procedure can be used within the *asiCheck* method to get the values of analysis fields from the form for subsequent value checking. You can also use this procedure in an expression that controls whether a field is displayed in the form.

Arguments

<i>r_form</i>	Form containing the analysis field.
<i>s_analysisName</i>	Name of the analysis.
<i>s_fieldName</i>	Name of the analysis field.

Value Returned

<i>g_fieldValue</i>	Returns the value for a field on the Choosing Analyses form.
<i>nil</i>	The field does not exist.

Examples

The following example shows how you might use the *asiGetAnalysisFormFieldVal* function to get the value of the *incrType* field (for an ac analysis) within the display expression

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

for the *lin* field. In this example, the *lin* field is displayed on the form only when *incrType* is *"Linear"*.

```
asiCreateAnalysisField(  
    ?name          'lin  
    ?prompt        "Step Size (Hz)"  
    ?display        'equal(asiGetAnalysisFormFieldVal( form 'ac  
                    'incrType ) "Linear")  
)
```

The first argument must be explicitly specified for proper evaluation.

Related Topics

[asiGetAnalysisFieldVal](#)

[asiCheck](#)

asiGetAnalysisName

```
asiGetAnalysisName(  
    o_analysis  
)  
=> s_analysisName / nil
```

Description

Gets the name of the analysis.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

<i>s_analysisName</i>	Returns the name of the analysis.
<i>nil</i>	The analysis is invalid.

Related Topics

[asiGetAnalysis](#)

asiGetAnalysisNameList

```
asiGetAnalysisNameList(  
    { o_session | o_tool }  
)  
=> l_analysesNames
```

Description

Returns a list of analysis names defined for a tool.

Arguments

<i>o_session</i>	Specifies a simulation session object.
<i>o_tool</i>	Specifies a simulation tool object.

Value Returned

<i>l_analysesNames</i>	Returns a list of analysis names.

Examples

The following example returns the list of analysis names for *o_session*:

```
listOfAnalysisNames = asiGetAnalysisNameList(o_session)
```

,

[asiGetAnalysis](#)

[asiGetAnalysisName](#)

asiGetAnalysisOptionChoices

```
asiGetAnalysisOptionChoices(  
    o_analysis  
    s_optionName  
)  
=> l_choices / nil
```

Description

Gets the list of choices for an analysis option that is set up as a list box.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_optionName</i>	Name of the option for which you want to get the list of choices.

Value Returned

<i>l_choices</i>	Returns the list of choices for the analysis option.
<i>nil</i>	The option does not exist.

Related Topics

[asiDisplayAnalysisOption](#)

asiGetAnalysisOptionList

```
asiGetAnalysisOptionList(  
    o_analysis  
)  
=> l_optionObjects / nil
```

Description

Gets a list of analysis option objects defined for a particular analysis object.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

<i>l_optionObjects</i>	Returns a list of analysis option objects.
<i>nil</i>	The analysis has no options.

Examples

The following example returns the list of analysis options:

```
asiGetAnalysisOptionList( analysis )
```

The following example shows how you might use the `asiGetAnalysisOptionList` routine in your `asiFormatAnalysisOption` routine:

```
defmethod( asiFormatAnalysisOption ( ( analysis XYZ_trial_analysis )  
    _fp )  
  
    let(( name value command sendMethod (session asiGetSession  
        (analysis )) )  
  
        foreach( option asiGetAnalysisOptionList( analysis )  
            name = asiGetName( option )  
            value = asiGetAnalysisOptionVal( analysis name )  
            sendMethod = asiGetAnalysisOptionSendMethod( analysis  
                name)  
  
            if( artBlankString( value ) then  
                sprintf( command "deprop XYZ_trial %s\n" name )  
            else  
                caseq( sendMethod
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
( set
  sprintf( command "set %s=%s\n" name
           value )
)
( ptprop
  sprintf( command "ptprop XYZ_trial
                 %s %s\n" name value )
)
( psprop
  sprintf( command "psprop XYZ_trial %s
                 \"%s\" \"\n\" name value )
)
( t
  warn( "don't know how to send XYZ trial
        option %s\n." name )
  command = nil
)
)

when( command
      asiSendSim( session command nil nil nil )
)
)
)
```

Related Topics

[asiDisplayAnalysisOption](#)

asiGetAnalysisOptionSendMethod

```
asiGetAnalysisOptionSendMethod(  
    o_analysis  
    s_optionName  
)  
=> s_sendMethod
```

Description

Gets the `sendMethod` for an option in an analysis. The `sendMethod` indicates how the analysis option is sent to Cadence SPICE.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_optionName</i>	Name of an analysis option for which you want to get the <i>sendMethod</i> .

Value Returned

<i>s_sendMethod</i>	Returns the <i>sendMethod</i> for the analysis option.

Examples

The following example returns the *sendMethod* for the *delmax* analysis option:

```
asiGetAnalysisOptionSendMethod( analysis 'delmax )
```

Related Topics

[asiGetAnalysis](#)

asiGetAnalysisOptionVal

```
asiGetAnalysisOptionVal(  
    o_analysis  
    s_optionName  
)  
=> g_value / nil
```

Description

Gets the value for the given option in an analysis.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_optionName</i>	Name of the option for which you want to get the value.

Value Returned

<i>g_value</i>	Returns the value for the analysis option.
<i>nil</i>	The option does not exist.

Examples

The following example returns the value for the *delmax* analysis option:

```
asiGetAnalysisOptionVal( analysis 'delmax )
```

Related Topics

[asiDisplayAnalysisOption](#)

[asiGetAnalysis](#)

asiGetAnalysisParamNameList

```
asiGetAnalysisParamNameList(  
    o_analysis  
)  
=> l_analysisParamNameList / nil
```

Description

Returns a concatenated list of fields and options defined for an analysis object.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

<i>l_analysisParamNameList</i>	A concatenated list of fields and options defined for the analysis object.
<i>nil</i>	The analysis has no fields or options associated with it.

Examples

The following function displays the names of fields and options defined for tran analysis in a list:

```
analysis = asiGetAnalysis( session 'tran )  
asiGetAnalysisParamNameList(analysis)
```

Related Topics

[asiGetAnalysis](#)

asiGetEnabledAnalysisList

```
asiGetEnabledAnalysisList(  
    o_session  
)  
=> l_analysisEnabledList / nil
```

Description

Returns a list of all the enabled analyses.

Arguments

<i>o_session</i>	Specifies a simulation session object.

Value Returned

<i>l_analysisEnabledList</i>	Returns a list of the enabled analyses.
<i>nil</i>	Returns nil if no analysis is enabled.

Examples

The following example returns the list of enabled analyses for *o_session*:

```
l_analysisEnabledList = asiGetEnabledAnalysisList(o_session)
```

Related Topics

[asiEnableAnalysis](#)

[asiDisableAnalysis](#)

asiInit<yourSimulator>Analysis

```
asiInit<yourSimulator>Analysis(  
    o_tool  
)  
=> t
```

Description

Calls the procedures that modify analyses in your simulator. You must write `asiInit<yourSimulator>Analysis`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	Successful operation operation. You must write this procedure to return <code>t</code> .

Examples

The following example creates a procedure that calls the procedures to specify or change the analyses for the *XYZ* simulator.

```
procedure( asiInitXYZAnalysis( tool )  
    <insert your code>  
    t  
)
```

asiIsAnalysisEnabled

```
asiIsAnalysisEnabled(  
    o_analysis  
)  
=> t / nil
```

Description

Determines whether an analysis is enabled.

Arguments

<i>o_analysis</i>	Specifies an analysis object.

Value Returned

t	The analysis is enabled.
nil	The analysis is not enabled.

Examples

The following example prints The transient analysis is enabled if tran is enabled:

```
analysis = asiGetAnalysis(session 'tran)  
when( asiIsAnalysisEnabled(analysis)  
println("The transient analysis is enabled")  
)
```

Related Topics

[asiEnableAnalysis](#)

[asiDisableAnalysis](#)

asiSetAnalysisFieldChoices

```
asiSetAnalysisFieldChoices(  
    o_analysis  
    s_fieldName  
    l_choices  
)  
=> l_choices / nil
```

Description

Specifies the list of choices to appear in the list box for an analysis field.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_fieldName</i>	Name of the analysis field for which you want to set the value.
<i>l_choices</i>	List of choices to appear in the list box field.

Value Returned

<i>l_choices</i>	Returns the new list of choices for the analysis field.
<i>nil</i>	The field does not exist.

asiSetAnalysisFieldVal

```
asiSetAnalysisFieldVal(  
    o_analysis  
    s_fieldName  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value for a field of an analysis.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_fieldName</i>	Name of the field for which you want to set the value.
<i>g_value</i>	Value for the field.

Value Returned

<i>g_value</i>	Returns the new value for the analysis field.
<i>nil</i>	The field does not exist.

Examples

The following example sets the `from` field to `5n`:

```
analysis = asiGetAnalysis(session 'tran)  
asiSetAnalysisFieldVal( analysis 'from "5n" )
```

Related Topics

[asiDisplayAnalysisField](#)

[asiGetAnalysis](#)

asiSetAnalysisFormFieldChoices

```
asiSetAnalysisFormFieldChoices(  
    r_form  
    s_analysisName  
    s_fieldName  
    l_choices  
)  
=> l_choices / nil
```

Description

Sets the list of choices for the specified field on the Choosing Analyses form.

Arguments

<i>r_form</i>	Form containing the analysis fields.
<i>s_analysisName</i>	Name of the analysis.
<i>s_fieldName</i>	Name of an analysis field of the type <code>listBox</code> .
<i>l_choices</i>	List of choices for the analysis field.

Value Returned

<i>l_choices</i>	Returns the new list of choices for the analysis field.
<code>nil</code>	The field cannot be accessed.

Related Topics

[asiGetAnalysisFormFieldChoices](#)

[asiGetAnalysis](#)

asiSetAnalysisFormFieldVal

```
asiSetAnalysisFormFieldVal(  
    r_form  
    s_analysisName  
    s_fieldName  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value of a field on the *Choosing Analyses* form.

Arguments

<i>r_form</i>	Form containing the analysis fields.
<i>s_analysisName</i>	Name of the analysis.
<i>s_fieldName</i>	Name of the analysis field.
<i>g_value</i>	Value for the analysis field.

Value Returned

<i>g_value</i>	Returns the new value for the analysis field.
<i>nil</i>	The field cannot be accessed.

Examples

The following example sets the `to` field to 10μ for a transient analysis:

```
asiSetAnalysisFormFieldVal( form 'tran 'to "10u")
```

Related Topics

[asiSetAnalysisFormFieldVal](#)

[asiGetAnalysis](#)

asiSetAnalysisFormWidth

```
asiSetAnalysisFormWidth(  
    o_tool  
    x_width  
)  
=> x_width / nil
```

Description

Sets the width of the Choosing Analyses form. You need to add a call to this procedure to the `analysis.il` file if you do not want the inherited width of the analysis form.

Arguments

<code>o_tool</code>	Simulation tool object.
<code>x_width</code>	Width, in pixels, of the form.

Value Returned

<code>x_width</code>	Returns <code>x_width</code> when the width is set.
<code>nil</code>	Unsuccessful operation.

Examples

The following example sets the width of the analysis form for the XYZ simulator to 500 pixels:

```
asiSetAnalysisFormWidth( asiGetTool( 'XYZ ' ) 500 )
```

Related Topics

[asiGetAnalysisFormFieldVal](#)

[asiSetAnalysisFormFieldVal](#)

[asiGetAnalysis](#)

asiSetAnalysisOptionFormProperties

```
asiSetAnalysisOptionFormProperties(  
    o_analysis  
    [ ?type s_type ]  
    [ ?width x_width ]  
    [ ?columns x_columns ]  
)  
=> o_formObj / nil
```

Description

Sets the display characteristics for a new analysis options form.

Arguments

<code>o_analysis</code>	Analysis object.
<code>?type s_type</code>	Specifies the form type. Valid Values: 'oneD—Specifies a sequential display of fields in one column. ■ 'twoD—Specifies a two dimensional display of fields based on row and column positions. (Use the <code>asiAddAnalysisOption</code>, <code>asiChangeAnalysisOption</code>, or <code>asiCreateAnalysisOption</code> functions to specify values for the rows and columns.) ■ 'custom—Lets you specify the exact coordinate locations for each field. (Use the <code>asiAddAnalysisOption</code>, <code>asiChangeAnalysisOption</code>, or <code>asiCreateAnalysisOption</code> functions to specify the coordinates.) ■ 'matrix—Specifies a matrix of equally sized fields. Default Value: 'oneD
<code>?width x_width</code>	Width of the form, in pixels. Default Value: The minimum default width of the form is 400 pixels. If the fields require more space than this, the form defaults to the smallest width that can accommodate the fields.
<code>?columns x_columns</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Number of columns. Use this argument only with matrix type forms.

Default Value: 2

Value Returned

o_formObj

Returns the analysis option form object if successful.

nil

Unsuccessful operation.

Examples

The following example sets the form properties for the analysis options for a new XYZ analysis:

```
asiSetAnalysisOptionFormProperties( asiGetAnalysis( asiGetTool( 'XYZ_simulator )  
'XYZ_analysis ) ?width 500)
```

Related Topics

[asiSetAnalysisFormFieldVal](#)

[asiGetAnalysis](#)

asiSetAnalysisOptionChoices

```
asiSetAnalysisOptionChoices(  
    o_analysis  
    s_optionName  
    l_choices  
)  
=> l_choices / nil
```

Description

Specifies the list of choices for an analysis option that is set up as a list box.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_optionName</i>	Name of an analysis option of the type <code>listBox</code> .
<i>l_choices</i>	List of choices for the analysis option.

Value Returned

<i>l_choices</i>	Returns the new list of choices for the analysis option.
<code>nil</code>	The option does not exist.

Related Topics

[asiSetAnalysisFormFieldVal](#)

[asiGetAnalysis](#)

asiSetAnalysisOptionVal

```
asiSetAnalysisOptionVal(  
    o_analysis  
    s_optionName  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value of an option in an analysis.

Arguments

<i>o_analysis</i>	Specifies an analysis object.
<i>s_optionName</i>	Name of the analysis option for which you want to set the value.
<i>g_value</i>	Value for the analysis option.

Value Returned

<i>g_value</i>	Returns the new value for the analysis option.
<i>nil</i>	The option does not exist.

Examples

The following example sets the value of the *lvltim* analysis option to 2:

```
asiSetAnalysisOptionVal( analysis 'lvltim "2" )
```

Related Topics

[asiDisplayAnalysisOption](#)

Simulation Control Functions for Direct Interfaces

The OASIS Procedural Interfaces (PI) functions are defined for direct simulation. Except as noted, you can redefine these methods.

We recommend that you not use or overload the following methods for OASIS direct simulation because they are either not applicable or they do not fit the direct simulation use model:

<code>asiSendAnalysis</code>	<code>asiSendControlStmts</code>
<code>asiSendInitCond</code>	<code>asiSendNetlist</code>
<code>asiSendNodeSets</code>	<code>asiSendOptions</code>
<code>asiSendRestore</code>	<code>asiSendDesignVars</code>
<code>asiSendKeepList</code>	<code>asiSendModelPath</code>
<code>asiSendInitFile</code>	<code>asiSendSim</code>
<code>asiSendUpdateFile</code>	<code>asiFinalNetlist</code>
<code>asiRawNetlist</code>	<code>asiAddFlowchartLink</code>
<code>asiAddFlowchartStep</code>	<code>asiCreateFlowchart</code>
<code>asiChangeFlowchartStep</code>	<code>asiDeleteFlowchartLink</code>
<code>asiDeleteFlowchartStep</code>	<code>asiExecuteFlowchart</code>
<code>asiDisplayFlowchart</code>	<code>asiGetFlowchart</code>
<code>asiInvalidateFlowchartStep</code>	<code>asiGetMarchList</code>
<code>asiSetMarchList</code>	

The `asiAnalog` Class: Initialization and Simulation Control

The `asiAnalog` class is the base class for all analog simulators. A simulator derived from the `asiAnalog` class is integrated directly without using Cadence SPICE for netlisting and simulation control.

Related Topics

[Functions to Customize Simulator Integration](#)

asiInitialize

```
asiInitialize(  
    o_tool  
)  
=> o_tool / nil
```

Description

Initializes the tools that are derived from the `asiAnalog` class. This function is not called for tools that are derived from the `asiSocket` class.

For the `asiAnalog` class, it calls:

- `asiInitStartOption`
- `asiInitEnvOption`
- `asiInitSimOption`
- `asiInitFormatterClass`
- `asiInitAnalysis`
- `asiInitUI`
- `asiInitDataAccessFunction`
- and other procedures

Arguments

<code>o_tool</code>	The simulation session object.

Value Returned

<code>o_tool</code>	The simulation session object.
<code>nil</code>	When unsuccessful.

Examples

The following example initializes the specified tool:

```
defmethod( asiInitialize (( yourSimulator_session ))
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
asiInitFormatterClass(tool)
asiInitEnvOption(tool)
asiInitAnalysis(tool)
)
```


asiNetlist

```
asiNetlist(  
    o_session  
)  
=> g_status / nil
```

Description

Creates a design object and then a formatter object with `nlCreateFormatter`. After this, the netlister is run and a netlist is generated with `nlNetlist`. Netlist statistics are then printed. The netlister also provides a component count, as well as the addition of design variables found during netlisting.

This function is called by the environment, so you should not call it directly from the interface. This function can be redefined for the interface. Use `callNextMethod` in this definition.

Arguments

<code>o_session</code>	The OASIS session object.

Value Returned

<code>g_status</code>	Successful creation of netlist.
<code>nil</code>	Netlisting errors were found.

Examples

The following example creates a design object for an OASIS session:

```
asiNetlist( session )
```

asiInterruptSim

```
asiInterruptSim(  
    o_session  
)  
=> g_status / nil
```

Description

This method provides an interrupt to the simulation run process for a session. It is associated with the *Simulation->Stop* action in the user interface. This method is called by the environment. Therefore, you should not call it directly from the interface. This method can be re-defined for the interface. Use *callNextMethod* in this definition.

This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as illustrated in the example. If you want to send a soft interrupt signal to your simulator, refer to the example mentioned in the *asiQuitSimulator* function.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>g_status</i>	Successful interruption of a simulation run process.
<i>nil</i>	Errors when interrupting the simulation.

Examples

The following example shows how to override a method to interrupt a simulation run process:

```
defmethod( asiInterruptSim (( session yourSimulator_session ))  
    println("yourSimulator : Simulation stopped by user")  
    <insert code you need >  
    callNextMethod()  
    <insert code you need >  
)
```

asiSetProjectDirChangeSetup

```
asiSetProjectDirChangeSetup(  
    o_session  
)  
=> t / nil
```

Description

Enables you to modify the simulator settings for the given session after changing the project directory. This function is called by the environment. Therefore, you should not call it directly from the interface.

This function is defined as a method for the Analog Class and returns `t` at analog class. You can overload this method for your simulator class, as illustrated in the example.

Arguments

<code>o_session</code>	Simulation session object.

Value Returned

<code>t</code>	The function call was successful.
<code>nil</code>	Unsuccessful operation.

Examples

The following example code shows how to modify the simulator settings for the given session:

```
defmethod ( asiSetProjectDirChangeSetup ( session yourSimulator_session ))  
    <insert code you need >  
    callNextMethod()  
    <insert code you need >  
)
```

asiQuitSimulator

```
asiQuitSimulator(  
    o_session  
    [ ?mode g_mode ]  
)  
=> g_status / nil
```

Description

This function quits the simulator run process based on the `ipc` signal being sent to the specified process. By default, the direct integration code sends a hard-kill signal to the simulator process. To change this option and send a soft-kill signal, the `?mode` option should be set to `true` and overloaded for `yourSimulator` session. This method is called by the environment, therefore, you should not call it directly from the interface. This method can be re-defined for the interface. Use, *callNextMethod* in its definition.

The function *asiQuitSimulator* is called internally via *asiInterruptSim*. To send a hard or soft interrupt signal, *asiInterruptSim* should be overloaded for your simulator class. This is illustrated in the example. In case you are not concerned with the hard kill scenario, it is recommended that you do not overload *asiInterruptSim* or *asiQuitSimulator* methods and use the default behavior.

Arguments

<code>o_session</code>	OASIS session object.
<code>g_mode</code>	Specifies the type of <code>ipc</code> signal sent to quit the simulator process. By default, <code>g_mode</code> is <code>nil</code> , meaning a hard-kill signal (<i>ipcKillProcess</i>) is sent to the simulator. When, <code>g_mode</code> is <code>t</code> , a soft-kill signal (<i>ipcSoftKill</i>) is sent to the simulator process. Valid Values: <code>nil</code> : a hard-kill signal (<i>ipcKillProcess</i>) is sent <code>t</code> : a soft-kill signal (<i>ipcSoftKill</i>) is sent Default Value: <code>nil</code>

Value Returned

<code>g_status</code>	The simulator successfully quits the simulator.
<code>nil</code>	Otherwise.

Examples

The following examples show how to overload the `asiInterruptSim` function and use `asiQuitSimulator` to quit the simulator run process based on the `ipc` signal:

Note: *asiQuitSimulator* is called from *asiInterruptSim*. Therefore, a simulator session can quit based on the type of signal received as follows:

■ Hard-kill signal (default, for `asiAnalog_session`)

In this case, you are not concerned with the `ipc` hard kill signal being sent to your simulator's session. There is no need to overload *asiQuitSimulator*. Instead, use the *callNextMethod()*, which will serve your purpose:

```
defmethod( asiInterruptSim ((session yourSimulator_session))
  println( "yourSimulator: session hard killed.\n" )
  println( "yourSimulator: simulation is stopped by user.\n" )
  println( "yourSimulator: simulation results may not be complete.\n" )
  asiSetWasInterrupted( session t )
  asiQuitSimulator( session )
)
```

□ Soft-kill signal (customized code)

When you need to have a soft-kill signal sent to your simulator's session, you need to overload *asiQuitSimulator* method with `?mode` set to `t`, as follows:

```
defmethod( asiInterruptSim ((session yourSimulator_session))
  <insert code you need>
  println( "yourSimulator: session soft killed.\n" )
  println( "yourSimulator: simulation is stopped by user.\n" )
  println( "yourSimulator: simulation results may not be complete.\n" )
  asiSetWasInterrupted( session t )
  asiQuitSimulator( session ?mode t )
  <insert code you need>
)
```

asiRunSimulation

```
asiRunSimulation(  
    o_session  
)  
=> g_status / nil
```

Description

This function runs the simulation for the session. It is called by the environment, so you should not call it directly from the interface. This method can be redefined for the interface. Use `callNextMethod` in this definition.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

<i>g_status</i>	The simulation is run successfully.
<i>nil</i>	Errors were encountered when starting the simulation.

Examples

The following example shows how to overload the `asiRunSimulation` function:

```
defmethod( asiRunSimulation (( session yourSimulator_session))  
    <insert code you need >  
    callNextMethod()  
    <insert code you need>
```

asiGetPredefinedCommandLineOption

```
asiGetPredefinedCommandLineOption(  
    o_session  
)  
=> t_predefinedCmdLineOption
```

Description

Gets the predefined simulation command-line options. This function returns an empty string at the `asiAnalog` class. Overload this method for your simulator.

Arguments

o_session The simulation session object.

Value Returned

t_predefinedCmdLineOption
The predefined command-line options in a string.

Examples

The following example shows how to overload the `asiGetPredefinedCommandLineOption` function:

```
defmethod( asiGetPredefinedCommandLineOption (  
    ( _session yourSimulator_session ))  
    "any command-line args that you always send to your simulator"  
)
```

asiGetCommandFooter

```
asiGetCommandFooter(  
    o_session  
)  
=> t_commandFooter
```

Description

Specifies the footer of the simulation run command.

Arguments

o_session Simulation session object.

Value Returned

t_commandFooter Command-line footer.

Examples

The following example shows how to overload the `asiGetCommandFooter` function:

```
defmethod( asiGetCommandFooter (( session mySimulator_session ))  
    " > mySimulator.log"  
)
```


Integrator Overloadable Methods for Formatting Control Statements

This topic lists the formatting control statements that you can overload for your simulators:

[asiFormatControlStmts](#)

[asiFormatDesignVarList](#)

[asiFormatInitCond](#)

[asiFormatNodeSet](#)

[asiFormatKeepList](#)

[asiFormatSimulatorOptions](#)

[asiFormatAnalysisList](#)

[asiFormatAnalysis](#)

[asiFormatModelLibSelectionList](#)

[asiFormatDefinitionFileList](#)

[asiFormatTextStimulusFileList](#)

asiFormatControlStmts

```
asiFormatControlStmts(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Creates and formats all control statements. It formats the following in the given order:

- node sets by calling `asiFormatNoteSet`
- initial conditions by calling `asiFormatInitCond`
- simulator options by calling `asiFormatSimulatorOptions`
- analyses by calling `asiFormatAnalysisList`
- nets/currents to save by calling `asiFormatKeepList`

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The file pointer.

Value Returned

<i>t</i>	The complete simulator input file was generated successfully.
<i>nil</i>	Input file is not generated.

Examples

The following example shows how to overload the `asiFormatControlStmts` function:

```
defmethod( asiFormatControlStmts (( session asiAnalog_session ) fp )  
callNextMethod()  
asiFormatMyControlStmt( session fp )  
)
```

asiFormatDesignVarList

```
asiFormatDesignVarList(  
    o_session  
)  
=> t / nil
```

Description

Formats and prints the design variable statements to the design variable file. This routine first prints the string `.PARAM` followed by the design variables in name=value pairs. The design variables are obtained by calling `asiGetDesignVarList`.

Arguments

<code>o_session</code>	The simulation session object.

Value Returned

<code>t</code>	The design variable statements are generated successfully.
<code>nil</code>	The design variable statements are not generated.

Examples

The following example shows how to overload the `asiFormatDesignVarList` function:

```
defmethod( asiFormatDesignVarList ((session asiAnalog_session) fp )  
let( ( (varList asiGetDesignVarList( session ))  
      mappedVarList mappedVar)  
when( varList  
  mappedVarList = asiGetDesignVarMappedList( session )  
  artFprintf( fp ".PARAM")  
  foreach( var varList  
    mappedVar = assoc( car(var) mappedVarList)  
    if( mappedVar  
      artFprintf( fp " %s" (cadr mappedVar ) )  
      artFprintf( fp " %s" car( var ) )  
    )  
    unless( (artBlankString (cadr var))  
      artFprintf( fp "=%s" (cadr var)))
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
    )  
    artFprintf( fp "\n" )  
  )  
  t  
)  
)
```

asiFormatInitCond

```
asiFormatInitCond(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats and prints the initial condition commands to the control statement file. This routine prints the string `.IC` followed by the initial conditions in `V(net)=voltage` pairs.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	The initial condition commands are formatted successfully.
<i>nil</i>	The initial condition commands are not formatted.

Examples

The following example shows how to overload the `asiFormatInitCond` function:

```
defmethod( asiFormatInitCond (( session asiAnalog_session ) fp )  
    let( ( icList asiGetInitCondList( session ) )  
        ( netlistDir asiGetAnalogNetlistDir( session ) ) )  
    when( icList  
        artFprintf(fp ".IC ")  
        foreach( obj icList  
            foreach( name  
                asiMapOutputName( netlistDir asiGetSelObjType( obj )  
                                asiGetSelObjName( obj ) )  
                artFprintf(fp "V( %s )=%s" name asiGetSelObjValue(obj))  
            )  
        )  
        artFprintf(fp "\n")
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
)  
  t  
)  
)
```

asiFormatNodeSet

```
asiFormatNodeSet(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats and prints the nodeset commands to the control statement file. This routine prints .NODESET and then the nodesets in V(net)=voltage pairs.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	The nodeset commands are formatted successfully.
<i>nil</i>	The nodeset commands are not formatted.

Examples

The following example shows how to overload the asiFormatNodeSet function:

```
defmethod( asiFormatNodeSet( ( session asiAnalog_session ) fp )  
    let( ( (nodeSetList asiGetNodeSetList( session ))  
        ( netlistDir asiGetAnalogNetlistDir( session ) ) )  
        when( nodeSetList  
            artFprintf(fp ".NODESET ")  
            foreach( obj nodeSetList  
                foreach( name  
                    asiMapOutputName( netlistDir asiGetSelObjType( obj )  
                                        asiGetSelObjName( obj ) )  
                    artFprintf(fp "V( %s )=%s" name  
                                asiGetSelObjValue(obj))  
                )  
            )  
        )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
        artFprintf(fp "\n")
    )
    t
)
)
```


asiFormatKeepList

```
asiFormatKeepList(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats and prints the signal save commands to the control statement file. At the `asiAnalog` class this routine returns `t`. You need to create your own `asiFormatKeepList` routine.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<code>t</code>	The save commands are formatted successfully.
<code>nil</code>	The save commands are not formatted.

Examples

The following example shows how to overload the `asiFormatKeepList` function:

```
defmethod( asiFormatKeepList ( ( session spectre_session ) fp )  
    let( ((netlistDir (asiGetAnalogNetlistDir session))  
        (keepList (asiGetKeepList session))  
        nameList )  
    when( keepList  
        artFprintf(fp "save ")  
        foreach( keep keepList  
            when( memq( asiGetSelObjType( keep ) '( net terminal ) )  
                nameList = asiMapOutputName( netlistDir  
                    asiGetSelObjType( keep ) asiGetSelObjName( keep ) )  
                foreach( name nameList  
                    artFprintf( "%s " name )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
        )
    )
)
artFprintf(fp "\n")
)
t
)
)
```

asiFormatSimulatorOptions

```
asiFormatSimulatorOptions(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats and prints the simulator option statements to the designated file. This routine prints `.OPTIONS` followed by name=value pairs.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	The simulator options are formatted successfully.
<i>nil</i>	The simulator options are not formatted.

Examples

The following example shows how to overload the `asiFormatSimulatorOptions` function:

```
defmethod( asiFormatSimulatorOptions  
    (( session asiAnalog_session ) fp )  
    let(( value (firstOption t))  
        foreach( option asiGetSimOptionList( session )  
            when( value = asiGetFormattedVal( option )  
                when( firstOption  
                    artFprintf(fp ".OPTIONS ")  
                    firstOption=nil  
                )  
                artFprintf(fp "%s=%s " asiGetName(option) value)  
            )  
        )  
    )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

t
)
)

asiFormatAnalysisList

```
asiFormatAnalysisList(  
    o_ana  
    p_fp  
)  
=> t / nil
```

Description

Formats all enabled analyses by calling `asiFormatAnalysis`.

Arguments

<i>o_ana</i>	The analysis object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	All enabled analyses are formatted successfully.
<i>nil</i>	All enabled analyses are not formatted.

Examples

The following example shows how to overload the `asiFormatAnalysisList` function:

```
defmethod(asiFormatAnalysisList ((session yourSimulator_session) fp)  
;; add your code  
callNextMethod()  
;; add your code  
t  
)
```

atAnalysis

```
asiFormatAnalysis(  
    o_ana  
    p_fp  
)  
=> t / nil
```

Description

Formats and prints analysis statements to the control file. For the general `asiAnalog` class, it follows this routine: prints the analysis name by calling `asiGetAnalysisName`; prints the list of signals by calling `asiGetAnalysisSigList` and formats them in parentheses [for example: `(net1 net2)`]; prints the analysis field list in name=value pairs; prints the analysis options in name=value pairs; uses `asiGetFormattedVal()` to obtain the print string for an analysis field value or an analysis option value. Please see the description of `asiGetFormattedVal` routine for more details.

Arguments

<i>o_ana</i>	The analysis object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	The analysis statements are formatted successfully.
<i>nil</i>	The analysis statements are not formatted.

Examples

The following example shows how to overload the `asiFormatAnalysis` function:

```
defmethod( asiFormatAnalysis ((ana asiAnalog_analysis) fp)  
    let(( name type sigList simVal (session asiGetSession(ana) ))  
        name = asiGetAnalysisName( ana )  
        sigList = asiGetAnalysisSigList( session ana )  
        ;;; prints analysis name  
        artFprintf(fp "%s " name )  
  
        ;;; handles analysis signals
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
when( sigList
    artFprintf( fp "( " )
    foreach( netField sigList
        when( simVal = asiGetFormattedVal( netField )
            artFprintf(fp "%s " simVal)
        )
    )
    artFprintf( fp ") " )
)

;;; prints analysis fields
foreach( f asiGetAnalysisSimFieldList( session ana )
    when( simVal = asiGetFormattedVal( f )
        artFprintf(fp "%s=%s " asiGetName(f) simVal)
    )
)

;;; prints analysis options
foreach( o asiGetAnalysisOptionList( ana )
    when( simVal = asiGetFormattedVal(o)
        artFprintf(fp "%s=%s " asiGetName(o) simVal)
    )
)

artFprintf(fp "\n")
t
)
)
```

asiFormatModelLibSelectionList

```
asiFormatModelLibSelectionList(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats the statement which specifies the model library information.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	All model library selections are formatted successfully.
<i>nil</i>	All model library selections are not formatted.

Examples

The generic method for asiAnalog class does nothing. The simulator interface has to define its own method, as shown in the following example:

```
defmethod(asiFormatAnalysisList ((session xyz_session) fp)  
foreach( obj asiGetModelLibSelectionList( session )  
    artFprintf( fp "include %L" asiGetModelLibFile(obj))  
    unless( artBlankString(section)  
        artFprintf( fp " section=%s" asiGetModelLibSection(obj))  
    )  
    artFprintf( fp "\n")  
    ) ; foreach  
t )
```


asiFormatDefinitionFileList

```
asiFormatDefinitionFileList(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats the statement which includes the specified definition files.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

t	All definition files are formatted successfully.
nil	All definition files are not formatted.

Examples

The following example shows how to overload the `asiFormatDefinitionFileList` function:

```
defmethod( asiFormatDefinitionFileList (( session xyz_session ) fp )  
    foreach( d asiGetDefinitionFileList(session)  
        artFprintf( fp "include '%s'\n" d )  
    )  
)
```

asiFormatTextStimulusFileList

```
asiFormatTextStimulusFileList(  
    o_session  
    p_fp  
)  
=> t / nil
```

Description

Formats the statement which includes the textual stimulus files.

Arguments

<i>o_session</i>	The simulation session object.
<i>p_fp</i>	The pointer to the control statement file.

Value Returned

<i>t</i>	All textual stimulus files are formatted successfully.
<i>nil</i>	All textual stimulus files are not formatted.

Examples

The following example shows how to overload the `asiFormatTextStimulusFileList` function:

```
defmethod(asiFormatTextStimulusFileList (( session xyz_session ) fp )  
    foreach( d asiGetTextStimulusFileList(session)  
        artFprintf( fp "include '%s'\n" d )  
    )  
)
```

asiNeedSuffixEvaluation

```
asiNeedSuffixEvaluation(  
    o_session  
)  
=> t / nil
```

Description

Specify whether the interface needs suffix evaluation or not. When this method returns *t*, the numeric suffixes specified in a *numericString* field will be evaluated. For example, suppose the start frequency field for the AC analysis has a value of 10M, `asiGetFormattedVal(ac_start_fieldObj)` returns 1e7 provided it is created as a 'numericString' field.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>t</i>	The Numeric suffix needs to be evaluated by the environment.
<i>nil</i>	The Numeric suffix need not be evaluated by the environment.

Examples

The following example shows how to overload the `asiNeedSuffixEvaluation` function:

```
defmethod(asiNeedSuffixEvaluation (( session xyz_session ) fp )  
nil  
)
```

In this case, the numeric suffix will not be evaluated.

asiInvalidateControlStmts

```
asiInvalidateControlStmts(  
    { o_session | o_tool }  
)  
=> t / nil
```

Description

The `asiInvalidateControlStmts` function is a wrapper to `asiInvalidateFlowchartStep`, which invalidates the `asiSendControlStmts` flowchart step.

Arguments

<code>o_session</code>	The simulation session object.
<code>o_tool</code>	The simulation session object.

Value Returned

<code>t</code>	Invalidates the <code>asiSendControlStmts</code> flowchart step.
<code>nil</code>	The step is not invalidated.

Examples

The following example shows how to overload the `asiInvalidateControlStmts` function:

```
defmethod( asiInvalidateControlStmts ( ( session Spectre_session ) )  
asiInvalidateFlowchartStep( session 'asiSendControlStmts )  
)
```

Utility Functions

This topic lists the various utility functions.

[asiGetSimExecName](#)

[asiGetCommandLineOption](#)

[asiGetAnalysisSigList](#)

[asiGetAnalysisType](#)

[asiGetAnalysisSimFieldList](#)

[asiGetModelLibSelectionList](#)

[asiGetModelLibFile](#)

[asiGetModelLibSection](#)

[asiGetDefinitionFileList](#)

[asiGetTextStimulusFileList](#)

[asiGetFormattedVal](#)

[asiGetSelObjName](#)

[asiGetSelObjType](#)

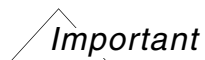
[asiGetSelObjValue](#)

[asiGetSelObjValue](#)

[asiMapOutputName](#)

[asiGetSimInputFileList](#)

[artlInvalidateAmap](#)



Do not overload the utility functions described in this section.

asiGetSimExecName

```
asiGetSimExecName(  
    o_session  
)  
=> t_simulatorExecutableName
```

Description

Gets the name of the simulator executable by calling `asiGetSimName`.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>t_simulatorExecutableName</i>	The simulator executable name.

Examples

```
defmethod( asiGetSimExecName (( yourSimulator_session ))  
    "your simulator executable name"  
)
```

asiGetCommandLineOption

```
asiGetCommandLineOption(  
    o_session  
)  
=> t_CommandLineOption
```

Description

Gets the simulation command line options. At the `asiAnalog` class this method returns the value of the environment option `'userCmdLineOption`.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>t_CommandLineOption</i>	The command line options in a string.

Examples

```
defmethod( asiGetCommandLineOption (   
    ( _session yourSimulator_session ) )  
strcat( asiGetPredefinedCommandLineOption( session )  
    asiGetEnvOptionVal( session 'userCmdLineOption )  
    )  
)
```

asiGetAnalysisSigList

```
asiGetAnalysisSigList(  
    o_session  
    o_ana  
)  
=> l_sigObjList
```

Description

Gets a list of analysis field objects which are of the type net. For example, the p and n nodes for the Spectre noise analysis.

Arguments

<i>o_session</i>	The simulation session object.
<i>o_ana</i>	The analysis object.

Value Returned

<i>l_sigObjList</i>	A list of analysis signal objects.

Examples

```
asiGetAnalysisSigList( session ana )
```


asiGetAnalysisType

```
asiGetAnalysisType(  
    o_analysis  
)  
=> s_analysisType
```

Description

Gets the type of the analysis.

Argument

<i>o_analysis</i>	Specifies an analysis object

Value Returned

<i>s_analysisType</i>	The type of the analysis

Example

```
asiGetAnalysisType( analysis )
```

asiGetAnalysisSimFieldList

```
asiGetAnalysisSimFieldList(  
    o_session  
    o_ana  
)  
=> l_simFieldObjList
```

Description

Gets a list of simulator analysis field objects that need to be netlisted.

Arguments

<i>o_session</i>	The simulation session object.
<i>o_ana</i>	The analysis object.

Value Returned

<i>l_simFieldObjList</i>	The list of simulator field objects.

Examples

```
asiGetAnalysisSimFieldList( session ana )
```

asiGetModelLibSelectionList

```
asiGetModelLibSelectionList(  
    o_session  
)  
=> l_modelLibSelectionList / nil
```

Description

Formats the statement which specifies the model library information.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>l_modelLibSelectionList</i>	The list of model library selection objects.
<i>nil</i>	Otherwise.

Examples

The following example shows how to get the list of model file names in the current ADE L session:

```
session = asiGetCurrentSession()  
=> stdobj@0x235ff080  
  
modelList = asiGetModelLibSelectionList( session )  
=> (("./Models/myModels.scs" "FF")  
   (("./Models/InlineModels.scs" "" )  
   (("./Models/mySingle.scs" "fastfast")  
   )  
;The following statement returns the model file name for the first model library  
in the current ADE L session  
asiGetModelLibFile(car(modelList))  
=> "./Models/myModels.scs"
```

asiGetModelLibFile

```
asiGetModelLibFile(  
    o_modelLibSelection  
)  
=> t_fileName / nil
```

Description

Gets the file name of a model library selection object.

Arguments

o_modelLibSelection The model library selection object.

Value Returned

<i>t_fileName</i>	The model file name.
<i>nil</i>	Unsuccessful operation.

Examples

The following example shows how to get the model file names for the first model library in the current ADE L session:

```
session = asiGetCurrentSession()  
=> stdobj@0x235ff080  
  
modelList = asiGetModelLibSelectionList( session )  
=> (("./Models/myModels.scs" "FF")  
   (("./Models/InlineModels.scs" "" )  
   (("./Models/mySingle.scs" "fastfast")  
   )  
  
asiGetModelLibFile(car(modelList))  
=> "./Models/myModels.scs"
```

asiGetModelLibSection

```
asiGetModelLibSection(  
    o_modelLibSelection  
)  
=> t_sectionName / nil
```

Description

Gets the section name of a model library selection object.

Arguments

o_modelLibSelection The model library selection object.

Value Returned

<i>t_sectionName</i>	The section name within a model library file.
<i>nil</i>	Otherwise.

Examples

The following example shows how to get the section name for the first model library in the current ADE L session.

```
session = asiGetCurrentSession()  
=> stdobj@0x235ff080  
  
modelList = asiGetModelLibSelectionList( session )  
=> (("./Models/myModels.scs" "FF")  
   (("./Models/InlineModels.scs" "" )  
   (("./Models/mySingle.scs" "fastfast")  
   )  
  
asiGetModelLibSection(car(modelList))  
=> "FF"
```

asiGetDefinitionFileList

```
asiGetDefinitionFileList(  
    o_session  
)  
=> l_definitionFileList / nil
```

Description

Gets the list of definition file names associated with the given simulation session.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>l_definitionFileList</i>	The list of definition file names.
<i>nil</i>	Unsuccessful operation.

Examples

```
asiGetDefinitionFileList( session )
```

asiGetTextStimulusFileList

```
asiGetDefinitionFileList(  
    o_session  
)  
=> l_definitionFileList / nil
```

Description

Gets the list of textual stimulus file names associated with the given simulation session.

Arguments

<i>o_session</i>	The simulation session object.

Value Returned

<i>l_stimFileList</i>	The list of textual stimulus file names.
<i>nil</i>	Otherwise.

Examples

```
asiGetTextStimulusFileList( session )
```

asiGetFormattedVal

```
asiGetFormattedVal(  
    o_anaField | o_anaOption | o_simOption  
)  
=> t_formattedVal / nil
```

Description

Gets the string value of an analysis field object, an analysis option object, or a simulator option object. The format of the string values are based on the object types which are specified with functions such as `asiCreateAnalysisField`, `asiCreateAnalysisOption`, `asiAddSimOption`. A number of examples can be found in the example section.

Arguments

<i>o_anaField</i>	The analysis field object.
<i>o_anaOption</i>	The analysis option object.
<i>o_simOption</i>	The simulator option object.

Value Returned

<i>t_formattedVal</i>	The formatted simulator value.
<i>nil</i>	An object should not be printed in the control statement file.

Examples

```
option = asiGetSimOptionList(session)  
value = asiGetFormattedVal(option)
```

For example, if a simulator option of the type 'literalString' is defined by :

```
asiAddSimOption( tool  
    ?name 'option  
    ?type 'literalString  
    ?value "value1"  
)
```

`asiGetFormattedVal()` for this option returns

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
"\"value1\""
```

When the simulator option is defined by:

```
asiAddSimOption( tool
    ?name 'option
    ?type 'literalString
    ?value "value1"
    ?literalStartMarker ""
    ?literalEndMarker ""
)
```

`asiGetFormattedVal()` for this option returns

```
""value1""
```

The following is an example of printing out a list:

```
asiAddSimOption( tool
    ?name 'option
    ?type 'list
    ?value '( 1 2 3 )
    ?startMarker "["
    ?endMarker "]" )
```

`asiGetFormattedVal()` for this option returns

```
[ 1 2 3 ]
```

The following example illustrate the `numericString` type formatting in relation to the Value Returned of `asiNeedSuffixEvaluation` method.

Given that a simulator option is defined by:

```
asiAddSimOption( tool
    ?name 'option1
    ?type 'numericString
    ?value "1M"
)
```

When `asiNeedSuffixEvaluation` method for `xyz` tool returns `t` then

`asiGetFormattedVal()` for this option returns `"1e6"`.

When `asiNeedSuffixEvaluation` method for `xyz` tool returns `nil` then

`asiGetFormattedVal()` for this option returns `"1M"`.

asiGetSelObjName

```
asiGetSelObjName (  
    o_selObj  
)  
=> t_name
```

Description

Returns the schematic name of the selected signal object.

Arguments

<i>o_selObj</i>	Selected signal object. Initial condition, nodeset, and keep objects are all selection objects.

Value Returned

<i>t_name</i>	Name of the selected signal object.

Examples

An example usage of this routine can be found in the example of `asiFormatInitCond`.

asiGetSelObjType

```
asiGetSelObjType(  
    o_selObj  
)  
=> t_signalType
```

Description

Returns the type of the selected signal object.

Arguments

<i>o_selObj</i>	Selected signal object. Initial condition, nodeset, and keep objects are all signal objects

Value Returned

<i>t_signalType</i>	Name of the selected signal object. The possible values are 'net', 'terminal', and 'instance'.

Examples

An example usage of this routine can be found in the example of `asiFormatInitCond`.

asiGetSelObjValue

```
asiGetSelObjValue(  
    o_selObj  
)  
=> t_value
```

Description

Returns the initial condition or nodeset values specified on the selected signal object.

Arguments

<i>o_selObj</i>	Selected signal object. Initial condition, nodeset, and keep objects are all signal objects

Value Returned

<i>t_value</i>	The initial condition value or nodeset voltage value specified on the selected signal object.

Examples

An example usage of this routine can be found in the example of `asiFormatInitCond`.

asiMapOutputName

```
asiMapOutputName (
    t_dir
    s_type
    t_name
    [ ?formatflag s_formatflag ]
)
=> l_nameList
```

Description

Maps the given schematic name of the given type using the netlist directory. The result is a list of mapped strings.

Arguments

<i>t_dir</i>	Netlist directory.
<i>s_type</i>	Type of object. Valid types are 'net, 'instance and 'terminal.
<i>t_name</i>	Schematic name.
<i>?formatflag s_formatflag</i>	Performs name mapping of transient simulation data available in SST2 format. Set the value to "t" if this function is used in context of transient simulation data analysis and the transient simulation data is available in SST2 format.

Value Returned

<i>l_nameList</i>	List of mapped names.
<i>nil</i>	Unsuccessful operation.

Examples

```
asiMapOutputName ( "netlistDir" 'net "/net13" )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Another example of the usage of this routine can be found in the example section of `asiFormatInitCond`.

asiGetSimInputFileList

```
asiGetSimInputFileList(  
    o_session  
)  
=> l_fileNamesList
```

Description

Returns a list of all file names concatenated to generate the input file to the simulator. You can override this method to add/delete files used to generate the final input file to your simulator.

Arguments

<i>o_session</i>	Specifies a simulation session object.

Value Returned

<i>l_fileNames</i>	List list of file names used to generate input file to the simulator.

Examples

```
defmethod( asiGetSimInputFileList (( session <yourSimulator>_session))  
    cons( "veriloga.inc" callNextMethod() )  
)
```

Adds the contents of file `veriloga.inc` to the final input file to the simulator.

artInvalidateAmap

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

Description

Resets the in-memory `Amap` cache. Further mapping function calls will result in re-reading the `amap` files from disk.

Arguments

None

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

Flowchart Functions

This topic lists the flowchart functions that let you modify the simulation flowchart for socket and direct interfaces.

You can modify the flowchart for the direct interface, if required. The flowchart describes the tool flow (the order in which certain events are expected to occur).

Related Topics

[Functions to Customize Simulator Integration](#)

asiAddFlowchartLink

```
asiAddFlowchartLink(  
    o_flowchart  
    s_parentStep  
    s_childStep  
)  
=> t / nil
```

Description

Creates a new link between the specified parent and child steps, which were created with `asiAddFlowchartStep`.

Arguments

<i>o_flowchart</i>	Flowchart object.
<i>s_parentStep</i>	Name of the parent step.
<i>s_childStep</i>	Name of the child step.

Value Returned

<i>t</i>	The link is created.
<i>nil</i>	Returns an error message and <i>nil</i> otherwise.

Examples

```
flowchart = asiGetFlowchart( session )  
asiAddFlowchartLink( flowchart 'asiRawNetlist  
    'asiRunSimulation )
```

Adds a flowchart link between the netlist and simulate steps.

asiAddFlowchartStep

```
asiAddFlowchartStep(  
    o_flowchart  
    [ ?name s_name ]  
    [ ?description t_description ]  
    [ ?runMessage t_runMessage ]  
    [ ?function s_function ]  
    [ ?checkFunc s_checkFunc ]  
    [ ?preFunc s_preFunc ]  
    [ ?postFunc s_postFunc ]  
    [ ?ignoreFunc s_ignoreFunc ]  
)  
=> o_step / nil
```

Description

Adds a new step to an existing flowchart.

Arguments

<i>o_flowchart</i>	Flowchart object to which you want to add a step.
<i>?name s_name</i>	Name of the step to add.
<i>?description t_description</i>	Textual description of the step.
<i>?runMessage t_runMessage</i>	Message to print when this step is executed.
<i>?function s_function</i>	The (primary) procedure to call to execute this step. Callback parameter list: (<i>o_session</i>)
<i>?checkFunc s_checkFunc</i>	Function to evaluate to update the status of this step. This function is executed after the dependency requirement is met. If this function returns <i>nil</i> , the step function and its children are invalidated. Callback parameter list: (<i>o_session</i>)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?preFunc s_preFunc</code>	Function to evaluate immediately before the primary procedure, <i>s_function</i> is executed. It can be added to a step instance to customize an existing step. Callback parameter list: (<i>o_session</i>)
<code>?postFunc s_postFunc</code>	Procedure to execute after <i>s_function</i> . Callback parameter list: (<i>o_session</i>)
<code>?ignoreFunc s_ignoreFunc</code>	Function to skip this step. If this function returns <i>t</i> , this step in flowchart is not executed.

Value Returned

<code>o_step</code>	Returns the flowchart step object.
<code>nil</code>	There is an error.

Examples

For the XYZ simulator, add a new step to send the ABC file to Cadence SPICE.

```
asiAddFlowchartStep( flowchart
?name 'sendABCfile
?description "Send the ABC file"
?runMessage "sending ABC file..."
?function 'sendABCfile
)
```

asiChangeFlowchartStep

```
asiChangeFlowchartStep(  
    o_flowchart  
    [ ?name t_name ]  
    [ ?description t_description ]  
    [ ?runMessage t_runMessage ]  
    [ ?function s_function ]  
    [ ?checkFunc s_checkFunc ]  
    [ ?preFunc s_preFunc ]  
    [ ?postFunc s_postFunc ]  
    [ ?ignoreFunc s_ignoreFunc ]  
)  
=> o_step / nil
```

Description

Changes a flowchart step in an existing flowchart

Arguments

<code>o_flowchart</code>	Flowchart object in which you want to modify a step.
<code>?name s_name</code>	Name of the step to be modified.
<code>?description t_description</code>	Textual description of the step.
<code>?runMessage t_runMessage</code>	Message to print when this step is executed.
<code>?function s_function</code>	The (primary) procedure to call to execute this step. Callback parameter list: (<code>o_session</code>)
<code>?checkFunc s_checkFunc</code>	Function to evaluate to update the status of this step. This function is executed after the dependency requirement is met. If this function returns <code>nil</code> , the step function and its children are invalidated. Callback parameter list: (<code>o_session</code>)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?preFunc s_preFunc</code>	Function to evaluate immediately before the primary procedure, <i>s_function</i> is executed. It can be added to a step instance to customize an existing step. Callback parameter list: (<i>o_session</i>)
<code>?postFunc s_postFunc</code>	Procedure to execute after <i>s_function</i> . Callback parameter list: (<i>o_session</i>)
<code>?ignoreFunc s_ignoreFunc</code>	Function to skip this step. If this function returns <i>t</i> , this step in flowchart is not executed.

Value Returned

<code>o_step</code>	Returns the new step object.
<code>nil</code>	The specified step does not exist.

Examples

Changes the run message for the *sendXYZFile* flowchart step.

```
asiChangeFlowchartStep( o_flowchart
    ?name sendXYZFile
    ?runMessage "Sending XYZ file"
)
```

Related Topics

[asiDisplayFlowchart](#)

asiCreateFlowchart

```
asiCreateFlowchart(  
    o_tool  
)  
=> o_flowchart
```

Description

Creates a new flowchart.

Arguments

<i>o_tool</i>	Simulation tool object.

Value Returned

<i>o_flowchart</i>	The flowchart object.

asiDeleteFlowchartLink

```
asiDeleteFlowchartLink(  
    o_flowchart  
    s_parentStep  
    s_childStep  
)  
=> t / nil
```

Description

Deletes the link between the specified parent and child steps.

Arguments

<i>o_flowchart</i>	Flowchart object.
<i>s_parentStep</i>	Name of the parent step.
<i>s_childStep</i>	Name of the child step.

Value Returned

<i>t</i>	The link between the steps is removed.
<i>nil</i>	Unsuccessful operation.

Examples

Removes the link connecting the `asiSendControlStmts` step to the `asiRunSimulation` step.

```
asiDeleteFlowchartLink( flowchart  
'asiSendControlStmts 'asiRunSimulation  
)
```

Related Topics

[asiDisplayFlowchart](#)

asiDeleteFlowchartStep

```
asiDeleteFlowchartStep(  
    o_flowchart  
    s_name  
    [ s_splice ]  
)  
=> t / nil
```

Description

Deletes a step and any attached links from an existing flowchart. Typically, you do not need this function because you can *unlink* any flowchart step that you do not want to use with the `asiDeleteFlowchartLink` function.

Arguments

<i>o_flowchart</i>	Flowchart object with the step you want to delete.
<i>s_name</i>	Name of the step to delete.
<i>s_splice</i>	Specifies whether the parents of the deleted step are to be linked directly to the children of the deleted step. Valid Values: <code>t</code> specifies that the new link is to be created, <code>nil</code> specifies that the new link is not to be created

Value Returned

<code>t</code>	The step is deleted.
<code>nil</code>	Unsuccessful operation.

Examples

Deletes the `asiSendOptions` step from flowchart for the XYZ simulation.

```
flowchart = asiGetFlowchart( asiGetTool( 'XYZ' ))  
asiDeleteFlowchartStep( flowchart 'asiSendOptions' )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Related Topics

[asiDisplayFlowchart](#)

[asiDeleteFlowchartLink](#)

asiDisplayFlowchart

```
asiDisplayFlowchart(  
    o_tool  
    [ s_rootstep ]  
)  
=> t / nil
```

Description

Displays the current steps and links for the flowchart. You can display all the step and link information, or you can display the steps and links that are below the *rootstep* step. Use this function only to determine which part of the flowchart you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>s_rootstep</i>	Name of the highest step (in the flowchart) to be displayed. The function returns this step and all the links and steps below it. If you do not specify this step, all the flowchart steps are displayed.

Value Returned

<i>t</i>	Returns <i>t</i> and displays a list of the current steps and links for the flowchart.
<i>nil</i>	Unsuccessful operation.

Examples

Returns the flowchart description for the *analog* tool.

```
asiDisplayFlowchart( asiGetTool( 'analog' ) )
```

asiExecuteFlowchart

```
asiExecuteFlowchart(  
    o_session  
    s_goalStep  
    [ s_printMessages ]  
)  
=> t / nil
```

Description

Executes the flowchart for a given session up to and including the goal step.

Arguments

<i>o_session</i>	Simulation session object.
<i>s_goalStep</i>	Name of the last flowchart step to execute.
<i>s_printMessages</i>	Specifies whether or not the <i>runmessages</i> for the steps are suppressed as they are executed. Valid Values: <i>t</i> displays the <i>runmessages</i> , <i>nil</i> suppresses the <i>runmessages</i> Default Value: <i>t</i>

Value Returned

<i>t</i>	The flowchart executes as far as the goal step.
<i>nil</i>	Indicates an error.

Examples

Executes the flowchart as far as the raw netlisting step.

```
asiExecuteFlowchart( session 'asiRawNetlist )
```

asiFinalNetlist

```
asiFinalNetlist(  
    o_session  
)  
=> t | l_dpl
```

Description

Creates the final netlist.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>t</i>	The final netlist is created (for analog only).
<i>l_dpl</i>	Returns a disembodied property list with the following properties (for mixed-signal only): ■ <i>netlistDir</i>—Name of the netlist directory. ■ <i>finalFileList</i>—List of final netlist files.

Examples

```
defmethod( asiFinalNetlist (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

asiGetFlowchart

```
asiGetFlowchart(  
    { o_tool | o_session }  
)  
=> o_flowchart
```

Description

Gets the flowchart object for a tool or session.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>o_session</i>	Simulation session object.

Value Returned

<i>o_flowchart</i>	Returns the flowchart object.

Examples

Returns the flowchart object for the *analog* tool.

```
flowchart = asiGetFlowchart( asiGetTool('analog') )
```

asiInit<yourSimulator>Flowchart

```
asiInit<yourSimulator>Flowchart(  
    o_tool  
)  
=> t
```

Description

Calls the procedures to initialize the flowchart for your simulator.

You must write `asiInit<yourSimulator>Flowchart`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	Returns <code>t</code> when your procedures are called. Write <code>asiInit<yourSimulator>Flowchart</code> to return <code>t</code> .

Examples

The following procedure initializes the flowchart for the XYZ simulator. It adds the step to send the library path, then inserts that step between the step to send the update file and the step to send the control statements.

```
procedure( asiInitXYZFlowchart( tool )  
    let( ( flowchart )  
  
        flowchart = asiGetFlowchart( session )  
  
        ;; SEND THE LIB PATH  
        asiAddFlowchartStep( flowchart  
            ?name 'asiSendLibPath  
            ?description "Sends the library path"  
            ?runMessage "send lib path"  
            ?function 'asiSendLibPath  
        )  
  
        asiDeleteFlowchartLink( flowchart 'asiSendUpdateFile
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
        'asiSendControlStmts )
    asiAddFlowchartLink(      flowchart 'asiSendUpdateFile
        'asiSendLibPath )
    asiAddFlowchartLink(      flowchart 'asiSendLibPath
        'asiSendControlStmts )
    t
)
)
```

Note: It is preferable to include the code for sending the `lib` path in the `asiSendControlStmts` step if possible. In other words, do not add extra links if they are not needed. (See the [asiSendControlStmts](#) function for an example.)

asiInvalidateFlowchartStep

```
asiInvalidateFlowchartStep(  
    o_session  
    s_step  
)  
=> t / nil
```

Description

Invalidates a flowchart step for a particular session.

You can use this function to invalidate a particular step that has become obsolete or out of date. This way, when a step that is dependent on the invalidated step must be executed, the system will first re-execute the invalidated step to make it current.

Arguments

<i>o_session</i>	Simulation session object.
<i>s_step</i>	Name of the flowchart step that you want to invalidate.

Value Returned

<i>t</i>	The flowchart step is invalidated.
<i>nil</i>	Unsuccessful operation.

Examples

Invalidates the netlisting step for the session.

```
asiInvalidateFlowchartStep( session 'asiRawNetlist )
```

Related Topics

[asiDisplayFlowchart](#)

asiRawNetlist

```
asiRawNetlist(  
    o_session  
)  
=> t / nil
```

Description

Creates a raw netlist.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>t</i>	The raw netlist is created.
<i>nil</i>	Unsuccessful operation.

Examples

```
defmethod( asiRawNetlist (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

asiSendAnalysis

```
asiSendAnalysis(  
    o_session  
)  
=> t / nil
```

Description

Sends analyses to Cadence SPICE by calling `asiFormatAnalysis` for each analysis.

Note for Integrators: This function is defined as a method for the Analog Class and is called by `asiSendControlStmts`. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>t</code>	The analyses are sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod( asiSendAnalysis (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

asiSendControlStmts

```
asiSendControlStmts(  
    o_session  
)  
=> t / nil
```

Description

Sends information such as nodesets, initial conditions, keep lists or output, analyses, restore files, include files, and stimulus files to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

t	The different commands are sent to Cadence SPICE.
nil	There is an error.

Examples

```
defmethod( asiSendControlStmts (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

You might want to send more information for your simulator. For example, you might send the information to send the lib path to your simulator. In this case, you can use `callNextMethod`, as shown in the following example:

```
defmethod( asiSendControlStmts (( session <yourSimulator>_session ))  
    ;Use callNextMethod to send nodesets, initial  
    ;conditions, force nodes, keepLists, and analyses  
  
    callNextMethod()  
    ;Now call your routine to send the lib path  
  
    <yourSimulator>SendLib( session )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

)

asiSendDesignVars

```
asiSendDesignVars(  
    o_session  
)  
=> t / nil
```

Description

Sends the design variables to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>t</i>	The design variables are sent to Cadence SPICE.
<i>nil</i>	There is an error.

Examples

```
defmethod( asiSendDesignVars (( session <yourSimulator>_session ))  
<insert any code you need>  
)
```

asiSendInitCond

```
asiSendInitCond(  
    o_session  
)  
=> t / nil
```

Description

Places all the initial conditions in `<netlistDirectory>/raw/ics` and sends a `ptprop` command to Cadence SPICE.

There is a mechanism in the `simdot.f` file that gets the value of the `ptprop` command (if the `ptprop` command was sent). If the `ptprop` command was sent, the `ics` file is automatically included in the final netlist. If the `ptprop` command was not sent, the `ics` file is not included. The initial conditions are of the form:

```
ic v(node)=value
```

Note for Integrators: This function is defined as a method for the Analog Class and is called by `asiSendControlStmts`. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>t</code>	The initial conditions are sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod( asiSendInitCond(( session <yourSimulator>_session ) )  
    let( ( icStr )  
  
        icStr = ".ic v(%s) = %s\n"  
        asiSendInitCond( session icStr )  
  
        t  
    )  
)
```

asiSendInitFile

```
asiSendInitFile(  
    o_session  
)  
=> t / nil
```

Description

Sends the `init.s` file to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>t</code>	The initialization file is sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod( asiSendInitFile (( session <yourSimulator>_session ))  
<insert any code you need>  
)
```


asiSendKeepList

```
asiSendKeepList(  
    o_session  
)  
=> t / nil
```

Description

Sends the keep list to Cadence SPICE. The keep list can contain a list of nets or currents to save and it can contain statements to *keep all nets* or *keep all currents*.

Note for Integrators: This function is defined as a method for the Analog Class and is called by `asiSendControlStmts`. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>t</code>	The keep list is sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod( asiSendKeepList (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

asiSendModelPath

```
asiSendModelPath(  
    o_session  
)  
=> t / nil
```

Description

Sends the model path to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>t</i>	The model path is sent to Cadence SPICE.
<i>nil</i>	There is an error.

Examples

```
defmethod( asiSendModelPath (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

asiSendNetlist

```
asiSendNetlist(  
    o_session  
)  
=> t / nil
```

Description

Sends the raw netlist to Cadence SPICE using the Cadence SPICE `sim` command.

For more information about the `sim` command, refer to the *Cadence SPICE Reference*.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>t</code>	The raw netlist is sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod( asiSendNetlist (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

asiSendNodeSets

```
asiSendNodeSets(  
    o_session  
)  
=> t / nil
```

Description

Places all the nodesets in `<netlistDirectory>/raw/nodesets` and sends a `ptprop` command to Cadence SPICE.

There is a mechanism in the `simdot.f` file that gets the value of the `ptprop` command (if the `ptprop` command was sent). If the `ptprop` command was sent, the `nodesets` file is automatically included in the final netlist. If the `ptprop` command was not sent, the `nodesets` file is not included. The nodesets are of the form:

```
.nodeset v(node)=value
```

Note for Integrators: This function is defined as a method for the Analog Class and is called by `asiSendControlStmts`. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.
<code>nil</code>	There is an error.

Value Returned

<code>t</code>	The nodesets are sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod(asiSendNodeSets(( session <yourSimulator>_session ))  
    let( ( nodesetStr )  
  
        nodesetStr = ".nodeset v(%s) = %s\n"  
        asiSendNodeSets( session nodesetStr )  
  
        t  
    )  
)
```

asiSendOptions

```
asiSendOptions(  
    o_session  
)  
=> t / nil
```

Description

Sends the simulation options to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>t</i>	The simulation options are sent to Cadence SPICE.
<i>nil</i>	There is an error.

Examples

```
defmethod( asiSendOptions ( ( session XYZ_session ) )  
    let( ( sendMethod simulatorOptionVariableList value command )  
  
        simulatorOptionVariableList = asiGetSimOptionNameList(  
            session )  
  
        foreach( name simulatorOptionVariableList  
  
            value = asiGetSimOptionVal( session name )  
            sendMethod = asiGetSimOptionSendMethod( session name )  
  
            if( artBlankString( value ) then  
                sprintf( command "deprop XYZ_Opt %s\n" name )  
            else  
                caseq( sendMethod  
                    ( set  
                        sprintf( command "set %s=%s\n"  
                            name value )  
                    )  
                    ( ptprop  
                        sprintf( command "ptprop XYZ_Opt  
                            %s %s\n" name value )  
                    )  
                )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
        ( psprop
          sprintf( command "psprop XYZ_Opt
                     %s \"%s\" \"\n"name value )
        )
        ( t
          warn( "don't know how to send XYZ
                option %s\n." name )
          command = nil
        )
      )
    )
  when( command
        asiSendSim( session command nil nil nil )
  )
)
t
)
)
```

asiSendRestore

```
asiSendRestore(  
    o_session  
)  
=> t
```

Description

If DC restore is *on*, send the commands to restore the DC node voltages to Cadence SPICE. If DC restore is *off*, sends the commands to turn off the DC restore function to Cadence SPICE. This function also works for the transient restore function.

Note for Integrators: This routine is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.

Value Returned

<i>t</i>	Sends the commands to Cadence SPICE.

Examples

```
defmethod( asiSendRestore ( ( session <yourSimulator>_session ) )  
    <insert any code you need>  
)
```

asiSendUpdateFile

```
asiSendUpdateFile(  
    o_session  
)  
=> t / nil
```

Description

Sends the `update.s` file to Cadence SPICE.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<code>o_session</code>	Specifies the session object.

Value Returned

<code>t</code>	The update file is sent to Cadence SPICE.
<code>nil</code>	There is an error.

Examples

```
defmethod( asiSendUpdateFile (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```


Keep Option Functions

The keep option functions let you specify whether or not *all* the outputs of a particular type of signal are saved during simulation. For example, you might use a keep option function to save all the node voltages or all the port currents.

For information about functions that let you save specific signals, refer to [Miscellaneous Functions](#).

Related Topics

[Functions to Customize Simulator Integration](#)

asiAddKeepOption

```
asiAddKeepOption(  
    o_tool  
    [ ?name s_name ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
)  
=> o_envVar / nil
```

Description

Adds a simulator keep option variable.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>?name s_name</i>	Name of the keep option.
<i>?prompt t_prompt</i>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <i>s_name</i>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an <code>integer</code>, <code>float</code>, or <code>scale</code> option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?row <i>x_row</i></code>	Row in the form where the field appears. This argument is valid only for <i>'twoD</i> type forms.
<code>?column <i>x_column</i></code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for <i>'twoD</i> type forms.
<code>?width <i>x_width</i></code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for <i>'twoD</i> type forms. Default Value: 1
<code>?coordinates <i>l_coordinates</i></code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) This argument is valid only for <i>'custom</i> type forms.
<code>?displayOrder <i>x_displayOrder</i></code>	Position (from the top) of the option in the form. Use <i>x_displayOrder</i> to reposition your options in an <i>inherited</i> form. By default, the options appear in the order defined; therefore, inherited options appear first. The value of <i>x_displayOrder</i> must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for <i>'oneD</i> type forms. Valid Values: Any integer
<code>?labelText <i>t_labelText</i></code>	Optional label displayed with frame type fields. Default Value: <i>nil</i>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?private s_private</code>	Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software. Valid Values: <code>t</code> (option does not appear in the UI), (option appears in the UI) Default Value: <code>nil</code>
<code>?display s_display</code>	Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code>
<code>?editable s_editable</code>	Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code> This argument only applies to type-in fields.
<code>?appCB s_appCB</code>	Specifies a callback function that is executed when the value of the option is changed. Callback parameter list: <code>(o_session)</code>
<code>?callback t_callback</code>	Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)
<code>?formApplyCB s_formApplyCB</code>	Specifies a callback function that is executed when the designer clicks on <i>Apply</i> or <i>OK</i> on the form containing the associated field. Callback parameter list: <code>(o_session r_form r_field)</code>
<code>?changeCB st_changeCB</code>	Specifies a callback function that is executed when the value of a <code>listBox</code> type field is changed on the form.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows`

Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: `(o_session)`

Value Returned

`o_envVar`

Returns the keep option environment variable object created by the procedure.

`nil`

There is an error.

Examples

Adds a keep option to the tool to save all digital node voltages.

```
asiAddKeepOption( tool
  ?name      'allDigitalNV
  ?prompt    "Select all digital node voltages"
  ?type      'Boolean
  ?value     t
)
```

Related Functions

[asiDisplayKeepOption](#)

[asiDisplayKeepOptionFormProperties](#)

asiChangeKeepOption

```
asiChangeKeepOption(  
    o_tool  
    [ ?name s_name ]  
    [ ?prompt t_prompt ]  
    [ ?type s_type ]  
    [ ?choices l_choices ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?value g_value ]  
    [ ?min g_min ]  
    [ ?max g_max ]  
    [ ?allowExpr s_allowExpr ]  
    [ ?row x_row ]  
    [ ?column x_column ]  
    [ ?width x_width ]  
    [ ?coordinates l_coordinates ]  
    [ ?displayOrder x_displayOrder ]  
    [ ?labelText t_labelText ]  
    [ ?private s_private ]  
    [ ?display s_display ]  
    [ ?editable s_editable ]  
    [ ?appCB s_appCB ]  
    [ ?callback t_callback ]  
    [ ?formApplyCB s_formApplyCB ]  
    [ ?changeCB st_changeCB ]  
    [ ?doubleClickCB st_doubleClickCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect s_multipleSelect ]  
    [ ?invalidateFunc s_invalidateFunc ]  
)  
=> o_envVar / nil
```

Description

Modifies an existing keep option variable for a simulator.

Arguments

<code>o_tool</code>	Simulation tool object.
<code>?name s_name</code>	Name of the keep option.
<code>?prompt t_prompt</code>	Optional argument that specifies the prompt (on the UI form) for the given option. Default Value: <code>s_name</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?type s_type</code>	Type of the option. Valid Values: <code>string</code>, <code>integer</code>, <code>float</code>, <code>toggle</code>, <code>cyclic</code>, <code>radio</code>, <code>Boolean</code>, <code>list</code>, <code>radioToggle</code>, <code>listBox</code>, <code>fileName</code> (string type for file names only), <code>label</code> (for the label on the UI form), <code>frame</code> (for a graphic frame around a field), <code>separator</code> (for the separator line on the UI form), <code>button</code> (for a button on the UI form), <code>scale</code> (for a slider field on the UI form) Default Value: <code>string</code> See the asiAddSimOption function for an example that shows how to use the <i>separator</i> argument.
<code>?choices l_choices</code>	List of choices if <i>s_type</i> is <code>cyclic</code>, <code>radio</code>, <code>radioToggle</code> or <code>listBox</code>, or the list of switches if <i>s_type</i> is <code>toggle</code>. This argument is valid only if <i>s_type</i> is <code>cyclic</code>, <code>toggle</code>, <code>radio</code>, <code>radioToggle</code>, or <code>listBox</code>.
<code>?itemsPerRow x_itemsPerRow</code>	Numbers of choices per row for <code>radio</code>, <code>cyclic</code>, <code>toggle</code>, and <code>radioToggle</code> fields. Default Value: Total number of choices specified in <i>l_choices</i>
<code>?value g_value</code>	Default value of the option.
<code>?min g_min</code>	Specifies the minimum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>-infinity</code>
<code>?max g_max</code>	Specifies the maximum value of an integer, float, or scale option. Default Value: <code>nil</code>, which means <code>+infinity</code>
<code>?allowExpr s_allowExpr</code>	Specifies whether <i>g_value</i> can contain expressions. Valid Values: <code>t</code> (value can contain expressions), <code>nil</code> (value cannot contain expressions) Default Value: <code>nil</code>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?row x_row</code>	Row in the form where the field appears. This argument is valid only for <i>'twoD</i> type forms.
<code>?column x_column</code>	Column in the form where the field appears. The fields are created according to their required widths and are not meant to align with fields in other rows of the form. This argument is valid only for <i>'twoD</i> type forms.
<code>?width x_width</code>	Specifies the width of the field in relation to other fields on the form. Numbers that you enter for this argument are relative values whose values are determined by the amount of space available. For example, assume there are three fields for a row that is 400 pixels wide. If the first two fields have an <i>x_width</i> of 1, and the last field has an <i>x_width</i> of 2, then the widths for the fields are as follows: 100, 100, and 200 pixels. This argument is valid only for <i>'twoD</i> type forms. Default Value: 1
<code>?coordinates l_coordinates</code>	List specifying the coordinates for the field on the UI form. (The format is the same as for the corresponding <i>hi</i> field.) This argument is valid only for <i>'custom</i> type forms.
<code>?displayOrder x_displayOrder</code>	Position (from the top) of the option in the form. Use <i>x_displayOrder</i> to reposition your options in an <i>inherited</i> form. By default, the options appear in the order defined; therefore, inherited options appear first. The value of <i>x_displayOrder</i> must be an integer that specifies the position you want relative to the top of the form. If more than one option has the same display order integer, the last option found (yours) takes precedence. The remaining options shift down on the form. This argument is valid only for <i>'oneD</i> type forms. Valid Values: Any integer
<code>?labelText t_labelText</code>	Optional label displayed with frame type fields. Default Value: <i>nil</i>

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?private s_private</code>	Optional argument that declares an option as private, which means the option is not visible to the user. Private options are not part of the UI and their values are not saved when a user saves the environment. You might use this argument for values that are constant for all users of the software. Valid Values: <code>t</code> (option does not appear in the UI), <code>nil</code> (option appears in the UI) Default Value: <code>nil</code>
<code>?display s_display</code>	Specifies an expression that determines whether the field is to be displayed on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code>
<code>?editable s_editable</code>	Specifies an expression that determines whether the field is to be editable on the UI form. This expression is evaluated when the form is first displayed and whenever a callback is executed on <i>any</i> field in the form. Default Value: <code>t</code> This argument only applies to type-in fields.
<code>?appCB s_appCB</code>	Specifies a callback function that is executed when the value of the option is changed. Callback parameter list: (<code>o_session</code>)
<code>?callback t_callback</code>	Specifies a callback function that is executed when a field is changed on the form while the form is displayed. (You can use this function to do error checking on the value entered by the designer.)
<code>?formApplyCB s_formApplyCB</code>	Specifies a callback function that is executed when the designer clicks on <i>Apply</i> or <i>OK</i> on the form containing the associated field. Callback parameter list: (<code>o_session r_form r_field</code>)
<code>?changeCB st_changeCB</code>	Specifies a callback function that is executed when the value of a <code>listBox</code> type field is changed on the form.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?doubleClickCB st_doubleClickCB`

Specifies a callback function that is executed when a designer double clicks on a `listBox` type field.

`?numRows x_numRows`

Number of rows shown on the form for a `listBox` type field.

`?multipleSelect s_multipleSelect`

Boolean flag that specifies whether multiple items can be selected from the `listBox` type field.

Valid Values: `t` (multiple items can be selected), `nil` (only one item can be selected at a time)

`?invalidateFunc s_invalidateFunc`

Specifies a function that is executed when the value of the option is changed. This function invalidates a step in the flowchart.

Callback parameter list: `(o_session)`

Value Returned

`o_envVar`

Returns the new keep option object.

`nil`

There is an error.

Examples

Changes the default value of the `allAnalogTC` keep option to `t`.

```
asiChangeKeepOption( asiGetTool('XYZ)
    ?name      'allAnalogTC
    ?value      t
)
```

Related Functions

[asiDisplayKeepOption](#)

[asiChangeKeepOptionFormProperties](#)

asiChangeKeepOptionFormProperties

```
asiChangeKeepOptionFormProperties(  
    o_tool  
    [ ?type s_type ]  
    [ ?width x_width ]  
    [ ?columns x_columns ]  
)  
=> o_formObj / nil
```

Description

Changes the display characteristics of the Keep Options form.

Arguments

<code>o_tool</code>	Simulation tool object.
<code>?type s_type</code>	Specifies the form type. Valid Values: ■ 'oneD—Specifies a sequential display of fields in one column. ■ 'twoD—Specifies a two dimensional display of fields based on row and column positions. (Use the <code>asiAddKeepOption</code> or <code>asiChangeKeepOption</code> function to specify values for the rows and columns.) ■ 'custom—Lets you specify the exact coordinate locations for each field. (Use the <code>asiAddKeepOption</code> or <code>asiChangeKeepOption</code> function to specify the coordinates.) ■ 'matrix—Specifies a matrix of equally sized fields.
<code>?width x_width</code>	Width of the form, in pixels. Default Value: The minimum default width of the form is 400 pixels. If the fields require more space than this, the form defaults to the smallest width that can accommodate the fields.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?columns x_columns` Number of columns. Use this argument only with matrix type forms.

Default Value: 2

Value Returned

`o_formObj` Returns the keep option form object if successful.

`nil` Indicates an error.

Examples

```
asiChangetKeepOptionFormProperties( asiGetTool('Spectre)
?width 300 )
```

For the Spectre simulator, changes the width of the Keep Options form to 300.

asiDeleteKeepOption

```
asiDeleteKeepOption(  
    o_tool  
    s_name  
)  
=> t / nil
```

Description

Deletes a simulator keep option variable.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>s_name</i>	Name of the keep option variable to delete.

Value Returned

<i>t</i>	Returns <i>t</i> when the keep option variable is deleted.
<i>nil</i>	The option does not exist.

Examples

```
asiDeleteKeepOption( asiGetTool('XYZ)  
    'allAnalogTC  
)
```

Deletes the keep option to save all analog terminal currents for the XYZ simulator.

asiDisplayKeepOption

```
asiDisplayKeepOption(  
    o_tool  
)  
=> t / nil
```

Description

Displays the current simulator keep option names. Use this function to determine which options you want to modify. Do not use this function as part of another procedure.

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	Displays the current keep option names and returns <code>t</code> .
<code>nil</code>	There is an error.

Examples

```
asiDisplayKeepOption( asiGetTool( 'Spectre' ))
```

Displays the keep option names for the *Spectre* simulator.

asiDisplayKeepOptionFormProperties

```
asiDisplayKeepOptionFormProperties(  
    o_tool  
)  
=> t / nil
```

Description

Displays the form characteristics for the Keep Options form. Use this function only to determine which form characteristics you want to modify. Do not use this function as part of another procedure.

Arguments

<i>o_tool</i>	Simulation tool object.

Value Returned

<i>t</i>	Displays a list of the Keep Option form characteristics and returns <i>t</i> .
<i>nil</i>	Indicates an error.

Examples

For the Spectre simulator, displays the form characteristics for the Keep Options form and returns *t*.

```
asiDisplayKeepOptionFormProperties( asiGetTool('Spectre'))
```


asiGetKeepOptionChoices

```
asiGetKeepOptionChoices(  
    { o_session | o_tool }  
    s_name  
)  
=> l_choices / nil
```

Description

Gets the list of choices for a keep option that is set up as a list box.

Arguments

<i>o_session</i>	Simulation session object.
<i>o_tool</i>	Simulation tool object.
<i>s_name</i>	Name of the option for which you want the list of choices.

Value Returned

<i>l_choices</i>	Returns the list of choices for the specified keep option.
<i>nil</i>	The option does not exist.

Related Function

[asiDisplayKeepOption](#)

asiGetKeepOptionVal

```
asiGetKeepOptionVal(  
    { o_tool | o_session }  
    s_name  
)  
=> g_value
```

Description

Gets the value of a keep option variable for a tool or session object.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>o_session</i>	Simulation session object.
<i>s_name</i>	Name of the simulation option variable for which you want to get the value.

Value Returned

<i>g_value</i>	Returns the value for the keep option variable.

Examples

Gets the value of the `allAnalogTC` keep option for the XYZ simulator.

```
asiGetKeepOptionVal( asiGetTool( 'XYZ' ) 'allAnalogTC' )
```

asiInit<yourSimulator>KeepOption

```
asiInit<yourSimulator>KeepOption(  
    o_tool  
)  
=> t
```

Description

Calls the procedures to initialize your simulator keep option variables.

Note: You must write `asiInit<yourSimulator>KeepOption`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>t</code>	Returns <code>t</code> when your procedures for initializing the keep option variables are called. Write <code>asiInit<yourSimulator>KeepOption</code> to return <code>t</code> .

Example

```
procedure( asiInitXYZKeepOption( tool )  
    <insert your code>  
    t  
)
```

Creates the procedure that initializes the keep options for the XYZ simulator.

asiSetKeepOptionChoices

```
asiSetKeepOptionChoices(  
    { o_session | o_tool }  
    s_name  
    l_choices  
)  
=> l_choices / nil
```

Description

Specifies the list of choices to appear in the list box field for the specified keep option.

Arguments

<i>o_session</i>	Simulation session object.
<i>o_tool</i>	Simulation tool object.
<i>s_name</i>	Name a keep option of the type <code>listBox</code> .
<i>l_choices</i>	List of choices to appear in the list box field.

Value Returned

<i>l_choices</i>	Returns the new list of choices to appear in the list box field.
<i>nil</i>	The option does not exist.

asiSetKeepOptionVal

```
asiSetKeepOptionVal(  
    { o_tool | o_session }  
    s_name  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value for the specified keep option variable for a tool or session object.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>o_session</i>	Simulation session object.
<i>s_name</i>	Name of the keep option variable that you want to set.
<i>g_value</i>	Value for the keep option variable.

Value Returned

<i>g_value</i>	Returns the new value for the keep option.
<i>nil</i>	The keep option does not exist.

Example

```
asiSetKeepOptionVal( session 'allAnalogNV t)
```

Saves all analog node voltages for the specified session.

Reliability Functions

This section describes the reliability functions that you can use to integrate the third party simulators into Virtuoso ADE Explorer. With the help of these SKILL functions, you can integrate your reliability flow and also modify the Reliability user interface.

calcValForRel

```
calcValForRel(  
    t_outputName  
    [ ?relxName t_relxName ]  
    [ ?ageValue t_ageValue ]  
    [ ?ageUnit t_ageUnit ]  
    [ ?she g_she ]  
    [ ?cornerName t_cornerName ]  
    [ ?historyName t_historyName ]  
    [ ?run t_runName ]  
)  
=> g_output / nil
```

Description

Retrieves the value of an output expression used in a reliability setup. You can use the value returned by this function in another output expression. When using the function `calcValForRel` in an output expression, you must specify different tests for stress and aged tests to avoid cyclic dependencies. You can create this expression in the *Outputs Setup* tab by using the Expression Builder too.

Video

You can watch a video demonstration on how to use the `calcValForRel` function at [Using the calcValForRel Function in Reliability Analysis](#). You can also read the related blog at [Virtuoso Video Diary: Enhancements in Reliability Analysis](#).

Arguments

`t_outputName` Name of the output to be used. The output can return a scalar value or waveform.

`?relxName t_relxName`
 Name of the reliability setup to be used.

`?ageValue t_ageValue`
 Specifies one of the following values:

- `Fresh`: For the fresh stage
- `Stress`: For stress stage
- `age_number`: For aged stage (Aging simulation)
 "`age_number`" specifies the age points at which the device degradation is calculated.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?ageUnit t_ageUnit`

Specifies the unit for the age points.

Valid Values: `yr`, `year`, `years`, `d`, `day`, `days`, `h`, `hour`, `hours`, `min`, `minute`, `minutes`, `sec`, `second`, `seconds`.

Default Value: `year`

`?she g_she`

Boolean value to specify whether the self-heating effect is enabled.

Valid Values:

- `t`: Self-heating effect is enabled
- `nil`: Self-heating effect is disabled

Default Value: `nil`

`?cornerName t_cornerName`

Name of the corner. When specified, the value of the given output for this corner is used.

Default Value: `nil`

`?historyName t_historyName`

Name of the history from which the results are to be retrieved. When specified, the scalar or waveform result of the output from the given history is used. Otherwise, the value of the output from the current simulation run is used.

The history must exist for the `calcValForRel` to return the expected result. To retrieve the value of an output of type `'signal'`, the simulation waveform results must be saved.

Default Value: `nil`

`?run t_runName`

Name of the run in the run plan from which the value of the given output is to be returned. This argument is useful only when the setup has a run plan.

Use this argument to get the value of an output from the results of one run (in the run plan) and use it in a variable or expression for another run in the same run plan or history. It cannot be used to refer to the results of another history. When both `?run` and `?historyName` are specified, `?historyName` is ignored by `calcValForRel`.

Default Value: `nil`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<i>g_output</i>	Returns the value of the specified output used in the given reliability setup.
<i>nil</i>	There is an error.

Examples

Retrieves the value of output *yMax* from "Stress" stage of the reliability setup "rel_0":

```
calcValForRel("yMax" ?relxName "rel_0" ?ageValue "stress")
```

Retrieves the value of output *yMax* from C2 corner for the age point 10 year in the reliability setup "rel_0":

```
calcValForRel("yMax" ?relxName "rel_0" ?ageValue "10" ?cornerName "C2")
```

Retrieves the value of output *yMax* from the reliability setup "rel_0" with the self-heating effect enabled.

```
calcValForRel("yMax" ?relxName "rel_0" ?she t)
```

Retrieves the value of output *yMax* from the history "Interactive.3". The history contains the simulation results of *fresh* stage in the reliability setup "rel_0":

```
calcValForRel("yMax" ?relxName "rel_0" ?ageValue "fresh" ?historyName  
"Interactive.3")
```

Retrieves the value of output *yMax* for the age point 20 days of gradual aging (10, 20, 30) in the reliability setup "rel_0":

```
calcValForRel("yMax" ?relxName "rel_0" ?ageValue "20" ?ageUnit "day")
```

Retrieves the value of output *yMax* from the reliability setup "rel_0" used in the run "Run.0":

```
calcValForRel("yMax" ?relxName "rel_0" ?run "Run.0")
```

maeGetStressFile

```
maeGetStressFile(  
    t_relxSetupName  
    [ ?cornerName t_cornerName ]  
    [ ?historyName t_historyName ]  
    [ ?run t_runName ]  
    [ ?matchParams l_matchParams ]  
)  
=> t_output / nil
```

Description

Returns the path of the stress file to be reused from the specified reliability setup.

Arguments

t_relxSetupName Name of the reliability setup from which the stress file is to be reused.

?cornerName t_cornerName
Name of the corner from which the stress file is to be reused.
If you have multiple corners in the test setup and you do not specify the corner name, the stress file generated for the first corner is reused.
Default Value: *nil*

?historyName t_historyName
Name of the history from which the stress file is to be reused.
If you do not specify the history, the stress file from the history of the last simulation run is reused.
Default Value: *nil*

?run t_runName Name of the run in the run plan for which the path of the stress file is to be reused. This argument is useful only when the setup has a run plan.
Default Value: *nil*

?matchParams l_matchParams
A list containing a list of name-value pairs of the sweep variables to identify the design point from which the stress file is to be reused.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<i>t_output</i>	The path of the stress file to be reused.
<i>nil</i>	There is an error.

Examples

Returns the path of the stress file that is reused from the reliability setup Rel1:

```
maeGetStressFile("Rel1")
=> "/servers/user/Reliability/simulation/library/cell/view/results/maestro/
history/1/Test/netlist/input.bs0"
```

Returns the path of the stress file that is reused from the C1 corner in the reliability setup Rel1:

```
maeGetStressFile("Rel1" ?cornerName "C1")
=>
"/servers/user/Reliability/simulation/library/cell/view/results/maestro/history/
2/Test/netlist/input.bs0"
```

Returns the path of the stress file that is reused from the history Interactive.2 for the reliability setup Rel1:

```
maeGetStressFile("Rel1" ?historyName "Interactive.2")
=>
"/servers/user/Reliability/simulation/library/cell/view/results/maestro/history/
3/Test/netlist/input.bs0"
```

Returns the path of the stress file that is reused from the run Run.0 for the reliability setup Rel1:

```
maeGetStressFile("Rel1" ?run "Run.0")
=>
"/servers/user/Reliability/simulation/library/cell/view/results/maestro/history/
4/Test/netlist/input.bs0"
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Consider the following setup in which the variable `cload` is a sweep variable. Note that the corner `C0` is also swept for different values of `vdc`.

The screenshot shows the Virtuoso ADE SKILL interface. On the left, the 'Data View' panel displays a tree of variables and corners. The 'Global Variables' section shows `vdc` with a value of 2.5, `cload` with a value of 1f 2f, and `temp1` with a value of 25. The 'Corners' section shows a 'Nominal' corner with `temperature` at 30 and `vdc` at 1.8 2.0 2.2. The 'Reliability Analyses' section shows 'Reliability0' selected. On the right, the 'Results' panel displays a table of test results for 'Oscillator_analysis' across different parameters and corners. The table has columns for Point, Age, Test, Output, Nominal, Spec, Weight, Pass/Fail, Min, Max, and three corner columns (C0_0, C0_1, C0_2). The results show that the test passes for all parameters and corners.

Point	Age	Test	Output	Nominal	Spec	Weight	Pass/Fail	Min	Max	C0_0	C0_1	C0_2
1	stress	Oscillator_analysis	/out	802.8p	< 1.5n		pass	611.7p	802.8p	709.3p	611.7p	648.1p
1	stress	Oscillator_analysis	RiseTime	2.474n	< 6n		pass	255.5p	2.474n	255.5p	1.289n	1.667n
1	10 yr	Oscillator_analysis	/out	845.4p	< 1.5n		pass	535p	845.4p	714.1p	535p	664.8p
1	10 yr	Oscillator_analysis	RiseTime	2.698n	< 6n		pass	265.3p	2.698n	265.3p	1.308n	1.731n
2	stress	Oscillator_analysis	/out	846.1p	< 1.5n		pass	646.5p	846.1p	709.7p	646.5p	684.5p
2	stress	Oscillator_analysis	RiseTime	2.699n	< 6n		pass	361.7p	2.699n	361.7p	1.439n	1.847n
2	10 yr	Oscillator_analysis	/out	894.1p	< 1.5n		pass	566.2p	894.1p	714.6p	566.2p	702.5p
2	10 yr	Oscillator_analysis	RiseTime	2.938n	< 6n		pass	372.6p	2.938n	372.6p	1.462n	1.916n

When you run the reliability setup `Reliability0`, six stress files are generated for the following design points:

- `cload=1f, vdc=1.8 (C0_0)`
- `cload=2f, vdc=1.8 (C0_0)`
- `cload=1f, vdc=2.0 (C0_1)`
- `cload=2f, vdc=2.0 (C0_1)`
- `cload=1f, vdc=2.2 (C0_2)`
- `cload=2f, vdc=2.2 (C0_2)`

To identify the design point from which you want to reuse the stress file, use the `?matchParams` argument as follows:

```
maeGetStressFile("Reliability0" ?matchParams list(list("cload" "1f") list("vdc" "1.8")))
maeGetStressFile("Reliability0" ?matchParams list(list("cload" "2f") list("vdc" "1.8")))
maeGetStressFile("Reliability0" ?matchParams list(list("cload" "1f") list("vdc" "2.0")))
maeGetStressFile("Reliability0" ?matchParams list(list("cload" "2f") list("vdc" "2.0")))
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
maeGetStressFile("Reliability0" ?matchParams list(list("cload" "1f") list("vdc"
"2.2")))
maeGetStressFile("Reliability0" ?matchParams list(list("cload" "2f") list("vdc"
"2.2")))
```

maeAddRelxSetup

```
maeAddRelxSetup(  
    t_relxSetupName  
    t_freshTest  
    t_agedTest  
    [ ?stressTest t_stressTest ]  
    [ ?stressVarList l_stressVarList ]  
    [ ?agedVarList l_agedVarList ]  
    [ ?freshTestEnabled g_freshTestEnabled ]  
    [ ?stressTestEnabled g_stressTestEnabled ]  
    [ ?agedTestEnabled g_agedTestEnabled ]  
    [ ?stressFile t_stressFile ]  
    [ ?relxOptions l_relxOptions ]  
    [ ?relxModelSetup l_relxModelSetup ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Adds a reliability setup with the specified settings in the specified ADE Explorer or ADE Assembler session.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup to be added.
<i>t_freshTest</i>	Name of the test to be used as the fresh test.
<i>t_agedTest</i>	Name of the test to be used as the aged test.
<i>?stressTest</i> <i>t_stressTest</i>	Name of the test to be used as the stress test. Default: " "
<i>?stressVarList l_stressVarList</i>	List of name-value pairs of the variables to be used in the stress test. For example: <pre>?stressVarList ' (("vdc" "1.7") ("a" "7") ("b" "9"))</pre> If you do not specify this argument, variables and their values are used from the specified stress test.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

`?agedVarList l_agedVarList`

List of name-value pairs of the variables to be used in the aged test.

For example:

```
?agedVarList ' (("vdc" "1.9") ("a" "9") ("b" "11"))
```

If you do not specify this argument, variables and their values are used from the specified aged test.

`?freshTestEnabled g_freshTestEnabled`

Specifies whether to enable the fresh test.

Valid values:

- `t`: Fresh test is enabled.
- `nil`: Fresh test is disabled.

Default: `t`

`?stressTestEnabled g_stressTestEnabled`

Specifies whether to enable the stress test.

Valid values:

- `t`: Stress test is enabled.
- `nil`: Stress test is disabled.

Default: `t`

`?agedTestEnabled g_agedTestEnabled`

Specifies whether to enable the aged test.

Valid values:

- `t`: Aged test is enabled.
- `nil`: Aged test is disabled.

Default: `t`

`?stressFile t_stressFile`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Name of the stress file to be reused for stress simulation.

If you specify this argument, you must specify the `?stressTestEnabled` as `nil`. Otherwise, the stress file specified by the argument `?stressFile` will be ignored.

`?relxOptions` *l_relxOptions*

List of name-value pairs of the reliability options.

For example:

```
?relxOptions '(nil simulator "spectre" relx-
  Opts (((relxOpts uriLibs) "") ((relxOpts mos-
    AgingTime) "1 2") ((relxOpts gaAgePoints)
    "1,2") ((relxOpts enableSelfheating) t)
    ((relxOpts enableGradualAging) t) ((relxOpts
    enableDegSort) t) ((relxOpts agingPointsType)
    "Gradual aging"))
```

If you do not specify this argument, the default values of the reliability options are used in the reliability setup.

`?relxModelSetup` *l_relxModelSetup*

List specifying the reliability model setup.

`?run` *t_runName*

Name of the run in the run plan in which you want to add the reliability setup. If you do not specify this argument, the reliability setup is added to the active setup.

This argument is useful only when the setup has a run plan.

`?session` *t_session*

Name of the ADE Explorer or ADE Assembler session.

Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The reliability setup is added.
<code>nil</code>	The reliability setup is not added because of an error.

Examples

The following example adds a reliability setup `relA` in the active setup of the current maestro session:

```
maeAddRelxSetup("relA" "testA" "testB" ?stressTest "testC" ?stressVarList '(("vdc"
"1.7") ("a" "7") ("b" "9")) ?agedVarList '(("vdc" "1.9") ("a" "9") ("b" "11"))
?freshTestEnabled t ?stressTestEnabled t ?agedTestEnabled t ?relxOptions '(nil
simulator "spectre" relxOpts (((relxOpts uriLibs) "") ((relxOpts mosAgingTime) "1
2") ((relxOpts gaAgePoints) "1,2") ((relxOpts enableSelfheating) t) ((relxOpts
enableGradualAging) t) ((relxOpts enableDegSort) t) ((relxOpts agingPointsType)
"Gradual aging"))))
```

The following example adds a reliability setup `relB` in `Run.0` in the current maestro session:

```
maeAddRelxSetup("relB" "testA" "testA" ?stressTest "testA" ?stressVarList '(("vdc"
"1.7") ("a" "7") ("b" "9")) ?agedVarList '(("vdc" "1.9") ("a" "9") ("b" "11"))
?freshTestEnabled t ?stressTestEnabled t ?agedTestEnabled t ?run "Run.0")
```

maeCopyRelxSetup

```
maeCopyRelxSetup(  
    t_relxSourceName  
    [ ?relxDestName t_relxDestName ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Creates a copy of a reliability setup and saves this copy with the specified name.

Arguments

t_relxSourceName Name of the reliability setup to be copied.

?relxDestName t_relxDestName

Name with which you want to save the copy of the reliability setup.

If you do not specify this argument, the copy of the reliability setup is saved with the name *relxSourceName:1*.

?run t_runName

Name of the run in the run plan in which you want to copy the reliability setup.

Specify this argument only when you want to create a copy of the reliability setup that is saved in a specific run of a run plan.

The reliability setup to be copied must exist in the specified run.

?session t_session

Name of the ADE Explorer or ADE Assembler session.

Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The reliability setup is copied.
<code>nil</code>	The reliability setup is not copied because of an error.

Examples

The following example creates a copy of reliability setup `relA` and saves this copy with the name `relB` in the active setup of the current maestro session.

```
maeCopyRelxSetup("relA" ?relxDestName "relB")  
=> t
```

The following example creates a copy of reliability setup `relA` and saves this copy in the active setup of the current maestro session. The copy of the reliability setup `relA` is saved with the name `relA:1`.

```
maeCopyRelxSetup("relA")  
=> t
```

The following example creates a copy of reliability setup `aging_10Years` and saves this copy with the name `aging_5Years` in `Run.0` in the current maestro session.

```
maeCopyRelxSetup("aging_10Years" ?relxDestName "aging_5Years" ?run "Run.0")  
=> t
```

Note that the reliability setup `aging_10Years` must exist in `Run.0`.

maeDeleteRelxSetup

```
maeDeleteRelxSetup(  
    t_relxSetupName  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Deletes a reliability setup from the specified ADE Explorer or ADE Assembler session.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup to be deleted.
<i>?run t_runName</i>	Name of the run in a run plan from which you want to delete the reliability setup. If you do not specify this argument, the reliability setup is deleted from the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>t</i>	The reliability setup is deleted.
<i>nil</i>	The reliability setup is not deleted because of an error.

Examples

The following example deletes the reliability setup `relA` from the active setup of the current maestro session.

```
maeDeleteRelxSetup("relA")  
=> t
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The following example deletes the reliability setup `aging_10Years` from `Run.0` in the current maestro session.

```
maeDeleteRelxSetup("aging_10Years" ?run "Run.0")  
=> t
```

maeCreateNetlistForRel

```
maeCreateNetlistForRel(  
    t_relxSetupName  
    [ ?stage t_stage ]  
    [ ?cornerName t_cornerName ]  
    [ ?netlistDir t_netlistDir ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Creates a netlist for the specified reliability setup. This function supports only *Single Run*, *Sweeps* and *Corners* run mode.

Arguments

<code>t_relxSetupName</code>	Name of the reliability setup for which the netlist is to be created.
<code>?stage t_stage</code>	Stage of the reliability analysis. Valid values: <code>fresh</code> , <code>stress</code> , and <code>age</code> . Default: <code>fresh</code>
<code>?cornerName t_cornerName</code>	Name of the corner. Default: <code>Nominal</code>
<code>?netlistDir t_netlistDir</code>	Name of the directory where the netlist is to be created. Default: The current working directory
<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the netlist is created for the reliability setup that is saved in the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Name of the ADE Explorer or ADE Assembler session.

Default: The current session.

Value Returned

<code>t</code>	The netlist for the reliability setup is created.
<code>nil</code>	The netlist for the reliability setup is not created because of an error.

Examples

The following example creates a netlist for the `stress` stage of the reliability setup `SP_AGING` saved in the active setup of the current maestro session.

```
maeCreateNetlistForRel("SP_AGING" ?stage "stress")  
=> t
```

The following example creates a netlist for the `stress` stage of the reliability setup `SP_AGING` saved in `Run.0` in the current maestro session.

```
maeCreateNetlistForRel("SP_AGING" ?stage "stress" ?run "Run.0")  
=> t
```

maeExportScenarioSetup

```
maeExportScenarioSetup(  
    t_reliabilityScenarioSetup  
    [ ?session t_sessionName ]  
    [ ?run t_runName ]  
    [ ?file t_fileName ]  
    [ ?allReadOnly g_allReadOnly ]  
    [ ?enabledReadOnly g_enabledReadOnly ]  
)  
=> t / nil
```

Description

Exports the details of the specified reliability scenario setup to a CSV file.

Arguments

t_reliabilityScenarioSetup

Name of the reliability scenario setup whose details are to be exported to the CSV file.

?session t_sessionName

Name of the session.

If you do not specify this argument, the current session is used.

To get a list of available maestro sessions, use the function `maeGetSessions`.

?run t_runName

Name of the run in a run plan. The default value is an empty string (`" "`).

This argument is useful only when the setup has a run plan.

?file t_fileName

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Name of the CSV file to which you want to export the details of reliability scenario setup.

If you do not specify this argument, details of the reliability scenario setup are exported to the `scenarioSetup.csv` file in the current working directory.

You can specify either the absolute path or the relative path to the CSV file in this argument. For example:

- `"/home/user/ReliabilitySetup/test.csv"`
- `"./test.csv"`

`?allReadOnly` *g_allReadOnly*

Specifies whether to export all reliability scenario rows from the setup in read-only mode. Valid values are `t` and `nil`.

The default value is `nil`.

The arguments `?allReadOnly` and `?enabledReadOnly` are mutually exclusive. You cannot specify the value `t` for both the arguments, simultaneously.

`?enabledReadOnly` *g_enabledReadOnly*

Specifies whether to export only enabled reliability scenario rows in read-only mode. Valid values are `t` and `nil`.

The default value is `nil`.

The arguments `?allReadOnly` and `?enabledReadOnly` are mutually exclusive. You cannot specify the value `t` simultaneously for both the arguments.

Value Returned

`t` The details of the specified reliability scenario are successfully exported.

`nil` The details of the specified reliability scenario cannot be exported because of an error.

Examples

The following example exports the details of the reliability scenario setup `RelxScenario` to the `testCSV.csv` file at the specified location.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
maeExportScenarioSetup("RelxScenario" ?file "/home/user/ReliabilitySetup/testCSV.csv")
```

```
=> RelxScenario is exported to /home/user/ReliabilitySetup/testCSV.csv
```

```
=> t
```

The following example exports the details of the reliability scenario setup `RelxScenario` to the `scenarioSetup.csv` file in the current working directory.

```
maeExportScenarioSetup("RelxScenario")
```

```
=> RelxScenario is exported to ./scenarioSetup.csv
```

```
=> t
```

The following example exports the details of the reliability scenario setup `RelxScenarios` to the specified CSV file in the current working directory. All the reliability scenario rows from the Reliability Scenarios Setup form are exported in read-only mode to the CSV file.

```
maeExportScenarioSetup("RelxScenarios" ?file "./myExportedSetup" ?allReadOnly t)
```

```
=> RelxScenarios0 is exported as ./myExportedSetup
```

```
=> t
```

The following example exports the details of the reliability scenario setup `RelxScenarios` to the specified CSV file in the current working directory. Only the reliability scenario rows that are enabled in the Reliability Scenarios Setup form are exported in read-only mode to the CSV file.

```
maeExportScenarioSetup("RelxScenarios" ?file "./myExportedSetupReadOnly" ?enabledReadOnly t)
```

```
=> RelxScenarios0 is exported as ./myExportedSetupReadOnly
```

```
=> t
```

maeGetReliabilityAnalysisEnabled

```
maeGetReliabilityAnalysisEnabled(  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Returns the current status of the *Reliability Analyses* node in the active setup or in the specific run of a run plan.

Arguments

<code>?run t_runName</code>	Name of the run in a run plan in which you want to check the status of the <i>Reliability Analyses</i> node. If you do not specify this argument the status of the <i>Reliability Analyses</i> node is checked in the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<code>t</code>	The <i>Reliability Analyses</i> node is enabled.
<code>nil</code>	The <i>Reliability Analyses</i> node is disabled.

Examples

The following example indicates that the *Reliability Analyses* node is enabled in the active setup of the current maestro session.

```
maeGetReliabilityAnalysisEnabled()  
=> t
```

The following example indicates that the *Reliability Analyses* node is disabled in Run . 0 in the current maestro session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
maeGetReliabilityAnalysisEnabled(?run "Run.0")  
=> nil
```

maeGetReliabilityModel

```
maeGetReliabilityModel(  
    t_relxSetupName  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> l_modelFile / nil
```

Description

Returns the model file settings used for the specified reliability setup in the active setup or in the specific run of a run plan.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup for which the model file settings are to be retrieved.
<i>?run t_runName</i>	Name of the run in a run plan for which the model file settings are to be retrieved. If you do not specify this argument, the model file settings for the reliability setup are retrieved from the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>l_modelFile</i>	Model file settings used in the reliability setup.
<i>nil</i>	Model file settings are not retrieved because of an error.

Examples

The following example returns the model file settings used in the reliability setup `relA`, which is saved in the active setup of the current maestro session.

```
maeGetReliabilityModel("relA")
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
=> (nil placementtype "After Standard Model" modelfiles "((\\servers/user/Reliability/model_spectre/age.scs\\" \"tt\\\"))")
```

The following example returns the model file settings used in the reliability setup `relA`, which is saved in `Run.0` in the current `maestro` session.

```
maeGetReliabilityModel("relA" ?run "Run.0")  
=> (nil placementtype "Before Standard Model" modelfiles "((\\servers/user/Reliability/model_spectre/new.scs\\" \"tt\\\"))")
```

maeGetReliabilityOptionVal

```
maeGetReliabilityOptionVal(  
    t_relxSetupName  
    s_varName  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t_varValue / nil
```

Description

Returns the value of the specified reliability option in a reliability setup.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>s_varName</i>	Name of the reliability option. For more details about the reliability options, refer to spec-tre.relxOpts environment variables.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the value of the specified reliability option is retrieved from the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<code>t_varValue</code>	The value of the reliability option in the specified reliability setup.
<code>nil</code>	The value of the reliability option is not retrieved because of an error.

Examples

The following example returns the value of the reliability option `agingPointsType` in the reliability setup `relA` from the active setup.

```
maeGetReliabilityOptionVal("relA" 'agingPointsType)
=> "Standard aging"
```

The following example returns the value of the `degSortType` in the reliability setup `relA` from the active setup.

```
maeGetReliabilityOptionVal("relA" 'degSortType)
=> "threshold"
```

The following example returns the value of the reliability option `agingPointsType` in the reliability setup `relA` from `Run.0`.

```
maeGetReliabilityOptionVal("relA" 'agingPointsType ?run "Run.0")
=> "Gradual aging"
```

The following example returns the value of the `degSortType` in the reliability setup `relA` from `Run.0`.

```
maeGetReliabilityOptionVal("relA" 'degSortType ?run "Run.0")
=> "number"
```


maeGetReliabilityVarValue

```
maeGetReliabilityVarValue(  
    t_relxSetupName  
    t_varName  
    [ ?varType t_varType ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t_varValue / nil
```

Description

Returns the value of the specified variable used in the fresh, stress, or aged test in a reliability setup.

Arguments

<code>t_relxSetupName</code>	Name of the reliability setup.
<code>t_varName</code>	Name of the variable whose value is to be retrieved.
<code>?varType t_varType</code>	Name of the reliability stage. Valid values: <code>fresh</code>, <code>stress</code>, and <code>aged</code>. Default: <code>fresh</code>
<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the value of the specified variable is retrieved from the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t_varValue</code>	The value of the variable in the reliability setup.
<code>nil</code>	The value of the variable is not retrieved because of an error.

Examples

The following example returns the value of the variable `vdc` in the `stress` stage of the reliability setup `relA` saved in the active setup of the current maestro session.

```
maeGetReliabilityVarValue("relA" "vdc" ?varType "stress")  
=> "1.5"
```

The following example returns the value of the variable `vdc` in the `stress` stage of the reliability setup `relA` saved in `Run.0` in the current maestro session.

```
maeGetReliabilityVarValue("relA" "vdc" ?varType "stress" ?run "Run.0")  
=> "1.7"
```

maeGetRelxSetups

```
maeGetRelxSetups (
    [ ?run t_runName ]
    [ ?session t_session ]
    [ ?enabled g_enabled ]
)
=> l_relxSetups / nil
```

Description

Returns a list of reliability setups saved in the active setup or in the specific run of a run plan.

Arguments

<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the reliability setups are returned from the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.
<code>?enabled g_enabled</code>	Status of the reliability setup. Possible values: ■ <code>t</code>: Returns only those reliability setups which are enabled. ■ <code>nil</code>: Returns only those reliability setups which are disabled. If you do not specify this argument, all the reliability setups are returned, irrespective of their statuses.

Value Returned

<i>l_relxSetups</i>	A list of reliability setups.
<i>nil</i>	The reliability setups cannot be retrieved because of an error.

Examples

The following example returns all the reliability setups saved in the active setup of the current maestro session.

```
maeGetRelxSetups()  
=> ("relA" "relB")
```

The following example returns the reliability setups which are enabled in the active setup of the current maestro session.

```
maeGetRelxSetups(?enabled t)  
=> ("relB")
```

The following example returns the reliability setups which are disabled in the active setup of the current maestro session.

```
maeGetRelxSetups(?enabled nil)  
=> ("relA")
```

The following example returns all the reliability setups that are saved in `Run.0` in the current maestro session.

```
maeGetRelxSetups(?run "Run.0")  
=> ("Aging_10yr" "Aging_5yr")
```

maeGetRelxSetupEnabled

```
maeGetRelxSetupEnabled(  
    t_relxSetupName  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Returns the current status of a reliability setup in the active setup or in the specific run of a run plan.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the status of the reliability setup is retrieved from the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>t</i>	The reliability setup is enabled.
<i>nil</i>	The reliability setup is disabled.

Examples

The following example indicates that the reliability setup `relA` is enabled in the active setup of the current maestro session.

```
maeGetRelxSetupEnabled("relA")  
=> t
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The following example indicates that reliability setup `relA` is disabled in `Run.0` in the current maestro session.

```
maeGetRelxSetupEnabled("relA" ?run "Run.0")  
  
=> nil
```

maeGetRelxStressFile

```
maeGetRelxStressFile(  
    t_relxSetupName  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t_stressFile / nil
```

Description

Returns the stress file used in the specified reliability setup in the active setup or in the specific run of a run plan.

Arguments

<code>t_relxSetupName</code>	Name of the reliability setup.
<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the reliability setup in the active setup is used to retrieve the stress file. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<code>t_stressFile</code>	Name of the stress file.
<code>nil</code>	The stress file cannot be retrieved because of an error.

Examples

The following example returns the stress file being used in the reliability setup `relA` in the active setup of the current maestro session.

```
maeGetRelxStressFile("relA")  
=> "/servers/user/reliability/simulation/library/cell/maestro/results/maestro/  
Interactive.6/2/testA/netlist/input.bs0"
```

maeGetRelxTest

```
maeGetRelxTest(  
    t_relxSetupName  
    [ ?testType t_testType ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t_testName / nil
```

Description

Returns the name of the test used in the fresh, stress, or aged stage of a reliability setup.

Arguments

<code>t_relxSetupName</code>	Name of the reliability setup.
<code>?testType t_testType</code>	Stage of the reliability setup. Valid values: <code>fresh</code> , <code>stress</code> , <code>aged</code> . Default: <code>fresh</code>
<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the test name is retrieved from the reliability setup in the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<i>t_testName</i>	The test name used in the specified stage of the reliability setup.
<i>nil</i>	The test name is not retrieved because of an error.

Examples

The following example returns the name of the test used in the `fresh` stage of the reliability setup `relA` saved in the active setup of the current maestro session.

```
maeGetRelxTest("relA")  
=> "testA"
```

The following example returns the name of the test used in the `stress` stage of the reliability setup `relA` saved in the active setup of the current maestro session.

```
maeGetRelxTest("relA" ?testType "stress" )  
=> "testB"
```

The following example returns the name of the test used in the `stress` stage of the reliability setup `relA` saved in `Run.0` in current maestro session.

```
maeGetRelxTest("relA" ?testType "stress" ?run "Run.0")  
=> "testC"
```

maeGetRelxTestEnabled

```
maeGetRelxTestEnabled(  
    t_relxSetupName  
    [ ?testType t_testType ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Returns the current status of a reliability stage in the specified reliability setup.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>?testType t_testType</i>	Stage of the reliability setup. Valid values: <i>fresh</i> , <i>stress</i> , <i>aged</i> . Default value: <i>fresh</i>
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the status of the reliability stage is retrieved from the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The reliability stage is enabled in the reliability setup.
<code>nil</code>	The reliability stage is disabled in the reliability setup.

Examples

The following example indicates that the reliability stage `fresh` is enabled in the reliability setup `relA` saved in the active setup of the current maestro session.

```
maeGetRelxTestEnabled("relA")  
=> t
```

The following example indicates that the reliability stage `stress` is disabled in the reliability setup `relA` saved in the active setup of the current maestro session.

```
maeGetRelxTestEnabled("relB" ?testType "stress")  
=> nil
```

The following example indicates that the reliability stage `fresh` is disabled in the reliability setup `relA` saved in `Run.0` in the current maestro session.

```
maeGetRelxTestEnabled("relA" ?testType "fresh" ?run "Run.0")  
=> nil
```

The following example indicates that the reliability stage `stress` is enabled in the reliability setup `relA` saved in `Run.0` in the current maestro session.

```
maeGetRelxTestEnabled("relA" ?testType "stress" ?run "Run.0")  
=> t
```

maeImportScenarioSetup

```
maeImportScenarioSetup(  
    t_reliabilityScenarioSetup  
    [ ?session t_sessionName ]  
    [ ?run t_runName ]  
    [ ?file t_fileName ]  
    [ ?operation t_typeOfOperation ]  
)  
=> t / nil
```

Description

Imports the details from the specified CSV file to the existing reliability scenario setup.

Arguments

t_reliabilityScenarioSetup

Name of the reliability scenario setup to which you want to import details from the CSV file.

If the specified reliability scenario setup does not exist in the *Reliability Scenarios* list in the Data View assistant, an error message is displayed in the CIW.

?session t_sessionName

Name of the session.

If you do not specify this argument, the current session is used.

To get a list of available maestro sessions, use the `maeGetSessions`.

?run t_runName

Name of the run in a run plan. The default value is an empty string ("").

This argument is useful only when the setup has a run plan.

?file t_fileName

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Name of the CSV file from which you want to import the details to the existing reliability scenario setup.

If you do not specify this argument, details of the reliability scenario setup are imported from the `scenarioSetup.csv` file saved in the current working directory.

If the file `scenarioSetup.csv` does not exist in the current working directory, an error message is displayed in the CIW.

You can specify either the absolute path or the relative path to the CSV file in this argument. For example:

- `"/home/user/ReliabilitySetup/test.csv"`
- `"/test.csv"`

`?operation t_operationType`

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Type of operation to be performed.

Valid values are:

- **overwrite**: Overwrites the details from the CSV file in the existing reliability scenario setup. This is the default value.

All existing scenarios from the Reliability Scenarios Setup form are deleted and from the CSV file are added to the form.

- **merge**: Merges the details from the CSV file in the existing reliability scenario setup.

If a scenario name being imported from the CSV file already exists in the Reliability Scenarios Setup form, the existing scenario is deleted from the form, and its details are copied from the CSV to the form.

When you specify this value:

- ☐ Scenarios that exist only in the form, but not in CSV file, are kept as is.
- ☐ Scenarios that exist only in the CSV file, but not in the form, are appended in the form.
- **retain**: If a scenario name being imported from the CSV file already exists in the Reliability Scenarios Setup form, its import is discarded from the CSV file, and the existing scenario in the UI is retained as is.

When you specify this value:

- ☐ Scenarios that exist only in the Reliability Scenarios Setup form, but not in the CSV file being imported, are kept as is.
- ☐ Scenarios that exist only in the CSV file, but not in the form, are appended.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The details from the CSV file are successfully imported to the reliability scenario setup.
<code>nil</code>	The details from the CSV file cannot be imported to the reliability scenario setup because of an error.

Examples

The following example imports the details of the reliability scenario setup from the `testCSV.csv` file to the existing reliability scenario setup `RelxScenario`.

```
maeImportScenarioSetup("RelxScenario" ?file "/home/user/ReliabilitySetup/  
testCSV.csv")
```

```
=> t
```

The following example merges the details of the reliability scenario setup from the `scenarioSetup.csv` file saved in the current working directory to the existing reliability scenario setup `RelxScenario`.

```
maeImportScenarioSetup("RelxScenario" ?operation "merge")
```

```
=> t
```

maeRenameRelxSetup

```
maeRenameRelxSetup(  
    t_relxSourceName  
    t_relxDestName  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Renames a reliability setup with the specified name.

Arguments

<i>t_relxSourceName</i>	Name of the reliability setup to be renamed.
<i>t_relxDestName</i>	New name of the reliability setup.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the reliability setup is renamed in the active setup. This argument is useful when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>t</i>	The reliability setup is renamed.
<i>nil</i>	The reliability setup is not renamed because of an error.

Examples

The following example renames the reliability setup `AGING` to `NEW_AGING` in the active setup of the current maestro session.

```
maeRenameRelxSetup("AGING" "NEW_AGING")  
=> t
```


Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The following example renames the reliability setup `AGING` to `NEW_AGING` in `Run.0` in the current maestro session.

```
maeRenameRelxSetup("AGING" "NEW_AGING" ?run "Run.0")  
=> t
```

maeSetReliabilityAnalysisEnabled

```
maeSetReliabilityAnalysisEnabled(  
    g_enabled  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Sets the status of *Reliability Analyses* node to the specified value.

Arguments

<i>g_enabled</i>	The status of the <i>Reliability Analyses</i> node to be set. Valid values: ■ <i>t</i>: Enables the <i>Reliability Analyses</i> node. ■ <i>nil</i>: Disables the <i>Reliability Analyses</i> node.
<i>?run t_runName</i>	Name of the run in a run plan for which you want to set the status of the <i>Reliability Analyses</i> node. If you do not specify this argument, the status of the Reliability Analyses is set in the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>t</i>	The status of the <i>Reliability Analyses</i> node is set.
<i>nil</i>	The status of the <i>Reliability Analyses</i> node is not set because of an error.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Examples

The following example enables the *Reliability Analyses* node in the active setup of the current maestro session.

```
maeSetReliabilityAnalysisEnabled(t)
=> t
```

The following example disables the *Reliability Analyses* node in `Run.0` in the current maestro session.

```
maeGetReliabilityAnalysisEnabled(nil ?run "Run.0")
=> t
```

maeSetReliabilityModel

```
maeSetReliabilityModel(  
    t_relxSetupName  
    l_value  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Sets the model files to be used in the specified reliability setup.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>l_value</i>	The model file settings to be used in the reliability setup.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the model file settings are applied to the reliability setup in the active setup. This argument is useful when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>t</i>	The model file settings are applied to the reliability setup.
<i>nil</i>	The model file settings are not applied to the reliability setup because of an error.

Examples

The following example applies the model file settings to the reliability setup `relA` in the active setup of the current maestro session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
maeSetReliabilityModel("relA" '(nil placementtype "After Standard Model"
modelfiles "((\"/servers/user/reliability/model_spectre/mos.scs\" \"AGE_ON\")))")
=> t
```

The following example applies the model file settings to the reliability setup `relA` in `Run.0` in the current maestro session.

```
maeSetReliabilityModel("relA" '(nil placementtype "After Standard Model"
modelfiles "((\"/servers/user/reliability/model_spectre/mos.scs\" \"AGE_ON\")))")
?run "Run.0")
=> t
```

maeSetReliabilityOptionVal

```
maeSetReliabilityOptionVal(  
    t_relxSetupName  
    s_varName  
    t_varValue  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Sets the value of the specified reliability option in a reliability setup.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>s_varName</i>	Name of the reliability option. For more details about the reliability options, refer to spec-tre.relxOpts environment variables.
<i>t_varValue</i>	Value of the reliability option to be set.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the value of the reliability option is set in the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The value of the reliability option is set.
<code>nil</code>	The value of the reliability option is not set because of an error.

Examples

The following example sets the value of the reliability option `agingPointsType` to `Gradual aging` in the reliability setup `relA` in the active setup of the current maestro session.

```
maeSetReliabilityOptionVal("relA" 'agingPointsType "Gradual aging")  
=> t
```

The following example sets the value of the reliability option `degSortType` to `threshold` in the reliability setup `relA` saved in `Run.0` in the current maestro session.

```
maeSetReliabilityOptionVal("relA" 'degSortType "threshold" ?run "Run.0")  
=> t
```

maeSetReliabilityVarValue

```
maeSetReliabilityVarValue(  
    t_relxSetupName  
    t_varName  
    t_varValue  
    [ ?varType t_varType ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Sets the value of the specified variable used in the `fresh`, `stress`, or `aged` test in a reliability setup.

Arguments

<code>t_relxSetupName</code>	Name of the reliability setup.
<code>t_varName</code>	Name of the variable.
<code>t_varValue</code>	Value of the variable to be set.
<code>?varType t_varType</code>	Name of the reliability stage. Valid values: <code>fresh</code>, <code>stress</code>, and <code>aged</code>. Default: <code>fresh</code>
<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the value of the variable is set in the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The value of the variable is set.
<code>nil</code>	The value of the variable is not set because of an error.

Examples

The following example sets the value of the variable `vdc` to `1.7` in the `stress` stage of the reliability setup `relA` in the active setup of the current maestro session.

```
maeSetReliabilityVarValue("relA" "vdc" 1.7 ?varType "stress")
=> t
```

The following example sets the value of the variable `vdc` to `1.9` in the `stress` stage of the reliability setup `relA` saved in `Run.0` in the current maestro session.

```
maeSetReliabilityVarValue("relA" "vdc" 1.9 ?varType "stress" ?run "Run.0")
=> t
```

maeSetRelxSetupEnabled

```
maeSetRelxSetupEnabled(  
    t_relxSetupName  
    g_enabled  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Enables or disables a reliability setup in the active setup or in the specific run of a run plan.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>g_enabled</i>	Status of the reliability setup. Valid values: t: Enables the reliability setup. nil: Disables the reliability setup. Default: t
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the specified reliability setup is enabled or disabled in the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The status of the reliability setup is set.
<code>nil</code>	The status of the reliability setup is not set because of an error.

Examples

The following example enables the reliability setup `relA` in the active setup of the current maestro session.

```
maeSetRelxSetupEnabled("relA" t)
=> t
```

The following example disables the reliability setup `relA` in `Run.0` in the current maestro session.

```
maeSetRelxSetupEnabled("relA" nil ?run "Run.0")
=> t
```

maeSetRelxStressFile

```
maeSetRelxStressFile(  
    t_relxSetupName  
    t_stressFile  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Sets the stress file to be used in the specified reliability setup in the active setup or in the specific run of a run plan.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>t_stressFile</i>	Name and path to the stress file.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the reliability setup from the active setup is used. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Value Returned

<i>t</i>	Stress file is set successfully.
<i>nil</i>	The stress file cannot be set because of an error.

Examples

The following example sets the stress file `input.bs0` to the reliability setup `relA` in the active setup of the current maestro session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
maeGetRelxStressFile("relA" "/servers/user/reliability/simulation/library/cell/  
maestro/results/maestro/Interactive.6/2/testA/netlist/input.bs0")
```

```
=> t
```

maeSetRelxTest

```
maeSetRelxTest(  
    t_relxSetupName  
    t_testName  
    [ ?testType t_testType ]  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Sets the name of the `fresh`, `stress`, or `aged` test used in the specified reliability setup.

Arguments

<code>t_relxSetupName</code>	Name of the reliability setup.
<code>t_testName</code>	Name of the test to be set.
<code>?testType t_testType</code>	Stage of the reliability setup. Valid values: <code>fresh</code> , <code>stress</code> , <code>aged</code> . Default value: <code>fresh</code>
<code>?run t_runName</code>	Name of the run in a run plan. If you do not specify this argument, the test name is set in the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<code>?session t_session</code>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The test name of the reliability stage is set.
<code>nil</code>	The test name of the reliability stage is not set because of an error.

Examples

The following example sets the name of the test used in the `stress` stage of the reliability setup `relA` to `osc_Stress` in the active setup of the current maestro session.

```
maeSetRelxTest("relA" "osc_Stress" ?testType "stress")  
=> t
```

The following example sets the name of the test used in the `stress` stage of the reliability setup `relA` to `new_Stress` in `Run.0` in the current maestro session.

```
maeSetRelxTest("relA" "new_Stress" ?testType "stress" ?run "Run.0")  
=> t
```

maeSetRelxTestEnabled

```
maeSetRelxTestEnabled(  
    t_relxSetupName  
    g_enabled  
    [ ?run t_runName ]  
    [ ?session t_session ]  
)  
=> t / nil
```

Description

Enables or disables a reliability stage in the specified reliability setup.

Arguments

<i>t_relxSetupName</i>	Name of the reliability setup.
<i>g_enabled</i>	Status of the reliability stage. Valid values: ■ <i>t</i>: Enables the reliability stage. ■ <i>nil</i>: Disables the reliability stage.
<i>?run t_runName</i>	Name of the run in a run plan. If you do not specify this argument, the reliability stage is disabled from the reliability setup saved in the active setup. This argument is useful only when the setup has a run plan.
<i>?session t_session</i>	Name of the ADE Explorer or ADE Assembler session. Default: The current session.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The status of the reliability stage is set.
<code>nil</code>	The status of the reliability stage is not set because of an error.

Examples

The following example enables the reliability stage `aged` in the reliability setup `relA` saved in the active setup of the current maestro session.

```
maeSetRelxTestEnabled("relA" t ?testType "aged")  
=> t
```

The following example disables the reliability stage `aged` in the reliability setup `relA` saved in `Run.0` in the current maestro session.

```
maeSetRelxTestEnabled("relA" nil ?testType "aged" ?run "Run.0")  
=> t
```

relxEnableFormTab

```
relxEnableFormTab(  
    r_formObject  
    t_tabName  
    g_enableBoolean  
)  
=> t / nil
```

Description

Enables or disables the specified tab in the Reliability Options form.

Arguments

<i>r_formObject</i>	The form object: <code>relxOptionForm</code> .
<i>t_tabName</i>	Name of the tab to be enabled or disabled. Possible values are: ■ Basic ■ Modeling ■ Degradation ■ Output
<i>g_enableBoolean</i>	Specifies whether to enable or disable the specified tab. Possible values are: ■ <code>t</code>: Enables the specified tab. ■ <code>nil</code>: Disables the specified tab.

Value Returned

<code>t</code>	The tab is enabled.
<code>nil</code>	Returns <code>nil</code> the tab is disabled.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Examples

To disable the *Basic* and *Output* tabs in the Reliability Options form, add the following SKILL code in the `.cdsinit` file:

```
(defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)
  relxEnableFormTab(relxOptionForm "Basic" nil)
  relxEnableFormTab(relxOptionForm "Output" nil)
)
```

Reliability Options

BasicModelingDegradationOutput

▼ Unified Reliability Interface (URI) Models

URI Libraries

Age Calculation Approach

☐ URI Debug Messaging

Scale Param Method

▼ Age Modeling

Age Modeling Effect

Aging Effect Contributors

AGEMOS

☒ Hot-Carrier Injection (HCI)☒ Negative Bias Temperature Instability (NBTI)☒ Positive Bias Temperature Instability (PBTI)

URI ModelsAge LevelsAll

Minimum Age

▼ Hot-Carrier Injection Modeling

Accuracy

Drain Current Calculation Method

Vth, Igate, and Isub Calculation Method

Liberal

Ids Static Current

From Model Parameters

OKCancelDefaultsApplyHelp

relxDisplayDiscField

```
relxDisplayDiscField(  
    r_formObject  
    t_disclosureName  
    g_displayableBoolean  
)  
=> t / nil
```

Description

Shows or hides the specified disclosure in the Reliability Options form.

Arguments

<i>r_formObject</i>	The form object: <code>relxOptionForm</code> .
<i>t_disclosureName</i>	Name of the disclosure to be shown or hidden. For example, <code>Age Modeling</code>
<i>g_displayableBoolean</i>	Specifies whether to show or hide the specified disclosure. Possible values are: ■ <code>t</code>: Shows the disclosure. ■ <code>nil</code>: Hides the disclosure.

Value Returned

<code>t</code>	The disclosure is shown.
<code>nil</code>	Returns <code>nil</code> the disclosure is hidden.

Examples

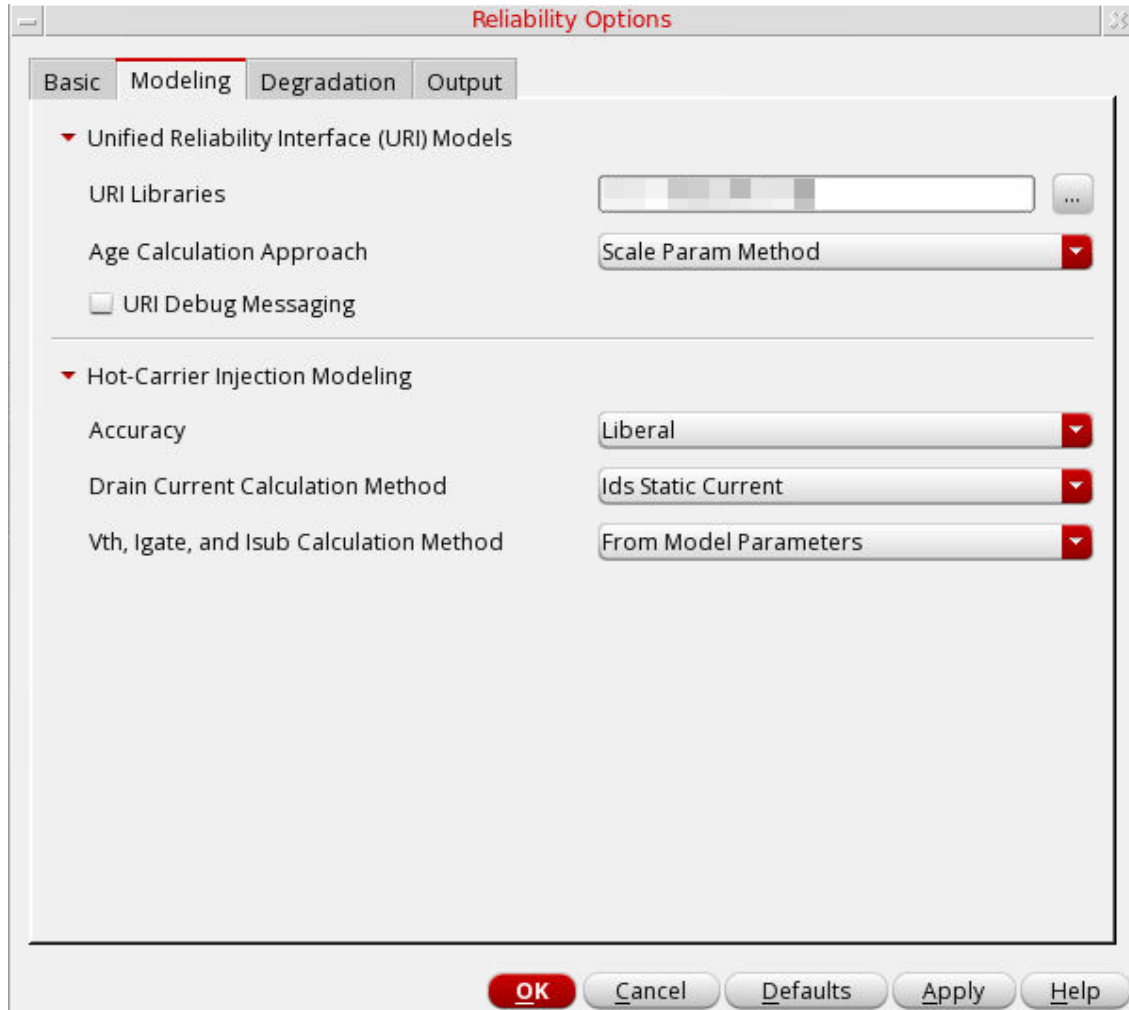
To hide the *Age Modeling* disclosure from the *Modeling* tab on the Reliability Options form, add the following SKILL code in the `.cdsinit` file:

```
(defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)  
    relxDisplayDiscField(relxOptionForm "Age Modeling" nil)  
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The *Age Modeling* disclosure is hidden from the *Modeling* tab on the Reliability Options form:



To hide the *Hot-Carrier Injection Modeling* disclosure and to show the *Age Modeling* disclosure in the *Modeling* tab in the Reliability Options form, add the following SKILL code in the `.cdsinit` file:

```
(defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)
  relxDisplayDiscField(relxOptionForm "Hot-Carrier Injection Modeling" nil)
  relxDisplayDiscField(relxOptionForm "Age Modeling" t)
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The *Hot-Carrier Injection Modeling* disclosure is hidden and the *Age Modeling* disclosure is shown on the *Modeling* tab in the Reliability Options form:

The screenshot shows the 'Reliability Options' dialog box with the 'Modeling' tab selected. The 'Age Modeling' section is highlighted with a red box. The 'Aging Effect Contributors' section is also highlighted with a red box. The 'URI Models' section is visible but not highlighted.

Reliability Options

Basic **Modeling** Degradation Output

▼ Unified Reliability Interface (URI) Models

URI Libraries [] ...

Age Calculation Approach Scale Param Method ▼

☐ URI Debug Messaging

▼ Age Modeling

Age Modeling Effect Include ▼

Aging Effect Contributors

AGEMOS ☒ Hot-Carrier Injection (HCI) ☒ Negative Bias Temperature Instability (NBTI)

☒ Positive Bias Temperature Instability (PBTI)

URI Models Age Levels All

Minimum Age []

OK Cancel Defaults Apply Help

relxEnableDiscField

```
relxEnableDiscField(  
    r_formObject  
    t_disclosureName  
    g_enableBoolean  
)  
=> t / nil
```

Description

Enables or disables the specified disclosure in the Reliability Options form.

Arguments

<i>r_formObject</i>	The form object: <code>relxOptionForm</code> .
<i>t_disclosureName</i>	Name of the disclosure to be enabled or disabled. For example, <code>Age Modeling</code>
<i>g_enableBoolean</i>	Specifies whether to enable or disable the specified disclosure. Possible values are: ■ <code>t</code>: Enables the disclosure. ■ <code>nil</code>: Disables the disclosure.

Value Returned

<code>t</code>	The disclosure is enabled.
<code>nil</code>	The disclosure is disabled.

Examples

To disable the *Hot-Carrier Injection Modeling* disclosure on the *Modeling* tab in the Reliability Options form, add the following SKILL code in the `.cdsinit` file:

```
(defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)  
    relxEnableDiscField(relxOptionForm "Hot-Carrier Injection Modeling" nil)  
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The *Hot-Carrier Injection Modeling* disclosure is disabled on the *Modeling* tab in the Reliability Options form:

The screenshot shows the 'Reliability Options' dialog box with the 'Modeling' tab selected. The 'Unified Reliability Interface (URI) Models' section includes 'URI Libraries' (set to /home/liuwei/lib.so), 'Age Calculation Approach' (Scale Param Method), and 'URI Debug Messaging' (unchecked). The 'Age Modeling' section has 'Age Modeling Effect' set to 'Include'. Under 'Aging Effect Contributors', 'Hot-Carrier Injection (HCI)', 'Positive Bias Temperature Instability (PBTI)', and 'Negative Bias Temperature Instability (NBTI)' are all checked. 'URI Models' is set to 'All' and 'Age Levels' is 'All'. The 'Minimum Age' field is empty. The 'Hot-Carrier Injection Modeling' section, which is highlighted with a red box, contains 'Accuracy' (Liberal), 'Drain Current Calculation Method' (Ids Static Current), and 'Vth, Igate, and Isub Calculation Method' (From Model Parameters). At the bottom are 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help' buttons.

To enable the *Hot-Carrier Injection Modeling* disclosure and to disable the *Age Modeling* disclosure on the *Modeling* tab in the Reliability Options form, add the following SKILL code in the `.cdsinit` file:

```
((defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)
  relxEnableDiscField(relxOptionForm "Hot-Carrier Injection Modeling" t)
  relxEnableDiscField(relxOptionForm "Age Modeling" nil)
)
```


Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The *Hot-Carrier Injection Modeling* disclosure is enabled and the *Age Modeling* disclosure is disabled on the *Modeling* tab in the Reliability Options form:

The screenshot shows the 'Reliability Options' dialog box with the 'Modeling' tab selected. The dialog has four tabs: 'Basic', 'Modeling', 'Degradation', and 'Output'. The 'Modeling' tab contains the following settings:

- Unified Reliability Interface (URI) Models**
 - URI Libraries: [Empty text box]
 - Age Calculation Approach: Scale Param Method
 - ☐ URI Debug Messaging
- Age Modeling**
 - Age Modeling Effect: Include
 - Aging Effect Contributors**
 - AGEMOS
 - ☒ Hot-Carrier Injection (HCI)
 - ☒ Negative Bias Temperature Instability (NBTI)
 - ☒ Positive Bias Temperature Instability (PBTI)
 - URI Models: [Empty text box]
 - Age Levels: All
 - Minimum Age: [Empty text box]
- Hot-Carrier Injection Modeling**
 - Accuracy: Liberal
 - Drain Current Calculation Method: Ids Static Current
 - Vth, Igate, and Isub Calculation Method: From Model Parameters

At the bottom of the dialog are buttons for 'OK', 'Cancel', 'Defaults', 'Apply', and 'Help'.

relxHideAgeCalculationApproachField

```
relxHideAgeCalculationApproachField(  
    r_formObject  
)  
=> t
```

Description

Hides the *Age Calculation Approach* field from the *Modeling* tab in the Reliability Options form.

Arguments

<i>r_formObject</i>	The form object: relxOptionForm.

Value Returned

t	The <i>Age Calculation Approach</i> field is hidden.

Examples

To hide the *Age Calculation Approach* field from the *Modeling* tab in the Reliability Options form, add the following SKILL code in the `.cdsinit` file:

```
(defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)  
    relxHideAgeCalculationApproachField(relxOptionForm)  
)
```

relxGetCustomTabName

```
relxGetCustomTabName (  
    )  
=> t_customTabName
```

Description

Returns the name of the custom tab. You can also use this function to rename the custom tab.

Arguments

None

Value Returned

t_customTabName Returns the name of the custom tab.

Examples

The following example returns the name of the custom tab:

```
relxGetCustomTabName ()  
=> "Custom"
```

To rename the custom tab to `myCustomTab`, add the following SKILL code in the `.cdsinit` file:

```
(defun relxGetCustomTabName ()  
    "myCustomTab"  
)
```

relxCustomizeDisplayOrEnableStatus

```
relxCustomizeDisplayOrEnableStatus(  
    r_formObject  
)  
=> t
```

Description

Defines the display and enable status of the disclosures, fields, and tabs.

Note: You need to define this function in the `.cdsinit` file. You need not call it.

Arguments

`r_formObject` The form object: `relxOptionForm`.

Value Returned

`t` Returns `t` by default.

Example

Add the following SKILL code in the `.cdsinit` file to:

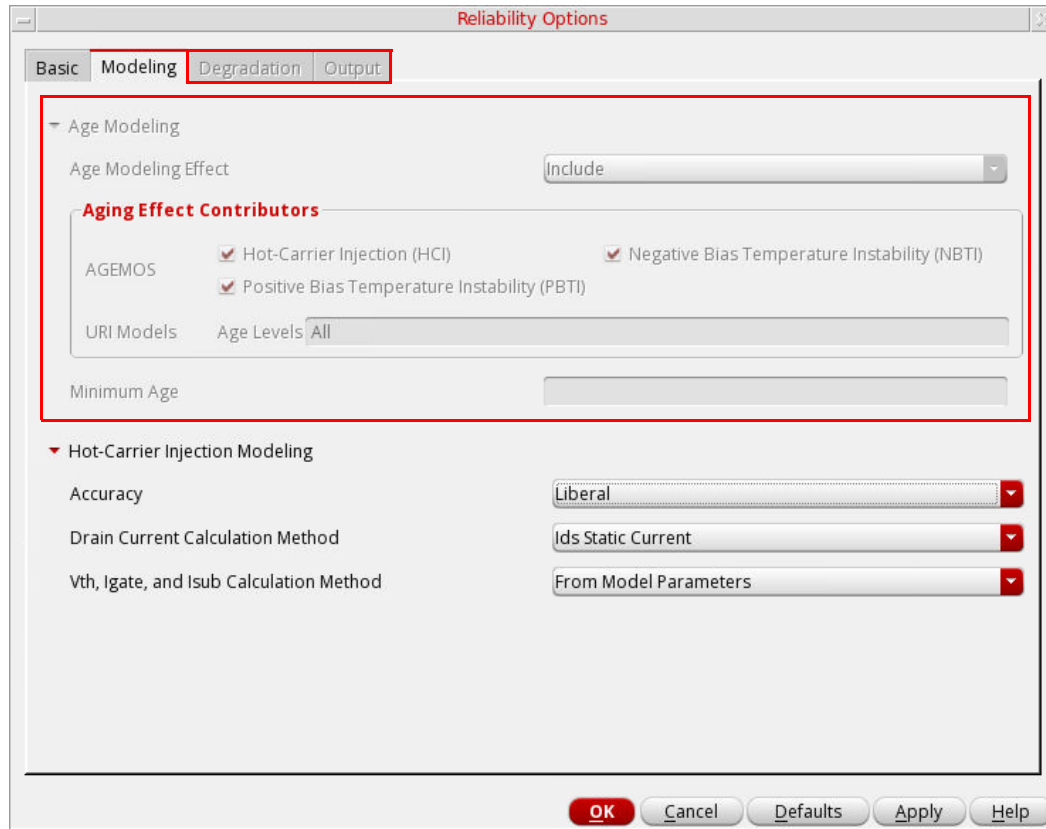
- Disable the *Degradation* and *Output* tabs
- Disable the *Age Modeling* section on the *Modeling* tab
- Hides the *Unified Reliability Interface (URI) Models* disclosure from the *Modeling* tab

```
(defun relxCustomizeDisplayOrEnableStatus (relxOptionForm)  
    relxEnableFormTab(relxOptionForm "Degradation" nil)  
    relxEnableFormTab(relxOptionForm "Output" nil)  
    relxEnableDiscField(relxOptionForm "Age Modeling" nil)  
    relxDisplayDiscField(relxOptionForm "Unified Reliability Interface (URI)  
Models" nil)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

)



relxCreateCustomizedTab

```
relxCreateCustomizedTab(  
    s_formSymbol  
)  
=> nil
```

Description

Adds a customized tab to the Reliability Options form.

Arguments

<i>s_formSymbol</i>	The form symbol

Value Returned

<i>nil</i>	Returns <i>nil</i> by default.

Example

Add the following SKILL code in the `.cdsinit` file to:

- Create a customized tab named `myCustomTab`
- Define the following fields:
 - ❑ *Circuit Report* of type `Boolean`
 - ❑ *Circuit Report Path* of type `string`
 - ❑ *Circuit Type* of type `radio`
 - ❑ *Include Circuit* of type `cyclic`
 - ❑ *Circuit Mode* of type `toggle`

```
(defun relxGetCustomTabName ()  
    "myCustomTab"  
)  
  
(defun relxCreateCustomizedTab (formSymbol)  
    let((cusTagVertLayout circuitReport circuitReportPath circuitType circuitMode  
        circuitInclude)  
        circuitReport = relxAddSetupRelxOption( ?name 'circuitReport
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

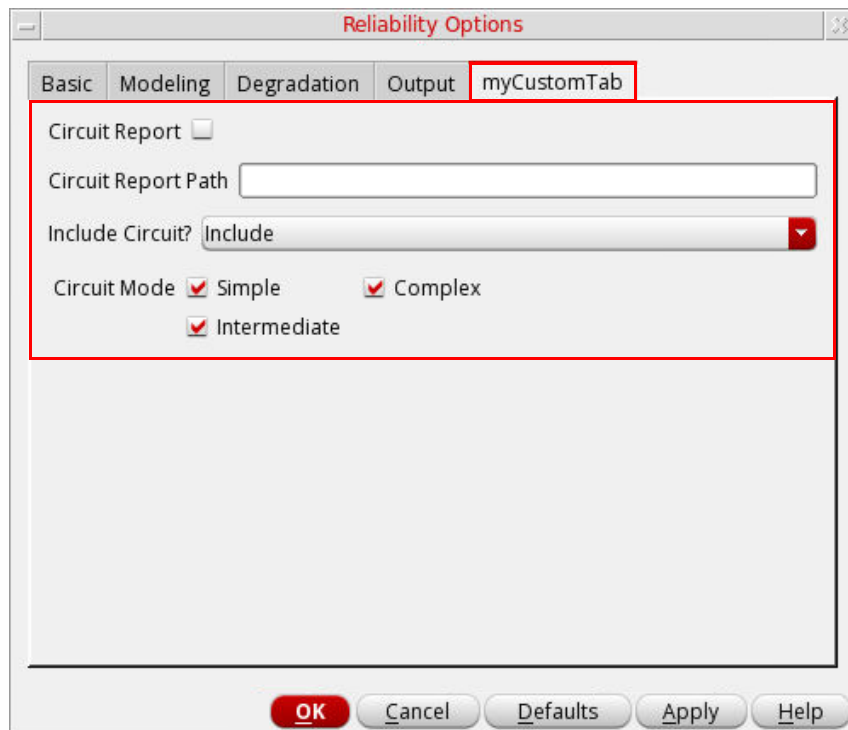
```
        ?type 'Boolean
        ?prompt "Circuit Report"
        ?defValue nil
    )
circuitReportPath = relxAddSetupRelxOption(?name 'circuitReportPath
        ?type 'string
        ?prompt "Circuit Report Path"
    )
circuitType = relxAddSetupRelxOption(?name 'circuitType
        ?type 'radio
        ?choices list("Static" "Dynamic")
        ?value "Static"
        ?prompt " Circuit Type"
    )
circuitMode = relxAddSetupRelxOption(?name 'circuitMode
        ?type 'toggle
        ?choices list( list( 't1 "Simple")
                        list( 't2 "Complex")
                        list( 't3 "Intermediate")
                      )
        ?value list( t t t)
        ?itemsPerRow 2
        ?prompt " Circuit Mode"
    )
circuitType = relxAddSetupRelxOption(?name 'circuitInclude
        ?type 'cyclic
        ?choices list("Include" "Exclude")
        ?value "Include"
        ?prompt "Include Circuit?"
    )
cusTagVertLayout = relxAddSetupRelxOption( ?name 'cusTagVertLayout
        ?type 'verticalBoxLayout
        ?items list(
            list(circuitReport)
            list(circuitReportPath)
            list(circuitType)
            list(circuitMode)
            list(circuitInclude)
            list('spacer_item 5)
            list('stretch_item 5)
        )
    )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
)  
    cusTagVertLayout  
)  
)
```

The customized tab, *myCustomTab* is added to the Reliability Options form. The tab displays the fields defined by using the `relxAddSeupRelxOption` SKILL function.



relxAddSetupRelxOption

```
relxAddSetupRelxOption (
    [ ?name s_name ]
    [ ?type s_type ]
    [ ?prompt t_prompt ]
    [ ?choices l_choices ]
    [ ?value g_value ]
    [ ?display g_display ]
    [ ?enabled g_enabled ]
    [ ?onFields l_onFields ]
    [ ?buttonLocation s_buttonLocation ]
    [ ?callback t_callback ]
    [ ?defValue g_defValue ]
    [ ?itemsPerRow x_itemsPerRow ]
    [ ?items l_items ]
    [ ?frame g_frame ]
    [ ?formApplyCB s_formApplyCB ]
    [ ?scrollable g_scrollable ]
    [ ?tooltip t_tooltip ]
    [ ?spacing x_spacing ]
    [ ?discLayout s_discLayout ]
    [ ?buttonIcon g_buttonIcon ]
)
=> r_fieldHandle
```

Description

Adds new fields to the Reliability Options form.

This function can be used as a unit function to customize the *Custom* tab.

Arguments

<code>?name s_name</code>	The symbol name of the field to be added.
<code>?type s_type</code>	Type of the field. Default value: string. Possible values: radio, Boolean, separator, label, cyclic, string, toggle, and button.
<code>?prompt t_prompt</code>	Prompt for the field.
<code>?choices l_choices</code>	List of choices.
<code>?value g_value</code>	Value of the field.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?display g_display</code>	Controls the display of fields on the form. If this argument is set to <code>nil</code> , the field is not displayed in the form.
<code>?enabled g_enabled</code>	Enables or disables the field. When set to <code>t</code> , the field is enabled, When set to <code>nil</code> , the field is disabled and becomes inactive in the form.
<code>?onFields l_onFields</code>	It is applicable only for the <code>disclosureTriangle</code> type of fields.
<code>?buttonLocation s_buttonLocation</code>	Location of the button. It is applicable only for the <code>Boolean</code> type of fields.
<code>?callback t_callback</code>	Specifies the SKILL functions (callbacks) that execute after you select <i>Apply</i> , <i>OK</i> , or <i>Cancel</i> in the form.
<code>?defValue g_defValue</code>	Default value of the field.
<code>?itemsPerRow x_itemsPerRow</code>	Number of items in a single row. It is applicable only for the <code>radio</code> and <code>toggle</code> type of fields.
<code>?items l_items</code>	Defines the items in the layout. It is applicable only for <code>gridLayout</code> , <code>horizontalBoxLayout</code> , and <code>verticalBoxLayout</code> fields.
<code>?frame g_frame</code>	Any of the following values: ■ <code>nil</code>: (Default) For an unframed layout. ■ <code>t</code>: For an untitled frame. ■ <code>t_title</code>: For a framed layout with a title. It is applicable only for <code>gridLayout</code> , <code>horizontalBoxLayout</code> , and <code>verticalBoxLayout</code> fields.
<code>?formApplyCB s_formApplyCB</code>	Calls the <code>formApplyCB</code> function set when a button, such as <i>OK</i> , <i>Apply</i> , and so on, is clicked.
<code>?scrollable g_scrollable</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

If set to `t`, adds a scroll bar in the layout if the content is large and cannot be accommodated. The default value is `nil`.

It is applicable only for `gridLayout`, `horizontalBoxLayout`, and `verticalBoxLayout` fields.

`?tooltip t_tooltip`

Tooltip for the field.

`?spacing x_spacing`

Specifies the integer value to apply for horizontal spacing.

`?discLayout s_discLayout`

Defines the layout of the disclosure triangle.

`?buttonIcon g_buttonIcon`

Defines the button icon.

Value Returned

`r_fieldHandle` Handle to the field.

Examples

Add the following SKILL code in the `.cdsinit` file to add these fields to the *Custom* tab:

- *Circuit Report* of type `Boolean`
- *Circuit Report Path* of type `string`
- *Circuit Type* of type `radio`
- *Include Circuit* of type `cyclic`
- *Circuit Mode* of type `toggle`

```
(defun relxCreateCustomizedTab (formSymbol)
  let((cusTagVertLayout circuitReport circuitReportPath circuitType circuitMode
      circuitInclude)
    circuitReport = relxAddSetupRelxOption( ?name 'circuitReport
                                           ?type 'Boolean
                                           ?prompt "Circuit Report"
                                           ?defValue nil
                                           )
  )
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
circuitReportPath = relxAddSetupRelxOption(?name 'circuitReportPath
                                           ?type 'string
                                           ?prompt "Circuit Report Path"
                                           )

circuitType = relxAddSetupRelxOption(?name 'circuitType
                                     ?type 'radio
                                     ?choices list("Static" "Dynamic")
                                     ?value "Static"
                                     ?prompt " Circuit Type"
                                     )

circuitMode = relxAddSetupRelxOption(?name 'circuitMode
                                     ?type 'toggle
                                     ?choices list( list( 't1 "Simple")
                                                    list('t2 "Complex")
                                                    list( 't3 "Intermediate")
                                                  )
                                     ?value list( t t t)
                                     ?itemsPerRow 2
                                     ?prompt " Circuit Mode"
                                     )

circuitType = relxAddSetupRelxOption(?name 'circuitInclude
                                     ?type 'cyclic
                                     ?choices list("Include" "Exclude")
                                     ?value "Include"
                                     ?prompt "Include Circuit?"
                                     )

cusTagVertLayout = relxAddSetupRelxOption( ?name 'cusTagVertLayout
                                           ?type 'verticalBoxLayout
                                           ?items list(
                                               list(circuitReport)
                                               list(circuitReportPath)
                                               list(circuitType)
                                               list(circuitMode)
                                               list(circuitInclude)
                                               list('spacer_item 5)
                                               list('stretch_item 5)
                                           )
                                           )

cusTagVertLayout
)

)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

The new fields are added to the Custom tab in the Reliability Options form.

The screenshot shows the 'Reliability Options' dialog box with the 'Custom' tab selected. A red rectangle highlights the new fields added to this tab. The fields include: 'Circuit Report' (checkbox), 'Circuit Report Path' (text input), 'Include Circuit?' (dropdown menu set to 'Include'), and 'Circuit Mode' (checkboxes for Simple, Complex, and Intermediate, all of which are checked). The 'OK' button is highlighted in red.

Reliability Options

Basic Modeling Degradation Output Custom

Circuit Report ☐

Circuit Report Path

Include Circuit? Include ▼

Circuit Mode ☒ Simple ☒ Complex ☒ Intermediate

OK Cancel Defaults Apply Help

asiFormatSpecialParameterForRel

```
asiFormatSpecialParameterForRel(  
    o_oasisSession  
    p_fp  
)  
=> t
```

Description

Netlists the new options added to the customized tab.

You need to define this function in the `.cdsinit` file. You need not call it.

Arguments

<code>o_oasisSession</code>	The OASIS session object.
<code>p_fp</code>	The handle to the netlist file.

Value Returned

<code>t</code>	Returns <code>t</code> by default.

Examples

Add the following SKILL code in the `.cdsinit` file to netlist the `circuitReport` and `circuitReportPath` options added to the *Custom* tab.

```
(defun asiFormatSpecialParameterForRel (session)  
    when(relxGetReliabilityOptionVal(session 'circuitReport)  
        (artFprintf fp "parameters circuit_report=1\n")  
        (artFprintf fp "parameters path_report=%s\n"  
            relxGetReliabilityOptionVal(session 'circuitReportPath))  
    )  
)
```

relxInitOptionsInCdsenv

```
relxInitOptionsInCdsenv(  
    o_tool  
)  
=> nil
```

Description

Creates environment variables for the customized options added to the Reliability Options form.

You need to define this function in the `.cdsinit` file. You need not call it.

Arguments

<code>o_tool</code>	Simulation tool object.

Value Returned

<code>nil</code>	Returns <code>nil</code> by default.

Examples

Add the following SKILL code in the `.cdsinit` file to add the customized variables `circuitReport` and `circuitReportPath` in the `.cdsenv` file: .

```
(defun relxInitOptionsInCdsenv (tool)  
    relxAddReliabilityOption( tool  
                                ?name      'circuitReport  
                                ?type      'Boolean  
                                ?value     nil  
                                ?page      "Custom"  
                                ?private   t  
                                )  
    relxAddReliabilityOption( tool  
                                ?name      'circuitReportPath  
                                ?type      'string  
                                ?value     ""  
                                ?page      "Custom"  
                                ?private   t  
                                )  
)
```

relxAddReliabilityInStateComponent

```
relxAddReliabilityInStateComponent (
    t_sessionName
)
=> t / nil
```

Description

Adds the Reliability Setup option in the save and load state forms of the specified session. To reuse the save and load state flow of ADE in your simulator, you can rewrite this function and ensure that the function returns `t`.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<code>t</code>	Returns <code>t</code> when the function is rewritten for other sessions.
<code>nil</code>	Returns <code>nil</code> when the function is run in the specified session.

Examples

In this example, the function adds the Reliability Setup option to the save and load state flow in the current session.

```
session = asiGetCurrentSession()
relxAddReliabilityInStateComponent(session)
```


relxAddReliabilityOption

```
relxAddReliabilityOption (
    t_toolName
    [ ?name t_name ]
    [ ?value t_value ]
    [ ?type t_type ]
    [ ?mode t_mode ]
    [ ?enabled g_enabled ]
    [ ?prompt t_prompt ]
    [ ?display g_display ]
    [ ?choices l_choices ]
    [ ?callback g_callback ]
    [ ?formApplyCB g_formApplyCB ]
    [ ?page t_page ]
    [ ?coordinates l_coordinates ]
    [ ?private g_private ]
)
=> t_relVar / nil
```

Description

Adds a variable to the specified Reliability tool.

Arguments

<i>t_toolName</i>	Name of the Reliability tool to which you want to add the variable.
<i>?name t_name</i>	Name of the variable to be added.
<i>?value t_value</i>	Value of the variable according to the type of the variable.
<i>?type t_type</i>	Type of the variable to be added. Default value: <code>string</code> . Possible values: <code>radio</code> , <code>Boolean</code> , <code>separator</code> , <code>label</code> , <code>cyclic</code> , <code>string</code> , <code>toggle</code> , or <code>button</code> .
<i>?mode t_mode</i>	Specifies the file in which the specified variable is to be added. Possible values: <code>existingFile</code> or <code>anyFile</code> .
<i>?enabled g_enabled</i>	Enables or disables the variable. When set to <code>t</code> , the variable is enabled, When set to <code>nil</code> , the variable is disabled and becomes inactive in the form.
<i>?prompt t_prompt</i>	Description to be displayed on the form.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?display g_display</code>	Controls the display of variable on the form. If this argument is set to <code>nil</code> , the variable is not displayed on the form.
<code>?choices t_choices</code>	List of choices, if the variable has multiple choices of values.
<code>?callback g_callback</code>	Calls the <code>callback</code> function when the value of the variable is changed.
<code>?formApplyCB g_formApplyCB</code>	Calls the <code>formApplyCB</code> function set when a button, such as <i>OK</i> , <i>Apply</i> , and so on, is clicked.
<code>?page t_page</code>	If the form includes multiple pages, specify the page where the variable is to be added.
<code>?coordinates l_coordinates</code>	Specifies the coordinates on the Reliability form where the variable is to be added. The coordinates are specified as: <code>list(x:y width:height labelWidth)</code> .
<code>?private g_private</code>	Boolean that indicates whether the variable is private and not to be included in netlist. If set to <code>t</code> , the variable is not included in the netlist. The default value of this argument is <code>nil</code> , which means that the variable is netlisted by default.

Value Returned

<code>t_relVar</code>	For the specified analog tool, the function returns the variable object.
<code>nil</code>	For any other tool, return value is <code>nil</code> .

Examples

In the example given below, the function adds a variable, *test2*, of type `radio` on the Reliability form with the following argument values:

```
relxAddReliabilityOption (  
tool  
?name 'test2  
?type 'radio  
?prompt "Test2 Mode"  
?display 'FuncGetDisplay(hiGetCurrentForm())
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
?choices list( "model" "mode2" )
?enabled      'FuncEnable( hiGetCurrentForm() )
?callback "TestModeCB()"
?formApplyCB 'TestModeAppCB
?value "model"
?page"Basic"
?coordinates list( xOrig:yCoor+delta fieldWidth:delta labelWidth )
?private nil
)
```

relxCreateRunObjectFile

```
relxCreateRunObjectFile(  
    t_sessionName  
)  
=> t / nil
```

Description

Creates the `runObject` file for the specified session.

You need to rewrite this function if you have a different results directory as compared to the results directory for analyses in Spectre.

Arguments

<code>t_sessionName</code>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<code>t</code>	If the session is created using Spectre, There is no error.
<code>nil</code>	Returns <code>nil</code> and displays an error message in CIW if there is an error. Also, for any other session, returns <code>nil</code> .

Examples

Consider the following example in which the ADE L session is created using Spectre. Now, if you run the `relxCreateRunObjectFile` function, it returns `t` when the `runObject` file is successfully created.

```
session = asiGetCurrentSession()  
relxCreateRunObjectFile(session)  
=> t
```

relxDisplayReliabilityForm

```
relxDisplayReliabilityForm(  
    s_sevSession  
)  
=> t / nil
```

Description

Displays the Reliability form for the specified session.

Arguments

<i>s_sevSession</i>	The simulation environment session.

Value Returned

<i>t</i>	For the specified session, the return value is <i>t</i> when the function runs successfully and reliability form is displayed.
<i>nil</i>	For any others session, returns <i>nil</i> . Also, There is an error.

Examples

In this example, the function displays the Reliability form for the specified *sev* session and returns *t*.

```
relxDisplayReliabilityForm(sevSession)  
=> t
```

relxDisplayResult

```
relxDisplayResult (
    t_sessionName
    t_file
)
=> t / nil
```

Description

Displays the specified result log file.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>t_file</i>	Name or path of the results log file. If the value of this argument is a filename, the function searches for the specified filename in the netlist directory and displays when found.

Value Returned

<i>t</i>	For the specified session, returns <i>t</i> if function runs successfully.
<i>nil</i>	For any other session, returns <i>nil</i> . Also, There is an error.

Examples

In the example given below, the function searches for `relXpert` log file in the netlist directory and opens the file when it is found. When the file is found, returns *t*, else displays a warning message.

```
session = asiGetCurrentSession()
relxDisplayResult(session "relxpert.log")
=> t
```

In the example given below, the function searches for `relXpert` log file at the specified path and opens the file when it is found. When the file is found, returns *t*, else displays a warning message.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
relxDisplayResult(session "/dir/relxpert.log")  
=> t
```

relxFormatRXControlFile

```
relxFormatRXControlFile(  
    t_sessionName  
)  
=> t / nil
```

Description

When the session is created using Spectre, this function creates a file that includes all the possible options for Relxpert Reliability simulator mode. This function does not work for other sessions. If Lynx's netlist flow is based on the result of this function, you need to rewrite the `relxGetRXControlFileName` function.

Arguments

<code>t_sessionName</code>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<code>t</code>	For the specified Spectre session, returns <code>t</code> when the function runs successfully. Also, for any other session, returns <code>t</code> .
<code>nil</code>	There is an error.

Examples

Consider the following example in which the ADE L session is created using Spectre. Now, if you run this function, it returns `t` when the file containing the Relxpert Reliability simulator options is successfully created.

```
session = asiGetCurrentSession()  
relxFormatRXControlFile(session)  
=> t
```


relxGetModifyNetlistVal

```
relxGetModifyNetlistVal(  
    t_sessionName  
)  
=> t / nil
```

Description

Checks whether to modify the netlist before aging starts. You can rewrite this function as per your requirements.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t</i>	When the session is created using Spectre, The netlist is to be modified.
<i>nil</i>	The netlist is not to be modified or if there is an error. Also, for any other session, returns <i>nil</i> .

Examples

Consider the following example in which you create an ADE L session using Spectre. Now, if you run the `relxGetModifyNetlistVal` function, the function The netlist is to be modified. Otherwise, returns *nil*.

```
session = asiGetCurrentSession()  
relxGetModifyNetlistVal(session)  
=> t
```

relxGetMosAgingTimeUnitVal

```
relxGetMosAgingTimeUnitVal(  
    t_sessionName  
)  
=> t_agingTimeUnit / nil
```

Description

Returns the unit of aging time, which can be represented in years, days, hours, minutes or seconds. Only one unit of time is supported for a given simulation run.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t_agingTimeUnit</i>	When the session is created using Spectre, returns the aging time unit.
<code>nil</code>	For any other session, returns <code>nil</code> . Also, There is an error.

Examples

Consider the following example in which you create a session using Spectre simulator. Now, if you run the `relxGetMosAgingTimeUnitVal` function, it returns the aging time unit in years.

```
session = asiGetCurrentSession()  
relxGetMosAgingTimeUnitVal(session)
```

relxGetMosAgingTimeVal

```
relxGetMosAgingTimeVal (
    t_sessionName
)
=> t_agingTimeValue / nil
```

Description

Returns the aging time value of the simulation in the specified session. The aging time can be in years, days, hours, minutes or seconds. Only one unit of time is supported for a given simulation run.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t_agingTimeValue</i>	When the session is created using Spectre, returns the aging time value of the simulation in the specified session.
<i>nil</i>	For any other session, returns <code>nil</code> . Also, There is an error.

Examples

In the example given below, suppose the simulation aging time value in the current Spectre session is 10. In this case, the `relxGetMosAgingTimeVal` function returns the value 10.

```
session = asiGetCurrentSession()
relxGetMosAgingTimeVal(session)
=> 10
```

relxGetReliabilityOptionChoices

```
relxGetReliabilityOptionChoices(  
    t_sessionName  
    s_optionName  
)  
=> t_optionValues / nil
```

Description

Returns the possible values for the specified Reliability option in the given analog session. This function works for fields that are of type `radio`, `cyclic`, and `toggle`.

It is a unit function used for netlist and control flow. It is recommended not to rewrite this function.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>s_optionName</i>	Name of the option whose possible values you want to return. Ensure that you specify the same option name as defined in your session. The name should be in symbol format.

Value Returned

<i>t_optionValues</i>	Returns possible values for the specified Reliability option in the given session.
<i>nil</i>	For any other session, returns <code>nil</code> .

Examples

Consider the example given below, in which you create an ADE L session using Spectre. The current session name is returned as shown:

```
session = asiGetCurrentSession()
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Now, if you run the `relxGetReliabilityOptionChoices` function for the current session and variable name is `AnalysisType`, the function returns the all possible types for analysis, such as `tran`, `AC`, `DC`.

```
relxGetReliabilityOptionChoices(session 'AnalysisType')
)
```

relxGetReliabilityOptionVal

```
relxGetReliabilityOptionVal(  
    t_sessionName  
    s_optionName  
)  
=> t_reliabilityOptionValue/ nil
```

Description

Returns the value of the specified Reliability option variable in the given session.

It is a unit function for netlist and control flow. It is recommended not to rewrite this function.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>s_optionName</i>	Name of the variable whose value you want to return. The value should be in symbol format.

Value Returned

<i>t_ReliabilityOptionValue</i>	For Spectre session, returns the value of the specified Reliability option.
<i>nil</i>	For any other session, returns <code>nil</code> .

Examples

In this example, the function returns the value of Reliability option `xyz`.

```
session = asiGetCurrentSession()  
relxGetReliabilityOptionVal(session 'xyz')
```

relxGetRelxStage

```
relxGetRelxStage(  
    o_session  
)  
=> t_relxStage / nil
```

Description

Returns the current stage of the reliability simulation in the given session.

Arguments

<i>o_session</i>	Session object of the current session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

t_relxStage

Returns one of the following values:

- `stress`: Stress simulation stage
- `aging`: Aging simulation stage
- `all`: Stress and aging simulations are running on a single netlist.

<i>nil</i>	Returns an empty string if reliability analysis is not enabled in the test setup or if there is an error.

Examples

In this example, the function returns "aging", which indicates that the current stage of reliability simulation is aging.

```
session = asiGetCurrentSession()  
relxGetRelxStage(session)  
=> "aging"
```

relxGetStressFileDir

```
relxGetStressFileDir(  
    o_session  
)  
=> t_stressFileDir / nil
```

Description

Returns the directory of the stress file.

Arguments

<i>o_session</i>	Session object of the current session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t_stressFileDir</i>	Directory where the stress file, <code>input.bs0</code> , is saved.
<i>nil</i>	Returns <code>nil</code> if reliability analysis is not enabled in the test setup or if there is an error.

Examples

In this example, the function returns the directory of the stress file.

```
session = asiGetCurrentSession()  
relxGetStressFileDir(session)
```


relxGetRXControlFileName

```
relxGetRXControlFileName(  
    t_sessionName  
)  
=> t_RXControlFilename / nil
```

Description

Returns the name of the `RXControl` file.

If you want to format your `RXControl` file, you need to rewrite this function. Then, the function returns the name of the formatted `RXControl` file.

Arguments

<code>t_sessionName</code>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<code>" "</code>	Returns an empty string as the filename in the specified Spectre session because by default netlist does not perform any operation.
<code>t_RXControlFilename</code>	Returns the <code>RXControl</code> filename when the function is rewritten.
<code>nil</code>	There is an error.

Examples

Consider the following example in which the ADE L session is created using Spectre. If you run this function, it returns an empty string.

```
session = asiGetCurrentSession()  
relxGetRXControlFileName(session)  
=> ""
```

relxGetSimulationRunCommand

```
relxGetSimulationRunCommand (
    t_sessionName
)
=> t_runSimulationFileName / nil
```

Description

Formats the content of the simulation run file present in the netlist directory. This file is required to run simulation flow.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t_runSimulationFileName</i>	For the specified Spectre session, returns the name of the simulation run file.
<i>nil</i>	Returns <code>nil</code> for any other sessions or if there is an error.

Examples

In this example, the function returns the name of the simulation run file for the current session.

```
session = asiGetCurrentSession()
relxGetSimulationRunCommand( session )
```

relxGetSpecifiedReliabilityStateFileName

```
relxGetSpecifiedReliabilityStateFileName(  
    t_sessionName  
)  
=> t_filename/ nil
```

Description

Returns the name of the state file used the Reliability form. By default, the name of the state file is `relxOptions`. However, you can rewrite this function to define another name for the state file for your simulator.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t_filename</i>	If the session is created using Spectre, returns the name of the state file.
<code>nil</code>	For any other sessions, returns <code>nil</code> . Also, There is an error.

Examples

Consider the below example in which you define the name of the state file for your simulator. The `asiGetSpecifiedReliabilityStateFileName` function returns the specified name of the state file.

```
(defmethod asiGetSpecifiedReliabilityStateFileName((session  
    asiAnalogAdapter_session))  
    "LynxOptions"  
)  
  
session = asiGetCurrentSession()  
asiGetSpecifiedReliabilityStateFileName(session)  
=> LynxOptions
```

relxGetUserCmdLine

```
relxGetUserCmdLine(  
    t_sessionName  
)  
=> t_UserCmdLine / nil
```

Description

Returns the value of the `userCmdLine` option in the specified session.

Arguments

<code>t_sessionName</code>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<code>t_UserCmdLine</code>	When the session is created using Spectre, returns the value of <code>userCmdLine</code> option(s) for Spectre.
<code>nil</code>	For any others sessions, returns <code>nil</code> .

Examples

Consider the example given below, in which you create an ADE L session using Spectre and specify `+aps` command-line option for Spectre. In this case, this function returns `+aps`.

```
session = asiGetCurrentSession()  
relxGetUserCmdLine (session)  
=> aps
```

relxHighLightDevices

```
relxHighLightDevices(  
    t_sessionName  
    l_instList  
    l_tagList  
)  
=> t / nil
```

Description

Highlights the specified device and instance list on schematic, and also displays information for the selected devices and instances in the form of tags.

It is recommended not to rewrite this function.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>l_instList</i>	List of instances that you want to highlight on schematic.
<i>l_tagList</i>	List of tags used to highlight the specified instances.

Value Returned

<i>t</i>	For the specified Spectre session, returns <i>t</i> when the function runs successfully.
<i>nil</i>	There is an error. Also, for any other session, returns <i>nil</i> .

Examples

Consider the example given below, in which you create an ADE L session using Spectre. You can use the `relxHighLightDevices` function to highlight the following instances with special tags.

```
session = asiGetCurrentSession()  
instList = '((nil nbtI 0.1232 hci_nbtI 0.1237  
    agetime "10.00yrs" instname "I0.I0.M#2310#" maxIb  
    1.686e-09 avgIb 6.572e-11 maxIg 1.265e-12  
    avgIg 6.387e-14 Degrad 0.1237  
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
(nil nbti 0.1229 hci_nbti 0.1234
agetime "10.00yrs" instname "I0.I1.M#2310#" maxIb
1.781e-09 avgIb 7.977e-11 maxIg 1.311e-12
avgIg 7.737e-14 Degrad 0.1234
)
(nil nbti 0.1258 hci_nbti 0.1264
agetime "10.00yrs" instname "I0.I2.M#2310#" maxIb
1.687e-09 avgIb 6.534e-11 maxIg 1.265e-12
avgIg 6.354e-14 Degrad 0.1264
)
)
taglist = '(hci nbti)
relxHighLightDevices(session instList taglist)
```

relxInitAdapterReliabilityOption

```
relxInitAdapterReliabilityOption(  
    t_toolName  
)
```

Description

Rewrites the `asiInitialize` function when called in the initialization flow for the specified tool. This function internally calls the `relxInitReliabilityOption` function and also adds some private variables to this function.



It is recommended not to rewrite this function.

Arguments

<code>t_toolName</code>	Name of the tool for which you want to rewrite the <code>asiInitialize</code> function.

Value Returned

None

Examples

```
relxInitAdapterReliability(tool)
```

relxInitReliabilityOption

```
relxInitReliabilityOption (
    t_toolName
)
=> nil
```

Description

Initializes the Reliability Option form in the specified tool. You can rewrite this function by using unit functions `relxAddReliabilityOption` and `relxSetReliabilityOptionFormProperties`.

Arguments

<i>t_toolName</i>	Name of the tool in which the Reliability Option form is to be initialized.

Value Returned

<i>nil</i>	Returns <i>nil</i> by default.
	This function can be rewritten to return any user specified value.

Examples

In this example, the function initializes the Reliability Option form in the specified tool.

```
relxInitReliabilityOption(tool)
```

The example below shows the function code used to rewrite the `relxInitReliabilityOption` function:

```
(defmethod relxInitReliabilityOption ((tool asiAnalogAdapter))
  (let ((xOrig 5) (yOrig 10) (delta 35) (fieldWidth 600) (labelWidth 260)
        (yCoor 10))

    pageList = list(list("Basic" 'custom) list("Advanced" 'custom))

    relxSetReliabilityOptionFormProperties( tool
      ?title      "Reliability"
      ?width      xOrig*2+fieldWidth
      ?height     yOrig+delta*30
      ?type       'multiPage
      ?groupType  'tabs
```


Virtuoso ADE SKILL Reference
Functions to Customize Simulator Integration

```
    ?pageList    pageList
  )
))
```

relxIsAgingOn

```
relxIsAgingOn(  
    t_sessionName  
)  
=> t / nil
```

Description

Checks whether aging is ON or OFF in the specified session.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t</i>	Aging is ON in the specified Spectre session.
<i>nil</i>	Aging is OFF in the specified Spectre session or if there is an error. Also, returns <i>nil</i> for any other sessions.

Examples

In this example, the function returns *t* if aging is ON in the current session.

```
session = asiGetCurrentSession()  
relxIsAgingOn (session)  
=> t
```

relxIsReliabilityEnabled

```
relxIsReliabilityEnabled(  
    t_sessionName  
)  
=> t / nil
```

Description

Checks whether the Reliability analysis is enabled or disabled in the specified session.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t</i>	The Reliability analysis is enabled in the specified Spectre session.

<i>nil</i>	The Reliability analysis is disabled in the specified Spectre session or if there is an error.

For other sessions, return value is always *nil*.

Examples

Consider the following example in which the ADE L session is created using Spectre. Now, when you run the `relxIsReliabilityEnabled` function, it The Reliability analysis is enabled.

```
session = asiGetCurrentSession()  
relxIsReliabilityEnabled(session)  
=> t
```

relxIsStressOn

```
relxIsStressOn(  
    t_sessionName  
)  
=> t / nil
```

Description

Checks whether the stress is **ON** or **OFF** in the specified session.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t</i>	The stress is ON in the specified session.
<i>nil</i>	The stress is OFF or if there is an error. Also, returns <i>nil</i> when you are using other sessions.

Examples

In this example, the stress is **ON** in the current session.

```
session = asiGetCurrentSession()  
relxIsStressOn (session)  
=> t
```

relxPostSimulation

```
relxPostSimulation (
    t_sessionName
)
```

Description

Used to post process the simulation result.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t</i>	Returns <i>t</i> when the function runs successfully.
<i>nil</i>	There is an error.

Examples

In this example, the function posts the simulation results in the current session.

```
session = asiGetCurrentSession()
relxPostSimulation(session)
```

relxRunSimulation

```
relxRunSimulation(  
    t_sessionName  
)  
=> t_processId / nil
```

Description

Runs the Relxpert simulation.

It is recommended not to rewrite this function. This function can be added to the `asiRunSimulation` function when you rewrite it.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.

Value Returned

<i>t_processId</i>	Returns a string that displays the process ID, which can be pending, error, complete, and so on.
<i>nil</i>	Returns <code>nil</code> for other sessions or if there is an error.

Examples

In this example, the function runs the simulation for Relxpert in the current session.

```
session = asiGetCurrentSession()  
relxRunSimulation(session)
```

relxSetAgingVal

```
relxSetAgingVal (
    t_sessionName
    g_value
)
=> t / nil
```

Description

Enables or disables aging for simulation in the specified session. Also sets the specified simulation aging value, while enabling aging.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>g_value</i>	Aging value that you want to set. The format of value depends on the definition of aging variable.

Value Returned

<i>t</i>	When the session is created using Spectre, this function returns <i>t</i> , when the aging value is successfully set.
<i>nil</i>	Returns <i>nil</i> when there is an error. Also, returns <i>nil</i> for any other sessions.

Examples

The function sets the aging value as 10 in the current session.

```
session = asiGetCurrentSession()
relxSetAgingVal(session '10')
=> t
```

relxSetReliabilityOptionFormProperties

```
relxSetReliabilityOptionFormProperties(  
    t_toolName  
    [ ?title t_title ]  
    [ ?width d_width ]  
    [ ?height d_height ]  
    [ ?columns d_columns ]  
    [ ?help s_help ]  
    [ ?type t_type ]  
    [ ?groupType t_groupType ]  
    [ ?pageList l_pageList ]  
)  
=> o_formObj / nil
```

Description

Sets the various properties of the Reliability form.

Arguments

<code>t_toolName</code>	Name of the tool.
<code>?title t_title</code>	Title of the form.
<code>?width d_width</code>	Width of the form (in pixels).
<code>?height d_height</code>	Height of the form (in pixels).
<code>?column d_columns</code>	Number of columns.
<code>?help s_helpSymbol</code>	Help symbol of the form.
<code>?type t_Type</code>	Type of the form. Indicates if the form covers one page or multiple pages. Possible values: <code>oneD</code> or <code>multiPage</code> .
<code>?groupType t_groupType</code>	Group type to be used in a multiple page form. Possible values: <code>tabs</code> or <code>trees</code> .
<code>?pageList t_title</code>	List of pages and their types.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<i>o_formObj</i>	Returns the Reliability form object.
<i>nil</i>	For any other tools, returns <i>nil</i> . Also There is an error.

Examples

Consider the example given below, in which you create an ADE L session using Spectre and run this function using the following argument values:

```
relxSetReliabilityOptionFormProperties (
tool
?title "Reliability"
?width 600
?height 80
?type 'multiPage
?groupType 'tabs
?pageList list(list("Page1" 'custom) list("Page2" 'custom))
)
```

relxSetReliabilityOptionVal

```
relxSetReliabilityOptionVal(  
    t_sessionName  
    s_optionName  
    t_value  
)  
=> g_value / nil
```

Description

Sets the value of the specified Reliability option variable in the given session.

It is a unit function for netlist and control flow. It is recommended not to rewrite this function.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>s_optionName</i>	Name of the Reliability option whose value you want to set. Ensure that you use the same option name as defined in your analog session. The option name should be in symbol format.
<i>t_value</i>	The value that you want to assign to the specified Reliability option. The value depends on the variable type.

Value Returned

<i>g_value</i>	For Spectre session, returns the value of the Reliability option variable.
<i>nil</i>	For any other sessions, returns <code>nil</code> . Also, There is an error.

Examples

Consider the example given below, in which the ADE L session is created using Spectre. You can use the following function to set the value of the specified reliability option variable as STRESS.

```
session = asiGetCurrentSession()  
relxSetReliabilityOptionVal(session 'variable_name "STRESS")
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Now, return the specified value using the following function:

```
relxGetReliabilityOptionVal(session 'variable_name')  
=> STRESS
```

relxSetReliabilityVal

```
relxSetReliabilityVal(  
    t_sessionName  
    g_value  
)  
=> t / nil
```

Description

Enables or disables the Reliability analysis in the specified session.

Arguments

<i>t_sessionName</i>	Name of the Spectre session. You can also use the <code>asiGetCurrentSession</code> function to specify the current session.
<i>g_value</i>	Boolean indicating whether you want to enable or disable reliability analysis in the specified session. When set to <code>t</code> , reliability analysis is enabled. When set to <code>nil</code> , it is disabled.

Value Returned

<code>t</code>	The reliability analysis is enabled in the specified Spectre session.
<code>nil</code>	The reliability analysis is disabled or if there is an error.

Examples

In this example, the function returns `t` because you enable the reliability analysis in the current session.

```
session = asiGetCurrentSession()  
relxSetReliabilityVal (session t)  
=> t
```

Stimulus Generation Functions

This topic lists the functions used to generate the waveform data required to preview stimuli created using the Stimuli Assignment form in ADE Explorer and ADE Assembler.

[asiStmSupportWaveformGeneration](#)

[asiStmGenerateNetlist](#)

[asiStmRunSimulation](#)

[asiStmGenerateWaveform](#)

asiStmSupportWaveformGeneration

```
asiStmSupportWaveformGeneration(  
    o_session  
)  
=> t / nil
```

Description

Determines whether the Stimuli Assignment form supports waveform preview functionality or not.

Arguments

<code>o_session</code>	The OASIS session object.

Value Returned

<code>t</code>	The integrated simulator supports waveform preview functionality in the <i>Stimuli Assignment form</i> .
<code>nil</code>	The integrated simulator does not support the waveform preview functionality in the <i>Stimuli Assignment form</i> .

asiStmGenerateNetlist

```
asiStmGenerateNetlist(  
    o_session  
    wavepath  
    nodeDesc  
    [ o_simOptions ]  
)  
=> l_netlistStatus
```

Description

Generates netlist, as part of preview waveform generation for a stimuli and saves it at the specified psf directory path.

Arguments

<i>o_session</i>	The Simulator OASIS session object.
<i>wavepath</i>	The path of the directory to create netlist.
<i>nodeDesc</i>	Stimuli input description from the OASIS stimulus session
<i>o_simOptions</i>	Optional argument that specifies the simulator options such as stopTime.

Value Returned

<i>l_netlistStatus</i>	Returns an association list of the following key/value pairs: list(list('success t/nil) list('waveName <waveform name>)

asiStmRunSimulation

```
asiStmRunSimulation(  
    o_session  
    wavepath  
    nodeDesc  
    [ o_simOptions ]  
    [ o_varDefinitions ]  
)  
=> l_simStatus
```

Description

Runs a simulation by using the netlist generated by SKILL function [asiStmGenerateNetlist](#), and writes the psf waveform data at the specified path.

Arguments

<i>o_session</i>	The Simulator OASIS session object.
<i>wavepath</i>	The psf directory path.
<i>nodeDesc</i>	Stimuli input description from the OASIS stimulus session
<i>o_simOptions</i>	Optional argument that specifies the simulator options such as stopTime
<i>o_varDefinitions</i>	Optional argument that specifies the list of variable values.

Value Returned

<i>l_simStatus</i>	Returns an association list of the following key/value pairs: <pre>list(list('success t/nil) list('waveName <name to plot> list('shortMsg "") list('logFile' <path relative to wavePath of simulation log file> list('netlist <path relative to wavePath for netlist file>)</pre>

asiStmGenerateWaveform

```
asiStmGenerateWaveform(  
    o_session  
    t_wavepath  
    t_stimuliDesc  
    [ o_simOptions ]  
    [ o_varDefinitions ]  
)  
=> l_waveStatus
```

Description

Generates preview waveforms for the specified stimuli under the given directory path.

Arguments

<i>o_session</i>	The Simulator OASIS session object.
<i>t_wavepath</i>	The psf directory path.
<i>t_stimuliDesc</i>	Stimuli input description from the OASIS stimulus session
<i>o_simOptions</i>	Optional argument that specifies the simulator options such as stopTime.
<i>o_varDefinitions</i>	Optional argument that specifies the list of variable values.

Value Returned

<i>l_waveStatus</i>	Returns an association list of the following key/value pairs: list(list('success t/nil) list('waveName <name to plot> list('shortMsg " list('logFile' <path relative to wavePath of simulation log file> list('netlist <path relative to wavePath for netlist file>)

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Miscellaneous Functions

This topic lists procedures and methods that help you integrate the simulator or customize the simulation environment.

[ahdlUpdateViewInfo](#)

[amseGeneralSetupForm](#)

[amseQuickSetupForm](#)

[amsUpdateTextviews](#)

[vmsUpdateCellViews](#)

[annRetrieveFromEffectiveCDF](#)

[artEnableAnnotationBalloon](#)

[artGenerateHierSymbolCDF](#)

[artGetCdfTargetCV](#)

[artGetCellViewDesignVarList](#)

[artCurrentInstSimName](#)

[artListToWaveform](#)

[artBlankString](#)

[artMakeString](#)

[artMakeStringPrec15](#)

[asiAddDesignVarList](#)

[asiAddVerilogArgs](#)

[asiLoadState](#)

[asiSaveState](#)

ahdlUpdateViewInfo

```
ahdlUpdateViewInfo(  
    t_lib  
    [ ?cell lt_cell ]  
    [ ?view lt_view ]  
    [ ?tool lt_tool ]  
)
```

Description

Updates cells and cellviews created with releases earlier than 4.4.2 so that the cells and cellviews can use cellview-specific parameters and parameter values. During the update, `ahdlUpdateViewInfo`, parses the Verilog-A or SpectreHDL modules that define the specified cellviews, issues any necessary error messages and updates the cellview CDF information.

Arguments

<code>t_lib</code>	The name of the library to be updated.
<code>?cell lt_cell</code>	An optional name or list of names of cells to be updated. If <code>lt_cell</code> is omitted, the function updates every <code>veriloga</code> and <code>ahdl</code> cellview in the library.
<code>?view lt_view</code>	An optional name or list of names of cellviews to be updated. If <code>lt_view</code> is omitted, the function updates every <code>veriloga</code> and <code>ahdl</code> cellview associated with the specified cell.
<code>?tool lt_tool</code>	An optional keyed argument added in order to copy the <code>termMapping</code> from the tool <code>simInfo</code> . If not specified, no copy takes place.

Value Returned

None

Examples

Updates all the `veriloga` and `ahdl` cellviews in a library.

```
ahdlUpdateViewInfo("myLibrary")
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Updates three cells in a library.

```
ahdlUpdateViewInfo("myLibrary" ?cell "res" "cmp" "opamp")
```

Updates the specified cellview.

```
ahdlUpdateViewInfo("myLibrary" ?cell "res" ?view "veriloga")
```

amseGeneralSetupForm

```
amseGeneralSetupForm(  
    o_session  
)  
=> t / nil
```

Description

Opens the General Setup form in Virtuoso AMS Designer. The General Setup form appears when you choose *Detailed Setup* and then General Setup from the AMS menu in Hierarchy Editor.

Arguments

<code>o_session</code>	The simulation session object.

Value Returned

<code>t</code>	Displays the General Setup form and returns <code>t</code> .
<code>nil</code>	Indicates an error.

Examples

Opens the General Setup form in Virtuoso AMS Designer.

```
amseGeneralSetupForm( session )  
=> t
```

amseQuickSetupForm

```
amseQuickSetupForm(  
    o_session  
)  
=> t / nil
```

Description

Opens the Quick Setup form in Virtuoso AMS Designer. The Quick Setup form appears when you choose *Quick Setup* from the AMS menu in Hierarchy Editor.

Arguments

<code>o_session</code>	The simulation session object.

Value Returned

<code>t</code>	Displays the Quick Setup form and returns <code>t</code> .
<code>nil</code>	Indicates an error.

Example

Opens the Quick Setup form in Virtuoso AMS Designer.

```
amseQuickSetupForm( session )  
=> t
```

amsOSSUnlExecutePostNetlistTrigger

```
amsOSSUnlExecutePostNetlistTrigger(  
    o_session  
)
```

Description

This is a callback function that gets triggered after AMS UNL netlisting.

Arguments

<i>o_session</i>	The OASIS session object.

Value Returned

None

Example

Returns the path to the netlist directory once the netlisting is finished.

```
defun(amsOSSUnlExecutePostNetlistTrigger (session)  
    netlistDir = asiGetNetlistDir(session)  
)
```


amsUpdateTextviews

```
amsUpdateTextviews (
    t_libName
    [ ?cell t_cellName ]
    [ ?view t_viewName ]
    [ ?incremental g_incremental ]
)
=> t / nil
```

Description

Creates a Virtuoso database, depending on the arguments passed for the following:

- For all the text views in the library.
- For all the text views of the specified cell in the library.
- For all the text views in the library that have the specified view name.
- For the specified text view, given that the view is a text view.
- For all the text views in the configuration, given that the specified view is a config view.
- For all the text views that do not have an existing Virtuoso database or have a database with an older timestamp than the specified text view.

if you have text views in read-only libraries, you must set an environment variable in .cdsinit file. It specifies the directory where the Virtuoso database for such text views must be created. You should set the environment variable as shown below:

```
envSetVal("ams.netlisterOpts" "amsTempDirForShadows" 'string'
"<pathToDirectory>")
```

Automatic creation of the Virtuoso database for read-only Verilog-A and VHDL-AMS text views is not supported.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<code>t_lib</code>	A string, which is the name of the library to be processed.
<code>?cell t_cellName</code>	A string, which is the name of the cell to be processed.
<code>?view t_viewName</code>	A string, which is the name of the view to be processed.
<code>?incremental lg_incremental</code>	If set to <code>t</code> , the Virtuoso database is created only for the text views that do not have an existing Virtuoso database or have a database with an older timestamp than the text view. If set to <code>nil</code> , the database is created for all the text views. The default value is <code>t</code> .

Value Returned

<code>t</code>	Successful creation of the Virtuoso database.
<code>nil</code>	Failed to create the Virtuoso database.

Examples

Creates a Virtuoso database for all the text views in myLib:

```
amsUpdateTextviews("myLib")
```

Creates a Virtuoso database for all the text views of mycell in mylib:

```
amsUpdateTextviews("myLib" ?cellName "mycell" )
```

Creates a Virtuoso database for all the text views in the config view of mycell in mylib:

```
amsUpdateTextviews("myLib" ?cellName "mycell" ?viewName "config" ?incremental nil  
)
```

vmsUpdateCellViews

```
vmsUpdateCellViews(  
    [ ?lib t_lib ]  
    [ ?cell t_cell ]  
    [ ?view t_view ]  
    [ ?viewt t_viewType ]  
)  
=> t / nil
```

Description

Updates AMS Designer information with the current state of verilog, systemVerilog, veriloga, verilogams and vhdl text views. You might use this function, for example, when you have updated a Verilog-AMS source file outside of the AMS Designer environment. You might also use it when you receive a Verilog-AMS library in a single source file, bring it into the Library.Cell:View structure using `xmvlog -use5x`, and then need to prepare the library for use in the AMS Designer environment.

Note: If you run this function without any arguments, a pop-up appears asking for lib/cell/view and viewType information.

Argument

<code>?lib t_lib</code>	A string, which is the name of a library or a list of library names to look in for cellviews to update. If this argument is not specified (with just " ") or is specified as <code>nil</code> , all libraries defined in the <code>cds.lib</code> file are searched.				
<code>?cell t_cell</code>	A string, which is the name of a cell or a list of cell names to be searched for update in the libraries. If this argument is not specified (with just " ") or is specified as <code>nil</code> , all cells are searched.				
<code>?view t_view</code>	A string, which is the name of a cellview or a list of cellview names to be searched for update. If this argument is not specified (with just "") or is specified as <code>nil</code> , all views are searched.				
<code>?viewt t_viewType</code>	The type of view that you want to update. Valid values: <table><tr><td><code>text.ahdl</code></td><td>Analog HDL text view</td></tr><tr><td><code>text.veriloga</code></td><td>Verilog-A text view</td></tr></table>	<code>text.ahdl</code>	Analog HDL text view	<code>text.veriloga</code>	Verilog-A text view
<code>text.ahdl</code>	Analog HDL text view				
<code>text.veriloga</code>	Verilog-A text view				

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

VHDLAMSText	VHDL-AMS text view
vhdl	VHDL text view
text.v	Verilog text view
VerilogAMSText	Verilog-AMS text view
systemVerilogText	SystemVerilog text view

Value Returned

t	The function runs successfully.
nil	Returns <code>nil</code> when: ■ The function runs unsuccessfully ■ The individual shadow generation results of any of the views return <code>nil</code>. ■ The view is of an unsupported type. ■ The source file corresponding to a cellview is read-only.

Example

This example updates the specified text cellview.

```
vmsUpdateCellViews(?lib "myLib" ?cell "myCell" ?view "verilogAMS" ?viewt  
"VerilogAMSText" )
```

The next example updates verilogAMS views in all the cells in the myLib library.

```
vmsUpdateCellViews(?lib "myLib" ?view "verilogAMS" ?viewt "VerilogAMSText" )
```

annRetrieveFromEffectiveCDF

```
annRetrieveFromEffectiveCDF(  
    [ l_window ]  
)  
=> l_WindowList / nil
```

Description

Retrieves the data from effective CDF. This function is used to match the CDF properties of the graphical window with the global effective CDF properties and needs to be called before annotating CDF properties to reflect the changes done in effective CDF properties.

Arguments

<i>l_window</i>	A graphical Window ID or a list of window IDs on which you want to retrieve the effective CDF properties. When you do not specify this argument, it includes the list of all opened windows for current the current session.

Value Returned

<i>l_windowList</i>	List of windows when the function runs successfully.
<i>nil</i>	Indicates an error.

Example

Suppose you have schematic window (window ID 2) and layout window (window ID 3) open. By default, both of them are annotating `netName`. Now, toggle display to Pin Name under Terminals (`cdsTerm`) sub-tab in the Interpreted Labels tab within CDF GUI for any schematic component, such as `nmos`.

```
window = setof(win hiGetWindowList() geIsGraphicalWindow(win))  
window ==> (window:2 window:3)  
annRetrieveFromEffectiveCDF(window)
```

Now, if you redraw both the layout and schematic, these views will annotate Pin name for the specified component.

artEnableAnnotationBalloon

```
artEnableAnnotationBalloon(  
    g_value  
    [ x_firstPoint ]  
    [ x_lastPoint ]  
)  
=> t / nil
```

Description

Enables or disables the display of parametric sweep results annotated on the schematic. When enabled, the first six result points are displayed in a pop-up window that appears when you hover the mouse pointer over an instance on the schematic. Use *x_firstPoint* and *x_lastPoint* to specify a range of result points to be displayed in the pop-up window.

Arguments

<i>g_value</i>	Enables or disables the pop-up window that displays parametric sweep results annotated on the schematic. Valid Values: <i>t</i> enables the pop-up window, <i>nil</i> disables the pop-up window. Default Value: <i>nil</i>
<i>x_firstPoint</i>	First result point to be displayed in the pop-up window.
<i>x_lastPoint</i>	Last result point to be displayed in the pop-up window. Ensure that the number of result points between <i>x_firstPoint</i> and <i>x_lastPoint</i> does not exceed six result points because the pop-up window will display only six result points.

Value Returned

<i>t</i>	Displays the pop-up window.
<i>nil</i>	Indicates an error.

Examples

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Enables the display of parametric sweep results in a pop-up window that appears when you hover the mouse pointer over an instance on the schematic.

```
artEnableAnnotationBalloon(t)
```

Displays the 15th result point in a pop-up window, when you hover the mouse pointer over an instance on the schematic.

```
artEnableAnnotationBalloon(t 15)
```

Displays the 10th to the 15th result points in a pop-up window, when you hover the mouse pointer over an instance on the schematic.

```
artEnableAnnotationBalloon(t 10 15)
```

Disables the display of parametric sweep results in a pop-up window.

```
artEnableAnnotationBalloon(nil)
```

artGenerateHierSymbolCDF

```
artGenerateHierSymbolCDF(  
    d_cellView  
    [ g_overwrite ]  
)  
=> t
```

Description

Creates the cell CDF for the specified cellView, in the same way as happens when you create a symbol from a schematic in Composer. The cellView will be examined for any use of `pPar()` in expressions, and the corresponding parameters will be added to the CDF if they are not already present. Whilst both schematic and symbol cellViews may be passed to this function, it is usually best to pass a schematic cellView, in order to get the `pPar()`s used into the CDF.

Arguments

<code>d_cellView</code>	Database object for the schematic or symbol.
<code>g_overwrite</code>	If the cell CDF already exists, a popup will normally be displayed to ask if you want to overwrite the CDF. If the overwrite argument is specified as non-nil, then the popup is suppressed, and the CDF will be overwritten.

Value Returned

<code>t</code>	Successful operation

Examples

Recreates the cell CDF for the current cellView.

```
artGenerateHierSymbolCDF(getEditCellView())
```


artGetCdfTargetCV

```
artGetCdfTargetCV(  
    )  
=> dbobject / nil
```

Description

This procedure returns the cell view *dbobject* that is the target of the update instance form or the create instance form. The function requires that either the arm property (`armGet 'cdfInfo 'currentCellView`) is set to a *cellView*, or that the variable *cdfgForm* is bound to a form with a property *cellViewId* attached (`getq cdfgForm cellViewId`). If neither of these is bound, *nil* is returned.

Arguments

None

Value Returned

<i>dbobject</i>	Database object
<i>nil</i>	Indicates an error

Examples

```
artGetCdfTargetCV() => db:28458028  
artGetCdfTargetCV() => nil
```

artGetCellViewDesignVarList

```
artGetCellViewDesignVarList(  
    d_cellViewId  
)  
=> l_nameValuePair
```

Description

Returns the list of design variable name value pairs associated with the top level cell view.

Arguments

d_cellViewId ID of a top level cellview.

Value Returned

l_nameValuePair List of name value pairs.

Example

```
artGetCellViewDesignVarList(asiGetTopCellView(asiGetCurrentSession())) =>  
(("Rp" "5")  
  ("Q" "50")  
  ("w" "2*3.14159*1.96e9")  
  ("Co" "33p")  
  ("Lb" "2.2n")  
)
```

artCurrentInstSimName

```
artCurrentInstSimName (  
    )  
    => t_extName
```

Description

This function provides the external instance name for the current instance being formatted in the netlister. The function should only be used for socket netlisters or custom netlist procedures for socket. The routine does not take into consideration case mapping registered for the tool or the prefix for the instance. It does take care of the netlisting mode (FNL, HNL or IHNL).

Arguments

None

Value Returned

<i>t_extName</i>	External name of current instance.

Example

```
artCurrentInstSimName ()
```

artListToWaveform

```
artListToWaveform(  
    l_xyPairs  
)  
=> o_waveform
```

Description

This function takes a list of X,Y points and translates it into a waveform. Each of the X,Y points is also in a list format. If Y points are represented in a list format, they are treated as complex numbers while creating the waveform.

Argument

<i>l_xyPairs</i>	X, Y points in list format.

Value Returned

<i>o_waveform</i>	Waveform

Example

This waveform contains {1,3,5} as X and {2,4,6} as Y points.

```
artListToWaveform( ' ( ( 1 2 ) ( 3 4 ) ( 5 6 ) ) ) => o_waveform
```

artBlankString

```
artBlankString(  
    g_value  
)  
=> t / nil
```

Description

Returns `true` if the given object is equal to `nil`. If the given object is a string, checks if the string is empty or has blank space characters only and returns `true`. If the object is not `nil` and if the string has non-space characters, it returns `nil`.

Argument

<i>g_value</i>	Data object.

Value Returned

<code>t</code>	If the given object is <code>nil</code> and also when the object is an empty string or has blank space characters only.
<code>nil</code>	Indicates an error.

Examples

```
artBlankString("") => t  
artBlankString("h") => nil  
artBlankString(nil) => t
```

artMakeString

```
artMakeString(  
    g_anyArg  
)  
=> t_argAsString / nil
```

Description

Converts data of the given data type to a string. The valid data types for the *g_anyArg* argument include symbol, integer, float, and string. The floating point numbers are converted into strings by using the '%.16g' format specification to produce the most precise output.

Argument

<i>g_anyArg</i>	Argument of any data type.

Value Returned

<i>t_argAsString</i>	Returns the string representation of the input argument.
<i>nil</i>	Indicates an error.

Examples

- With symbol data:

```
artMakeString( '\1\2\3' ) => "123"
```
- With integer data:

```
artMakeString( 2 ) => "2"
```
- With float/real data:

```
artMakeString( 2.999 ) => "2.999"  
artMakeString( 1e-9 ) => "1e-9"
```
- With list data:

```
artMakeString( '(1 a 2.0)' ) => "(1 a 2.0)"
```
- With CDBA inst data:

```
inst = car( geGetSelSet() )  
artMakeString( inst ) => "db:123456"
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

■ With other data types:

```
artMakeString( nil ) => "nil"  
artMakeString( t ) => "t"
```

artMakeStringPrec15

```
artMakeStringPrec15(  
    g_anyArg  
)  
=> t_argAsString / nil
```

Description

Converts data of the given data type to a string. The valid data types for the *g_anyArg* argument include symbol, integer, float, and string. It is similar to the function `artMakeString`, except that it uses the `'%.15g'` format specification to convert the floating point numbers into strings.

Argument

<i>g_anyArg</i>	Argument of any data type.

Value Returned

<i>t_argAsString</i>	Returns the string representation of the input argument.
<i>nil</i>	Indicates an error.

Examples

```
artMakeStringPrec15( 2.0 ) => "2.0"  
artMakeStringPrec15( 10p ) => "1e-11"
```


asiAddDesignVarList

```
asiAddDesignVarList(  
    o_session  
    l_designVarList  
)  
=> l_newDesignVarList
```

Description

Adds a list of variables to the existing session design variable list.

Arguments

<i>o_session</i>	Simulation session object.
<i>l_designVarList</i>	List of design variables to add to the session design variable list.

Value Returned

<i>l_newDesignVarList</i>	Returns the new list of design variables.

Example

Creates a list of name value pairs to add to the list of design variables.

```
asiAddDesignVarList( session ' ("rbias" "1k") ("ccap" "1n") ("area" "16u"))
```

asiAddVerilogArgs

```
asiAddVerilogArgs (
    o_session
    t_verilogArg
)
=> t_verilogArg
```

Description

For a mixed-signal simulation, this method lets you change the list of arguments that are sent to the Verilog-XL simulator with the *-slave* option.

These arguments are the options from the Verilog Options form. Typically, you use this function to add the *+vmxcconfig.vmx* option.

Note for Integrators: This function is defined as a method. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Simulation session object.
<i>t_verilogArg</i>	Arguments (or options) to send to the Verilog-XL simulator with the <i>-slave</i> option.

Value Returned

<i>t_verilogArg</i>	Returns the arguments (or options) that are sent to Verilog-XL with the <i>-slave</i> option.

Examples

```
defmethod( asiAddVerilogArgs ( ( session XYZ_session ) verilogArg )
    sprintf( verilogArg "%s +vmxcconfig.vmx" verilogArg )
)
```

Adds *+vmxcconfig.vmx* to the list of Verilog-XL arguments for the XYZ simulator class.

asiGetDataReaderVersion

```
asiGetDataReaderVersion(  
    t_format  
)  
=> t_formatVersion
```

Description

Returns the current version of data reader for the specified format.

Arguments

<i>t_format</i>	The format of data reader

Value Returned

<i>t_formatVersion</i>	Current version for the specified format of data reader

Examples

The following example returns the current version of data reader for the format fast-signal database (`fsdb`).

```
printf("The current version of data reader for the fsdb format is: %s\n"  
asiGetDataReaderVersion("fsdb"))  
=> The current version of data reader for the fsdb format is: 6.3  
=> t
```

asiLoadState

```
asiLoadState(  
    o_session  
    [ ?name t_name ]  
    [ ?option s_option ]  
    [ ?stateDir t_stateDir ]  
    [ ?lib t_lib ]  
    [ ?cell t_cell ]  
    [ ?simulator t_simulator ]  
)  
=> t / nil
```

Description

Loads a saved state into the current simulation environment directly from the CIW without displaying the *Loading State* form.

Arguments

<code>o_session</code>	Simulation session object.
<code>?name t_name</code>	Name of the state.
<code>?option s_option</code>	Option to specify whether the state is to be loaded from 'dir or 'cellview.
<code>?stateDir t_stateDir</code>	Directory in which the state is saved. The complete location of the saved state would be <code>stateDir/lib/cell/simulator</code> . Specify <code>stateDir</code> only when <code>option</code> is set to 'dir.
<code>?lib t_lib</code>	Library name.
<code>?cell t_cell</code>	Cell name.
<code>?simulator t_simulator</code>	Name of the simulator. Specify this when <code>option</code> is set to 'dir.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

Examples

Loads `MyState` directly from the CIW given the directory (`~/ .artist_state`) in which the state is saved and the session (`session1`) into which the state is to be loaded.

```
asiLoadState( session1 ?name "MyState" ?option 'dir ?stateDir "~/ .artist_state" )
```

Loads `spectre_state1` directly from the CIW given the library/cell path (`myLib/myCell`) and the session (`session2`) into which the state is to be loaded.

```
asiLoadState( session2 ?name "spectre_state1" ?option 'cellview ?lib "myLib" ?cell  
"myCell" )
```

asiSaveState

```
asiSaveState(  
    o_session  
    [ ?name t_name ]  
    [ ?option s_option ]  
    [ ?stateDir t_stateDir ]  
    [ ?lib t_lib ]  
    [ ?cell t_cell ]  
    [ ?simulator t_simulator ]  
    [ ?description t_description ]  
)  
=> t / nil
```

Description

Saves the current state of the current simulation environment directly from the CIW without displaying the *Saving State* form.

Arguments

<code>o_session</code>	Simulation session object.
<code>?name t_name</code>	Name of the state.
<code>?option s_option</code>	Option to specify whether the state is to be saved in 'dir or 'cellview.
<code>?stateDir t_stateDir</code>	Directory in which the state is to be saved. The complete location of the saved state would be <i>stateDir/lib/cell/simulator</i> . Specify <i>stateDir</i> only when <i>option</i> is set to 'dir.
<code>?lib t_lib</code>	Library name.
<code>?cell t_cell</code>	Cell name.
<code>?simulator t_simulator</code>	Name of the simulator. Specify this when <i>option</i> is set to 'dir.
<code>?description t_description</code>	Description to be saved with the state.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

Examples

Saves the state `NewState` directly from the CIW given the directory (`~/ .artist_state`) and session (`session1`) in which the state is to be saved and the description (`saved on Jan 1, 2006`) to be saved with the state.

```
asiSaveState( session1 ?name "NewState" ?option 'dir ?stateDir "~/ .artist_state"
?description "saved on Jan 1, 2006" )
```

asiCheck

```
asiCheck(  
    o_analysis | o_anaOption | o_envOption | o_simOption | o_keepOption  
    r_form  
)  
=> t / nil
```

Description

Called by the environment to check values in the analysis and option fields.

Note for Integrators: This procedure is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_analysis</i>	Specifies the analysis object.
<i>o_anaOption</i>	Specifies the analysis option object.
<i>o_envOption</i>	Specifies the environment option object.
<i>o_simOption</i>	Specifies the simulator option object.
<i>o_keepOption</i>	Specifies the keep option object.
<i>r_form</i>	Form containing the analysis or option fields. This argument is passed into the <code>asiCheck</code> method by the environment.

Value Returned

<i>t</i>	The field values are in the correct range.
<i>nil</i>	Indicates an error.

Examples

Checks the values of the form fields in an ac analysis.

```
defmethod( asiCheck ( ( ana analog_ac_analysis ) form )  
  
    asiCheckBlankNumericGreater( ana form 'from 0 )  
    asiCheckBlankNumericGreater( ana form 'to 0 )
```


Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
if( equal( asiGetAnalysisFormFieldVal( form 'ac 'incrType )
    "Linear" ) then asiCheckBlankNumericGreater( ana form
    'lin 0 )
else
    asiCheckBlankNumericGreater( ana form 'log 0 )
)

asiCheckMultipleGreater( ana form 'to 'from )

)
```

Checks whether the ABSTOL simulator option for Spectre has a value greater than zero. The Spectre simulator options class must be defined before the `asiCheck` function is used.

```
defclass( spectre_simOption ( analog_simOption ) ( ) )

defmethod( asiCheck ( ( option spectre_simOption ) form )
    asiCheckBlankNumericGreater( option form 'ABSTOL 0 )
)
```

Note: If you want to write methods for analyses, analysis options, environment options, simulator options, or keep options, you must define the appropriate class in your `classes.il` file.

asiCheckDesignVariable

```
asiCheckDesignVariable(  
    o_obj  
    r_form  
    s_fieldName  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a valid design variable.

asiCheckExpression

```
asiCheckExpression(  
    o_obj  
    r_form  
    s_fieldName  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a valid expression.

asiCheckExpressionGreater

```
asiCheckExpressionGreater(  
    o_obj  
    r_form  
    s_fieldName  
    g_value  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is an expression that evaluates to a value greater than *g_value*.

asiCheckBlankNumeric

```
asiCheckBlankNumeric(  
    o_obj  
    r_form  
    s_fieldName  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a numeric value.

asiCheckBlankNumericGreater

```
asiCheckBlankNumericGreater(  
    o_obj  
    r_form  
    s_fieldName  
    g_value  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a numeric value greater than *g_value*.

asiCheckBlankNumericNequal

```
asiCheckBlankNumericNequal(  
    o_obj  
    r_form  
    s_fieldName  
    g_value  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a numeric value not equal to *g_value*.

asiCheckBlankNetExists

```
asiCheckBlankNetExists(  
    o_obj  
    r_form  
    s_fieldName  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a valid net name.

asiCheckBlankInstExists

```
asiCheckBlankInstExists(  
    o_obj  
    r_form  
    s_fieldName  
    [ ?subAnaMsg t_subAnaMsg ]  
)  
=> t / nil
```

Description

Verifies that the *s_fieldName* entry is a valid instance name.

asiCheckMultipleGreater

```
asiCheckMultipleGreater(  
    o_obj  
    r_form  
    s_largerField  
    s_smallerField  
)  
=> t / nil
```

Description

Verifies that the value of the *s_largerField* entry is greater than that of the *s_smallerField* entry.

asiCheckSimulationSuccess

```
asiCheckSimulationSuccess(  
    o_session  
)  
=> t / nil
```

Description

Determines if a simulation was successful.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Argument

<i>o_session</i>	Simulation session object.

Value Returned

<i>t</i>	Successful operation.
<i>nil</i>	Indicates an error.

Examples

```
success = asiCheckSimulationSuccess( session )
```

Sets *success* to *t* if the simulation is successful. Sets *success* to *nil* otherwise.

```
defmethod( asiCheckSimulationSuccess (( session  
    <yourSimulator>_session))  
    let( ( success )  
    ;; This checks for the existence of the PSF file.  
    success = callNextMethod()  
    <insert code to perform simulator-specific checking>  
    success  
    )  
)
```

asiCreateLogFileVerilog

```
asiCreateLogFileVerilog(  
    o_session  
    l_entryList  
)
```

Description

Creates the file `logFileVerilog` in the `.../psf` directory to indicate the analysis result of Verilog-XL in a mixed-signal transient run.

Arguments

<i>o_session</i>	Specifies the session object.
<i>l_entryList</i>	Must be in the form. ■ <code>nil</code>—If there is no transient analysis. ■ <code>'(("<transient-analysis-instance-name>" "<transient-analysis-type-name>"))</code>—If there is a transient analysis and digital results are available.

Value Returned

None

Examples

For Spectre:

```
asiCreateLogFileVerilog( o_session '("timeSweep-tran" "tran")) )
```

When transient analysis is not performed or when no digital results are saved for Verilog:

```
asiCreateLogFileVerilog( o_session nil)
```

asiDcStore

```
asiDcStore(  
    o_session  
    t_fileName  
)  
=> t / nil
```

Description

Copies the DC node voltages in *processId*.dc to the name you pass in *fileName*. This function assumes that your simulator writes the DC node voltages to *netlistDir*/raw/*processId*.dc.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.
<i>t_fileName</i>	Name of the file in which to store the DC node voltages.

Value Returned

<i>t</i>	The DC voltages are stored in <i>fileName</i> .
<i>nil</i>	Indicates an error.

Examples

```
defmethod( asiDcStore ( ( session XYZ_session ) fileName )  
    let( ( netlistDir processId str )  
        netlistDir = asiGetNetlistDir(session)  
        processId = asiGetSimProcessId(session)  
        str = strcat( "/bin/cp " netlistDir "/raw/" processId ".dc "  
            fileName )  
        system( str )  
        t  
    )  
)
```

asiGetCurrentSession

```
asiGetCurrentSession(  
    )  
=> o_session / nil
```

Description

Returns the session object for the current session.

Use this procedure only from within menu callbacks to get the current session.

Argument

None

Value Returned

<i>o_session</i>	Returns the session object for the current session.
<i>nil</i>	There is no current session defined.

asiGetDesignVarList

```
asiGetDesignVarList(  
    o_session  
)  
=> l_designVarList / nil
```

Description

Gets the list of design variables for the design associated with the session you specify.

Argument

<i>o_session</i>	Simulation session object.

Value Returned

<i>l_designVarList</i>	Returns the list of design variables.
<i>nil</i>	There are no design variables associated with the specified session.

Examples

```
designVarList = asiGetDesignVarList( session )
```

Returns the list of design variables in `designVarList`.

asiGetFormFieldChoices

```
asiGetFormFieldChoices(  
    r_form  
    s_fieldName  
)  
=> l_choices / nil
```

Description

Gets the list of choices for a form field that is set up as a list box.

Arguments

<i>r_form</i>	Form containing the field. The form can be one of the following: ■ Environment options ■ Simulator options ■ Keep options ■ Analysis options
<i>s_fieldName</i>	Name of a field of the type <code>listBox</code> .

Value Returned

<i>l_choices</i>	Returns the list of choices for the field.
<code>nil</code>	The field cannot be accessed.

Examples

To get the list of choices for a field on the Choosing Analyses form, see the `asiGetAnalysisFormFieldChoices` function.

asiGetFormFieldVal

```
asiGetFormFieldVal(  
    r_form  
    s_fieldName  
)  
=> g_value / nil
```

Description

Gets the value of a field on a form.

Arguments

<i>r_form</i>	Form containing the field. The form can be one of the following: ■ Environment options ■ Simulator options ■ Keep options ■ Analysis options
<i>s_fieldName</i>	Name of the field.

Value Returned

<i>g_value</i>	Returns the value of the field.
<i>nil</i>	The field cannot be accessed.

Examples

```
asiGetFormFieldVal( form 'RELTOL )
```

Gets the value of the RELTOL field on the Simulator Options form.

To get the value of a field on the Choosing Analyses form, see the `asiGetAnalysisFormFieldVal` function.

asiGetKeepList

```
asiGetKeepList(  
    o_session  
)  
=> l_keepList / nil
```

Description

Gets a list of signals and currents that are saved during simulation.

Argument

<i>o_session</i>	Simulation session object.

Value Returned

<i>l_keepList</i>	Returns the list of signals and currents to be saved during simulation.
<i>nil</i>	Returns <i>nil</i> if no signals were specified to be kept during simulation.

Example

Returns the list of signals and currents for a given session in *keepList*.

```
keepList = asiGetKeepList( session )
```

asiGetLogFileList

```
asiGetLogFileList(  
    o_session  
)  
=> l_logFiles / nil
```

Description

Returns a list of the names of the log files.

Note for Integrators: This procedure is defined as a method for the Analog Class. You can overload this method to return your list of log files, as shown in the example for integrators.

Argument

<i>o_session</i>	Session object.

Value Returned

<i>l_logFiles</i>	Returns the list of log files. ■ For analog-only simulation, the following list is returned: ("logFile") ■ For mixed-signal simulation, the following list is returned: ("logFile" "logFileVerilog")
<i>nil</i>	There is an error.

Examples

```
asiGetLogFileList( asiGetCurrentSession() )  
=> ( "logFile" )
```

Returns the list of log files for the session.

Example for Integrators

```
defmethod( asiGetLogFileList ( ( _session <yourSimulator>_session ) )  
    '( "<yourSimulator>logFile" )  
)
```

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Overloads the `asiGetLogFileList` function to return the log file for your simulator.

asiGetMarchList

```
asiGetMarchList(  
    o_session  
)  
=> l_marchList / nil
```

Description

Returns a list of signals that are to be marched during simulation.

Argument

<i>o_session</i>	Simulation session object.

Value Returned

<i>l_marchList</i>	Returns the list of signals to march during simulation.
<i>nil</i>	Returns <i>nil</i> if no signals were specified to be marched.

Example

```
marchList = asiGetMarchList( session )
```

Returns the list of march signals for the session in `marchList`.

asiGetNetlistDir

```
asiGetNetlistDir(  
    o_session  
)  
=> t_netlistDir
```

Description

Returns the netlist directory. If the directory does not exist, this function creates it.

Argument

<i>o_session</i>	Simulation session object.

Value Returned

<i>t_netlistDir</i>	Returns the path name for the netlist directory.

Example

```
netlistDir=asiGetNetlistDir(session)
```

Returns the netlist directory.

asiGetOutputList

```
asiGetOutputList(  
    o_session  
)  
=> o_outputsList / nil
```

Description

Returns a list of structures. Each structure defines a output to be saved and/or plotted after simulation.

Argument

<i>o_session</i>	Specifies an analysis object.

Value Returned

<i>o_outputsList</i>	Returns a list of structures. Each structure defines one output.
<i>nil</i>	There was an error.

Example

After opening an ADE session and setting up outputs, in the CIW:

```
s=asiGetCurrentSession()  
=>stdobj@0x1d14060  
outs=asiGetOutputList(s)  
=>(sevOutputStruct@0x15e71b8 sevOutputStruct@0x15e7208 sevOutputStruct@0x153ef48)  
outs~>signal  
=>("/out" "/net15" nil)  
outs~>expression  
=>(nil nil ymax(VT("/net15")))
```

asiGetPlotList

```
asiGetPlotList(  
    o_session  
)  
=> l_plotList / nil
```

Description

Gets the list of signals that can be plotted for the current simulation session.

Argument

<i>o_session</i>	Simulation session object.

Value Returned

<i>l_plotList</i>	Returns the list of the items that can be plotted.
<i>nil</i>	Returns <i>nil</i> if no signals were specified to be plotted.

Example

```
plotList = asiGetPlotList( session )
```

Returns the list of items that can be plotted for the given session in `plotList`.

asiGetPsfDir

```
asiGetPsfDir(  
    o_session  
)  
=> t_dirName / nil
```

Description

Returns the name of the PSF directory.

Argument

<i>o_session</i>	Session object.

Value Returned

<i>t_dirName</i>	Returns the name of the PSF directory.
<i>nil</i>	There is an error.

Example

```
psfDir = asiGetPsfDir( session )
```

Returns the PSF directory.

asiGetSession

```
asiGetSession(  
    { x_id | w_window | r_form | s_name | o_analysis }  
)  
=> o_session / nil
```

Description

Returns the session object given one of five possible identifiers.

Arguments

<i>x_id</i>	Integer identifier for the session.
<i>w_window</i>	Encapsulation window for the session.
<i>r_form</i>	Form for the session, which must be one of the following: ■ Analyses ■ Environment options ■ Simulator options ■ Keep options ■ Analysis options
<i>s_name</i>	Symbolic name for the session, for example, <i>spectre0</i> .
<i>o_analysis</i>	Analysis object.

Value Returned

<i>o_session</i>	Returns the session object if successful.
<i>nil</i>	Indicates an error.

Example

```
session = asiGetSession( 'spectre0 '
```

Returns the session object for the *spectre0* session.

asiGetSimName

```
asiGetSimName(  
    { o_tool | o_session }  
)  
=> t_simulatorName
```

Description

Gets the name of the simulator for a tool or session object.

Arguments

<i>o_tool</i>	Simulation tool object.
<i>o_session</i>	Simulation session object.

Value Returned

<i>t_simulatorName</i>	Returns the name of the simulator.

Example

```
simulatorName = asiGetSimName( session )
```

Returns the simulator name for the given session in `simulatorName`.

asiGetTool

```
asiGetTool(  
    { t_toolName | o_session }  
)  
=> o_tool
```

Description

Returns the tool object associated with the specified tool name or session. If the tool object is not found, an attempt is made to create and initialize the tool.

Arguments

<i>ts_toolName</i>	Name of the tool.
<i>o_session</i>	Session object.

Value Returned

<i>o_tool</i>	Returns the tool object.

Example

```
tool = asiGetTool( 'XYZ' )
```

Returns the tool object for the XYZ simulator.

asiGetTopCellView

```
asiGetTopCellView(  
    o_session  
)  
=> d_cellView / nil
```

Description

Returns the top-level cellview associated with the session.

Arguments

<i>o_session</i>	Session object.

Value Returned

<i>d_cellView</i>	Returns the top-level cellview.
<i>nil</i>	There is an error.

Example

```
cv = asiGetTopCellView( session )  
lib = cv~>libName  
cell = cv~>cellName  
view = cv~>viewName
```

Returns the library, cell, and view names.

asiSendSim

```
asiSendSim(  
    o_session  
    t_command  
    t_callback  
    g_param  
    g_saveOutput  
)  
=> t
```

Description

Sends a command to Cadence SPICE to forward to the target simulator.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Arguments

<i>o_session</i>	Session object.
<i>t_command</i>	Command (string) to send to Cadence SPICE.
<i>t_callback</i>	Routine to call when this command terminates. Frequently, this argument is set to <code>nil</code>. Callback parameter list: (<i>l_list</i> <i>g_param</i>) <i>l_list</i> is a list of strings (one string per line) that are generated by Cadence SPICE as output from <i>t_command</i>. <i>l_list</i> is generated only if <i>g_saveOutput</i> is non-<code>nil</code>.
<i>g_param</i>	Parameter for the callback routine. Frequently, this argument is set to <code>nil</code> .
<i>g_saveOutput</i>	Boolean flag that specifies whether or not the output from the command you sent to Cadence SPICE is saved in <i>l_list</i> . If you specify this argument as non- <code>nil</code> , the output is saved in <i>l_list</i> ; otherwise, the output is not saved.

Value Returned

<i>t</i>	Returns <i>t</i> when the command is sent to Cadence SPICE.

Example

```
asiSendSim( session "restore off" nil nil nil )
```

Sends the "restore off" command to Cadence SPICE to be forwarded to the target simulator.

asiSetDesignVarList

```
asiSetDesignVarList(  
    o_session  
    l_designVarList  
)  
=> l_designVarList
```

Description

Sets the design variable list for a session.

This function is used to set the design variable list (a list of (*varName*, *varVal*) pairs) for a session, stripping of any invalid *varName* (the one that does not match the regular expression `^[a-zA-Z_][a-zA-Z_0-9]*$`).

Arguments

<i>o_session</i>	Simulator session object.
<i>l_designVarList</i>	List of design variables you want to use. The list should be comprised of one or more 2 element sub-lists of the following format: ("variableName" "value") If the list is 'nil', all design variables will be removed from the current session.

Value Returned

<i>l_designVarList</i>	Returns the list of design variables for the session.

Example

```
varList = '(("var1" "100") ("var2" "ABC"))  
asiSetDesignVarList(asiGetCurrentSession() varList)  
=> ("var2" "ABC") ("var1" "100")
```

Here, *var1* & *var2* are valid variable names starting with a-z or A-Z characters.

If you want add more design variables you can use `asiAddDesignVarList`:

```
varList = '("lvar1" "100") ("var3" "ABC")
```


Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

```
asiAddDesignVarList(asiGetCurrentSession() varList)
=>*Error* asiSetDesignVarList: bad name "lvar1", not added
=> (("var3" "ABC") ("var2" "ABC") ("var1" "100"))
```

Here `var3` is a valid variable name and therefore gets added to the design variable list, but `lvar1` is discarded as it is not a valid variable name (not starting with a-z or A-Z character).

asiSetFormFieldChoices

```
asiSetFormFieldChoices(  
    r_form  
    s_fieldName  
    l_choices  
)  
=> l_choices / nil
```

Description

Sets the list of choices to appear in the list box for the specified form field.

Arguments

<i>r_form</i>	Form containing the field. The form can be one of the following: ■ Environment options ■ Simulator options ■ Keep options ■ Analysis options
<i>s_fieldName</i>	Name of a field of the type <code>listBox</code> .
<i>l_choices</i>	List of choices for the field.

Value Returned

<i>l_choices</i>	Returns the new list of choices for the field.
<i>nil</i>	The field cannot be accessed.

Related Function

To set the list of choices for a field on the Choosing Analyses form, see the `asiSetAnalysisFormFieldChoices` function.

asiSetFormFieldVal

```
asiSetFormFieldVal(  
    r_form  
    s_fieldName  
    g_value  
)  
=> g_value / nil
```

Description

Sets the value of a field on a form.

Arguments

<i>r_form</i>	Form containing the field. The form can be one of the following: ■ Environment options ■ Simulator options ■ Keep options ■ Analysis options
<i>s_fieldName</i>	Name of the field.
<i>g_value</i>	Value for the field.

Value Returned

<i>g_value</i>	Returns the new value for the field.
<i>nil</i>	The field cannot be accessed.

Example

```
asiSetFormFieldVal( form 'RELTOL 1e-2 )
```

Sets the RELTOL field on the Simulator Options form to 1e-2.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Related Function

To set the value of a field on the Choosing Analyses form, see the `asiSetAnalysisFieldVal` function.

asiSetKeepList

```
asiSetKeepList(  
    o_session  
    l_KeepList  
)  
=> l_KeepList
```

Description

Sets the list of the specific signals and currents to save during simulation.

Note: If you want to save *all* the outputs of a particular type, refer to.

Arguments

<i>o_session</i>	Simulation session object.
<i>l_KeepList</i>	This is a list of selected signals (nets) or terminals comprising of the following information: window_id libName CellName ViewName

Value Returned

<i>l_KeepList</i>	Returns the list of signals and currents to be saved during simulation.

Example

```
asiSetKeepList( session myKeepList )
```

Sets *myKeepList* as the list of signals to be saved for the given session.

asiSetMarchList

```
asiSetMarchList(  
    o_session  
    l_MarchList  
)  
=> l_MarchList
```

Description

Sets the list of signals to march during simulation.

Arguments

<i>o_session</i>	Simulation session object.
<i>l_MarchList</i>	List of signals to march during simulation. This is a list of selected signals (nets) or terminals comprising of the following information: window_id libName CellName ViewName

Value Returned

<i>l_MarchList</i>	Returns the list of signals to march during simulation.

Example

```
asiSetMarchList( session myMarchList )
```

Sets `myMarchList` as the list of signals to march for the given session.

asiSetPlotList

```
asiSetPlotList(  
    o_session  
    l_PlotList  
)  
=> l_PlotList
```

Description

Sets the plot list for the current simulation session.

Arguments

<i>o_session</i>	Simulation session object.
<i>l_PlotList</i>	List of items to plot in this simulation session. This is a list of selected signals (nets) or terminals comprising of the following information: window_id libName CellName ViewName

Value Returned

<i>l_PlotList</i>	Returns the list of items that can be plotted.

Example

```
asiSetPlotList( session myPlotList )
```

Sets the plot list for the given session.

asiSetSyncFlag

```
asiSetSyncFlag(  
    g_flag  
)  
=> g_flag
```

Description

This function makes the simulation run in a blocking mode. This implies that control does not come back till the simulation is over. This functionality is needed in the replay files in order to wait for a simulation to finish before proceeding. Call this function (`asiSetSyncFlag(t)`) just before launching a simulation. This function should only be used for testing capabilities. OCEAN should be used for scripting simulations.

Arguments

<code><i>g_flag</i>(t/nil)</code>	To synchronize the simulation.

Value Returned

<code>t</code>	When the function sets the value as <code>true</code> .
<code>nil</code>	Otherwise.

asiTransientStore

```
asiTransientStore(  
    o_session  
    t_fileName  
)  
=> t / nil
```

Description

Copies the final transient operating points in *processId.tr* to the name you pass in *fileName*. This function assumes that your simulator writes the final transient operating points to *netlistDir/raw/processId.tr*.

Note for Integrators: This function is defined as a method for the Analog Class. You can overload this method for your simulator class, as shown in the example for integrators.

Arguments

<i>o_session</i>	Specifies the session object.
<i>t_fileName</i>	Name of the file in which to store the transient node voltages.

Value Returned

<i>t</i>	The transient node voltages are stored in <i>fileName</i> .
<i>nil</i>	Indicates an error.

Examples

```
defmethod( asiTransientStore (( session <yourSimulator>_session ))  
    <insert any code you need>  
)
```

Note: See the *asiDcStore* function for similar example that is more complete.

asiMapNetName

```
asiMapNetName (
    t_dataDir
    l_specifier
    [ ?formatflag s_formatflag ]
)
=> l_specifier
```

Description

Maps the hierarchical schematic net name to the name in the netlist.

Arguments

<i>t_dataDir</i>	Specifies the data directory.
<i>l_specifier</i>	Lists the name of the net to map.
<i>?formatflag s_formatflag</i>	Performs name mapping of transient simulation data available in SST2 format. Set the value to "t" if this function is used in context of transient simulation data analysis and the transient simulation data is available in SST2 format.

Value Returned

<i>l_specifier</i>	Returns the list containing the mapped name. The list <i>l_specifier</i> passed as the argument is also destructively modified to contain the mapped name.

asiMapTerminalName

```
asiMapTerminalName(  
    t_dataDir  
    l_specifier  
    [ ?formatflag s_formatflag ]  
)  
=> l_specifier
```

Description

Maps the hierarchical schematic terminal name to the name in the netlist.

Arguments

<i>t_dataDir</i>	Specifies the data directory.
<i>l_specifier</i>	Lists the name of the terminal to map.
<i>?formatflag s_formatflag</i>	Performs name mapping of transient simulation data available in SST2 format. Set the value to "t" if this function is used in context of transient simulation data analysis and the transient simulation data is available in SST2 format.

Value Returned

<i>l_specifier</i>	Returns the list containing the mapped name. The list <i>l_specifier</i> passed as the argument is also destructively modified to contain the mapped name.

asiMapInstanceName

```
asiMapInstanceName(  
    t_dataDir  
    l_specifier  
    [ ?formatflag s_formatflag ]  
)  
=> l_specifier
```

Description

Maps the hierarchical schematic instance name to the name in the netlist.

Arguments

<i>t_dataDir</i>	Specifies the data directory.
<i>l_specifier</i>	Lists the name of the instance to map.
<i>?formatflag s_formatflag</i>	Performs name mapping of transient simulation data available in SST2 format. Set the value to "t" if this function is used in context of transient simulation data analysis and the transient simulation data is available in SST2 format.

Value Returned\

<i>l_specifier</i>	Returns the list containing the mapped name. The list <i>l_specifier</i> passed as the argument is also destructively modified to contain the mapped name.

asiRegCallBackOnSimComp

```
asiRegCallBackOnSimComp(  
    o_session  
    t_callback  
)  
=> t / nil
```

Description

Registers the specified user-defined callback function to run after completion of a simulation. The callback function should accept two arguments, a session object and the simulation status.

Note: In case of a simulation with multiple points, the callback is executed multiple times.



The registered callback function is executed only for local simulations.

Arguments

<i>o_session</i>	Simulation session object.
<i>t_callback</i>	Name of the callback function to be registered.

Value Returned

<i>t</i>	Returns <i>t</i> when the callback function is registered successfully.
<i>nil</i>	Otherwise, returns <i>nil</i> .

Example

```
asiRegCallBackOnSimComp(asiGetCurrentSession 'myfunc)
```

In this example, the `myfunc` function is registered to run after completion of a simulation. You can define `myfunc` as:

```
(defmethod myfunc (session status)  
    ....  
)
```

asiUnRegCallBackOnSimComp

```
asiUnRegCallBackOnSimComp(  
    t_callback  
)  
=> t / nil
```

Description

Deregisters the specified callback function registered to run on completion of the simulation.

Arguments

<i>t_callback</i>	Name of callback function to be deregistered.

Value Returned

<i>t</i>	The Callback is deregistered successfully.
<i>nil</i>	Otherwise, returns <i>nil</i> .

Example

```
asiUnRegCallBackOnSimComp(asiGetCurrentSession 'myfunc)
```

In this example, the `myfunc` function is deregistered.

asiRegCallBackOnSimCompForDist

```
asiRegCallBackOnSimCompForDist(  
    o_session  
    t_callback  
)  
=> t / nil
```

Description

Registers the specified user-defined callback function to run after completion of distributed jobs.

Arguments

<i>o_session</i>	The OASIS session object.
<i>t_callback</i>	Name of the callback function to be registered.

Value Returned

<i>t</i>	Returns <i>t</i> when the callback function is registered successfully.
<i>nil</i>	There is an error.

Examples

The following example registers the `myDistributedCallbackFunc` function to run after all the distributed jobs are completed:

```
session = asiGetCurrentSession()  
procedure(myDistributedCallbackFunc(session status)  
    printf("Session:%L \n Status:%L \n " session status)  
)  
asiRegCallBackOnSimCompForDist(session 'myDistributedCallbackFunc)  
  
=> t
```

asiUnRegCallBackOnSimCompForDist

```
asiUnRegCallBackOnSimCompForDist(  
    o_session  
    t_callback  
)  
=> t / nil
```

Description

Unregisters the specified user-defined callback function that is registered using the asiUnRegCallBackOnSimComp function.

Arguments

<i>o_session</i>	The OASIS session object.
<i>t_callback</i>	Name of callback function to be unregistered.

Value Returned

<i>t</i>	The function is unregistered successfully.
<i>nil</i>	There is an error.

Example

The following example unregisters the `myDistributedCallbackFunc` function:

```
asiUnRegCallBackOnSimCompForDist(asiGetCurrentSession() 'myDistributedCallbackFunc  
)  
=> t
```


almDefineParam_accuracyMode

```
almDefineParam_accuracyMode(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

This function is used to define the accuracy mode.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_accuracyMode(nport)
```

almDefineParam_additionalParam

```
almDefineParam_additionalParam(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

This function is used to enable additional parameters.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_additionalParam(nport)
```

almDefineParam_fq

```
almDefineParam_fq(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

This function is used to define a *f_q* parameter.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_fq(indq)
```

almDefineParam_noiseParaLabel

```
almDefineParam_noiseParaLabel(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

Noise parameter label.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_noiseParaLabel(nport)
```

almDefineParam_nportFileB

```
almDefineParam_nportFileB(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

nport file.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_nportFileB(nport)
```

almDefineParam_otherParaLabel

```
almDefineParam_otherParaLabel(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

This function is used to enable other paramters.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_otherParaLabel(nport)
```

almDefineParam_tranAdvanParaLabel

```
almDefineParam_tranAdvanParaLabel(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

Advanced `tran` parameter.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<code>nil</code>	Indicates an error.

Example

```
almDefineParam_tranAdvanParaLabel(nport)
```

almDefineParam_tranParaLabel

```
almDefineParam_tranParaLabel(  
    t_cellName  
)  
=> s_cellParameter / nil
```

Description

tran parameter label.

Arguments

<i>t_cellName</i>	Name of the cell.

Value Returned

<i>s_cellParameter</i>	The cell parameter.
<i>nil</i>	Indicates an error.

Example

```
almDefineParam_tranParaLabel(nport)
```


almGetModuleName

```
almGetModuleName (
    t_lib
    t_cell
    [ ?view g_view ]
    [ ?tool g_tool ]
)
=> s_moduleName / nil
```

Description

Returns the module name for the arguments specified. The netlist procedure is set with `almSetModuleName`.

Arguments

<code>t_lib</code>	Name of the library which the cell resides.
<code>t_cell</code>	Name of the cell.
<code>?view g_view</code>	Name of the view.
<code>?tool g_tool</code>	Name of the tool. If the <code>g_view</code> argument has been specified, and any view-specific information is available for this view, the <code>g_tool</code> argument is ignored.

Value Returned

<code>s_moduleName</code>	The module name, if one is defined.
<code>nil</code>	Indicates an error.

Example

```
almGetModuleName("analogLib" "vdc" ?tool "spectre")
```

almGetNamePrefix

```
almGetNamePrefix(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)  
=> t_namePrefix / nil
```

Description

Returns the name prefix for the arguments specified. The netlist procedure is set with `almSetNamePrefix`.

Arguments

<i>t_lib</i>	Name of the library which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. If the <i>g_view</i> argument has been specified, and any view-specific information is available for this view, the <i>g_tool</i> argument is ignored.

Value Returned

<i>t_namePrefix</i>	The name prefix, if one is defined.
nil	Indicates an error.

Example

```
almGetNamePrefix("analogLib" "nnpn" ?tool "Spectre")
```

almGetParameterList

```
almGetParameterList(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)  
=> l_parameterList / nil
```

Description

Returns the list of parameters for the arguments specified. The parameter list is set with `almSetParameterList`.

Arguments

<i>t_lib</i>	Name of the library which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. If the <i>g_view</i> argument has been specified, and any view-specific information is available for this view, the <i>g_tool</i> argument is ignored.
?entry <i>g_entry</i>	Parameter entry. The default value is <code>instParameters</code> .

Value Returned

<i>l_parameterList</i>	The list of parameter names, when defined.
<code>nil</code>	Indicates an error.

Examples

```
almGetParameterList("analogLib" "vdc" ?tool "spectre" ?entry "instParameters")
```

almGetTerminalList

```
almGetTerminalList(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)  
=> l_terminalList / nil
```

Description

Returns the list of terminal names for the arguments specified. The netlist procedure is set with `almSetTerminalList`.

Arguments

<i>t_lib</i>	Name of the library which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. If the <i>g_view</i> argument has been specified, and any view-specific information is available for this view, the <i>g_tool</i> argument is ignored.

Value Returned

<i>l_terminalList</i>	The list of terminal names, when defined.
nil	Indicates an error.

Example

```
almGetTerminalList("analogLib" "vdc" ?tool "spectre")
```

almGetTerminalMap

```
almGetTerminalMap(  
    t_lib  
    t_cell  
    S_terminal  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)  
=> s_map / nil
```

Description

Returns the simulator name of a terminal for the arguments specified. The netlist procedure is set with `almSetTerminalMap`.

Arguments

<i>t_lib</i>	Name of the library which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>S_terminal</i>	Name of the terminal for which the mapped name is returned.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. If the <i>g_view</i> argument has been specified, and any view-specific information is available for this view, the <i>g_tool</i> argument is ignored.

Value Returned

<i>s_map</i>	Mapped name of the terminal, when one is available.
nil	Indicates an error.

Example

```
almGetTerminalMap("analogLib" "vdc" "PLUS" ?tool "spectre")
```

almSetTerminalMap

```
almSetTerminalMap(  
    t_lib  
    t_cell  
    S_name  
    g_map  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)
```

Description

Sets the mapped name (simulator name) for a terminal name (schematic name) for the arguments specified. The mapped name is used for results display and it is used by the simulator interface in the simulator input file.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>S_name</i>	Schematic name of the terminal.
<i>g_map</i>	Mapped simulator name of the terminal.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.

Value Returned

None

Example

```
almSetTerminalMap("analogLib" "res" "PLUS" ":1" ?tool 'spectre)
```

almGetOpPointParamMap

```
almGetOpPointParamMap(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)  
=> l_opMap / nil
```

Description

Returns the operating-point parameter map for the arguments specified. The netlist procedure is set with `almSetOpPointParamMap`.

Arguments

<i>t_lib</i>	Name of the library which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. If the <i>g_view</i> argument has been specified, and any view-specific information is available for this view, the <i>g_tool</i> argument is ignored.

Value Returned

<i>l_opMap</i>	Operating-point parameter map, if one is defined.
<i>nil</i>	Indicates an error.

Example

```
almGetOpPointParamMap( "analogLib" "npn" ?tool "spectre" )
```

almSetOpPointParamMap

```
almSetOpPointParamMap(  
    t_lib  
    t_cell  
    l_map  
    [ ?view t_view ]  
    [ ?tool t_tool ]  
)
```

Description

Sets the operating-point parameter map for the arguments specified. This map is used in results display.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>l_map</i>	Operating-point map.
<i>t_view</i>	Name of the view.
<i>t_tool</i>	Name of the tool. This argument is ignored when the <i>t_view</i> argument is specified.

Example

```
almSetOpPointParamMap( "analogLib" "npn" nil ?tool 'spectre )
```


almGetNetlistProcedure

```
almGetNetlistProcedure(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)  
=> s_procedure / nil
```

Description

Returns the netlist procedure for the arguments specified. The netlist procedure is used to netlist an instance. The netlist procedure is declared with `almSetNetlistProcedure`.

Arguments

<i>t_lib</i>	Name of the library which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. If the <i>g_view</i> argument has been specified, and any view-specific information is available for this view, the <i>g_tool</i> argument is ignored.

Value Returned

<i>s_procedure</i>	Returns the netlist procedure, if one is defined.
<i>nil</i>	There is an error.

Example

```
almGetNetlistProcedure("analogLib" "vdc" ?tool "spectre")
```

almGetViewInfoNameList

```
almGetViewInfoNameList(  
    t_lib  
    t_cell  
)  
=> l_view_list / nil
```

Description

Returns the list of view-name strings for which view-specific information is available, and for which the view exists.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.

Value Returned

<i>l_view_list</i>	A list of strings which are the names of the views that have view-specific information.
<i>nil</i>	If no views have view-specific information.

Example

```
almGetViewInfoNameList( "rfLib" "balun" )
```

almGetNetlistType

```
almGetNetlistType(  
    t_lib  
    t_cell  
    [ ?view t_view ]  
)  
=> t_netlistType / nil
```

Description

Returns the netlist type for the library, cell and view specified, if view-specific information is available. This procedure issued during netlisting.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>t_view</i>	Name of the view.

Value Returned

<i>t_netlistType</i>	Netlist type for the view or the tool specified, if the view has view-specific information.
<i>nil</i>	Returns <i>nil</i> if no view-specific information is available.

Example

```
almGetNetlistType( "rfLib" "balun" ?view "veriloga")
```

almHasViewInformation

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

Description

Determines if the view-specific information is available for the library, cell, and view specified.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>t_view</i>	Name of the view.

Value Returned

<i>t</i>	Returns <i>t</i> when the view-specific information is available for the lib, cell, and view specified.
nil	Indicates an error.

Example

```
almHasViewInformation("rfLib" "balun" "veriloga")
```

almSetNamePrefix

```
almSetNamePrefix(  
    t_lib  
    t_cell  
    t_name  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)
```

Description

Sets the name prefix for the view or the tool specified.

The name prefix is used for the instance name in the netlist. The use of the name prefix in the netlist depends on the simulator interface. If the simulator requires that the instance name of a type of components starts with a certain prefix, then the interface must use this name prefix. If this is not a requirement for a simulator, then the interface should not use the name prefix. For the spectre interface, the name prefix is ignored.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>t_name</i>	Name of the prefix.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.

Example

```
almSetNamePrefix ("analogLib" "nnp" "analog" "Q" ?tool "Spectre")
```

almSetModuleName

```
almSetModuleName (
    t_lib
    t_cell
    t_name
    [ ?view g_view ]
    [ ?tool g_tool ]
)
```

Description

Sets the module name for the view or tool.

The module name, also referred to as component name or model name, is used in the netlist to identify the type of component.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>t_name</i>	Module name.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.

Example

```
almSetModuleName("analogLib" "res" "resistor" ?tool "spectre")
```

almSetNetlistProcedure

```
almSetNetlistProcedure(  
    t_lib  
    t_cell  
    s_procedure  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
)
```

Description

Sets the netlist procedure for the arguments specified.

The netlist is used to netlist an instance.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>s_procedure</i>	Name of the netlist procedure.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.

Example

The following code sets the netlist procedure Spectrercprim for the vdc cell in library analogLib:

```
almSetNetlistProcedure( "analogLib" "vdc" 'SpectrercPrim ?tool "spectre")
```

almSetParameterList

```
almSetParameterList(  
    t_lib  
    t_cell  
    l_parameter  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)
```

Description

Sets list of parameter names for the arguments specified.

These parameters are printed to the netlist if the user has given these values.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>l_parameter</i>	List of parameter names. Each element must be a symbol.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.
?entry <i>s_entry</i>	Parameter entry. The default value is <code>instParameters</code> .

Example

```
almSetParameterList( "analogLib" "npn" '(area m trise region) ?tool "spectre"  
?entry "instParameters")
```


almSetTerminalList

```
almSetTerminalList(  
    t_lib  
    t_cell  
    l_termList  
    [ ?view t_view ]  
    [ ?tool t_tool ]  
)
```

Description

Sets the list of terminal names for the lib arguments specified.

The simulator names of the signals connected to these terminals are printed to the netlist, according to the order of this terminal list.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>l_termList</i>	Terminal list.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.

Example

```
almSetTerminalList( "analogLib" "npn" ' (C B E) "vdc" ?tool "spectre")
```

almSetPropMappingList

```
almSetPropMappingList(  
    t_lib  
    t_cell  
    l_parameter  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)
```

Description

Sets the list of propMapping for the arguments specified. The parameters specified in the *l_parameter* list are mapped and printed to the netlist if the user has specified these values.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>l_parameter</i>	List of parameter names. Each element must be a symbol.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.
?entry <i>s_entry</i>	Parameter entry. The default value is "propMapping".

Example

```
almSetPropMappingList("analogLib" "mind" ' (nil coupling K1) ?tool "spectre" ?entry  
"propMapping")
```

almGetPropMappingList

```
almGetPropMappingList(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)  
=> l_propMappingList / nil
```

Description

Returns the `propMapping` parameter list for the arguments specified. The `propMapping` list is set with `almSetPropMappingList`.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.
?entry <i>g_entry</i>	Parameter entry. The default value is "propMapping".

Value Returned

<i>l_propMappingList</i>	The list containing <code>propMapping</code> names, when defined.
<i>nil</i>	Indicates an error.

Example

```
almGetPropMappingList("analogLib" "vdc" ?tool "spectre" ?entry "propMapping")
```

almSetOtherParameterList

```
almSetOtherParameterList(  
    t_lib  
    t_cell  
    l_parameter  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)
```

Description

Sets list of other parameter names for the arguments specified.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>l_parameter</i>	List of parameter names. Each element must be a symbol
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.
?entry <i>g_entry</i>	Parameter entry. The default value is "otherParameters".

Example

```
almSetOtherParameterList("analogLib" "mind" '(ind1 ind2) ?tool "spectre" ?entry  
"otherParameters")
```

almGetOtherParameterList

```
almGetOtherParameterList(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool t_tool ]  
    [ ?entry s_entry ]  
)  
=> l_parameterList / nil
```

Description

Returns the `otherParameter` list for the arguments specified. The `otherParameter` list is set with `almSetOtherParameterList`.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>?view g_view</i>	Name of the view.
<i>?tool g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.
<i>?entry g_entry</i>	Parameter entry. The default value is "otherParameters".

Value Returned

<i>l_parameterList</i>	The list containing <code>otherParameter</code> names, when defined.
<i>nil</i>	Indicates an error.

Example

```
almGetOtherParameterList("analogLib" "vdc" ?tool "spectre" ?entry  
"otherParameters")
```

almGetStringParameterList

```
almGetStringParameterList(  
    t_lib  
    t_cell  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)=> l_parameterList / nil
```

Description

Returns the list of string type parameters for the arguments specified. The parameter list is set with `almSetStringParameterList`.

Arguments

<code>t_lib</code>	Name of the library which the cell resides.
<code>t_cell</code>	Name of the cell.
<code>?view g_view</code>	Name of the view.
<code>?tool g_tool</code>	Name of the tool. If the <code>g_view</code> argument has been specified, and any view-specific information is available for this view, the <code>g_tool</code> argument is ignored.
<code>?entry g_entry</code>	Parameter entry. The default value is <code>stringParameters</code> .

Value Returned

<code>l_parameterList</code>	The list of parameter names, when defined.
<code>nil</code>	Indicates an error.

Example

```
almGetStringParameterList("analogLib" "vdc" ?tool "spectre" ?entry  
'stringParameters)
```

almSetStringParameterList

```
almSetStringParameterList(  
    t_lib  
    t_cell  
    l_parameter  
    [ ?view g_view ]  
    [ ?tool g_tool ]  
    [ ?entry g_entry ]  
)
```

Description

Sets list of string parameter names for the arguments specified. These parameters are printed to the netlist if the user has given these values.

Arguments

<i>t_lib</i>	Name of the library in which the cell resides.
<i>t_cell</i>	Name of the cell.
<i>l_parameter</i>	List of parameter names. Each element must be a symbol.
?view <i>g_view</i>	Name of the view.
?tool <i>g_tool</i>	Name of the tool. This argument is ignored when the <i>g_view</i> argument is specified.
?entry <i>g_entry</i>	Parameter entry. The default value is stringParameters.

Example

```
almSetStringParameterList( "analogLib" "vpwlf" ' (fileName) ?tool "spectre" ?entry  
'stringParameters)
```

ancGetSimInstName

```
ancGetSimInstName(  
    l_netlistDpl  
)  
=> t_extName
```

Description

This function provides the external instance name for the current instance being formatted in the netlister. The function must only be used by customized socket netlisters or custom socket netlist procedures. The routine takes into consideration the netlist mode (FNL, HNL or IHNL), case mapping (lower, upper or mixed) and the prefix for the instance.

Argument

<i>l_netlistDpl</i>	Netlist DPL passed to the netlist procedure.

Value Returned

<i>t_extName</i>	Returns the external name for current instance.

Example

```
ancGetSimInstName( netdpl )
```


ancAdjustNameCase

```
ancAdjustNameCase (
    S_name
    s_type
)
=> S_name
```

Description

The function adjusts case for the name passed, based on the type specified.

Arguments

<i>S_name</i>	The name on which to operate.
<i>s_type</i>	The case adjustment to be made. Valid values are 'lower, 'upper or 'mixed.

Value Returned

<i>S_name</i>	Returns the case adjusted name.

Example

```
ancAdjustNameCase ( "xI0" 'upper )
```

Returns "XI0".

drbBrowseFormCB

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

Description

Opens the Browse Project Hierarchy window.

Argument

None

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

msgHelp

```
msgHelp(  
    S_prodID  
    S_msgID  
)  
=> t / nil
```

Description

This function is used to access extended help for an error or a warning message. Currently, Distributed Processing (DP) is the only product that supports this feature.

Arguments

<i>S_prodID</i>	The product ID. In case of Distributed Processing, it is 'DP'.
<i>S_msgID</i>	The error message ID. This is the numeric identifier that is available in the error message box.

Value Returned

t	If a valid product ID and error message ID is passed.
nil	If prodID or errorID is not valid.

Example

```
(msgHelp 'DP 8)
```

Displays extended help for the DP error message, whose message ID is 28.

addCheck

```
addCheck(  
    t_name  
    [ ?sub t_sub ]  
    [ ?dev t_dev ]  
    [ ?devlist l_devlist ]  
    [ ?prim t_prim ]  
    [ ?mod t_mod ]  
    [ ?instparam t_instparam ]  
    [ ?modelparam t_modelparam ]  
    [ ?opparam t_opparam ]  
    [ ?parameter t_parameter ]  
    [ ?expression t_expression ]  
    [ ?min t_min ]  
    [ ?max t_max ]  
    [ ?regions t_regions ]  
    [ ?duration t_duration ]  
    [ ?message t_message ]  
    [ ?severity t_severity ]  
    [ ?analyses l_analyses ]  
)  
=> t_name / nil
```

Description

Adds a new device check. Checks can be added to check a device parameter, a subcircuit parameter, a design variable or an expression. The scope of check is decided by the arguments ?sub, ?dev, ?mod, and ?prim. The ?min and ?max parameters can be used to set up the safe region if the parameter/expression being checked goes out of this region and a violation is reported.

Note: This function can be used only in an Analog Design Environment L session. When the [enableDeviceChecking](#) environment variable is set to t, the function can be used for an Analog Design Environment XL session too.

Arguments

<i>t_name</i>	Name for the device check.
?sub <i>t_sub</i>	Subcircuit name. When specified, the check will apply to all instances of this subcircuit.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?dev t_dev</code>	Device name. When specified, the check will apply only to that device. When specified with the subcircuit name, the check will apply to the specified device in all instances of the given subcircuit.
<code>?devlist l_devlist</code>	List of device names. When specified, the check will apply only to the those devices. When specified with the subcircuit name, the check will apply to the specified devices in all instances of the given subcircuit.
<code>?prim t_prim</code>	Primitive name. When specified, the check will apply to all instances of this primitive. When specified with the subcircuit name, the check will apply to all instances of this primitive in all instances of the given subcircuit.
<code>?mod t_mod</code>	Model name. When specified, the check will apply to all instances of this model. When specified with the subcircuit name, the check will apply to all instances of this model in all instances of the given subcircuit.
<code>?instparam t_instparam</code>	Instance parameter to be checked.
<code>?modelparam t_modelparam</code>	Model parameter to be checked.
<code>?opparam t_opparam</code>	Operating point parameter to be checked.
<code>?parameter t_parameter</code>	Circuit parameter (design variable) to be checked.
<code>?expression t_expression</code>	Expression to be checked.
<code>?min t_min</code>	Lower bound of the safe operating area. When the value of the specified parameter/expression goes below this limit, a violation is reported.
<code>?max t_max</code>	Upper bound of the safe operating area. When the value of the specified parameter/expression exceeds this limit, a violation is reported.
<code>?regions t_regions</code>	List of safe regions, specified as a space separated string or list of strings. Only valid when <code>opparam = region</code> . If a device operates outside these regions, a violation is reported.

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>?duration t_duration</code>	Duration for which a parameter must exceed the bound to cause a violation. Only applies to transient analysis.
<code>?message t_message</code>	Custom message that is printed when this check is violated.
<code>?severity t_severity</code>	Severity of the violation. Valid Values: Notice, Warning, Error.
<code>?analyses l_analyses</code>	List of analyses for which this assert should be turned on. Valid analyses are tran, dc and dcOp.

Value Returned

<code>t_name</code>	Returns the name of device check that was added.
<code>nil</code>	The command is not successful and prints an error.

Examples

```
addCheck("check1" ?primitive "bjt" ?opparam "vbe" ?min "0.7" ?severity "Warning")
```

Checks if vbe for any BJT device falls below 0.7.

```
addCheck("check2" ?primitive "mos3" ?opparam "region" ?regions "triode sat"  
?severity "Notice")
```

Checks if any mos3 device goes out of triode or saturation regions.

```
addCheck("check3" ?subckt "amplifier" ?model "mynpn" ?opparam "ibc" ?max 1u  
?severity "Warning")
```

Checks if any instance of the model mynpn has base to collector current greater than 1u and limits the check to all instances of the amplifier subcircuit.

deleteChecks

```
deleteChecks (
    t_check1
    [ t_check2 t_check3... ]
)
=> l_checks / nil
```

Description

Deletes device checks.

Note: This function can be used only in an Analog Design Environment L session. When the enableDeviceChecking environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Argument

t_check1 - t_checkn Names of checks to be deleted.

Value Returned

<i>l_checks</i>	Returns list of checks passed.
<i>nil</i>	Returns <code>nil</code> and prints an error if the simulator is not Spectre.

Example

```
deleteChecks ("check1" "check2" "check3")
```

densityEstimateWaveform

```
densityEstimateWaveform(  
    histWf  
)  
=> waveform
```

Description

Returns the density estimator waveform of a histogram.

Argument

<i>histWf</i>	Histogram waveform with enabled density estimator.

Value Returned

<i>waveform</i>	Returns the density estimator waveform.

Example

```
W00 = drCreateWaveform( drCreateVec( 'double list( 1 2 3 4 5 ) ) drCreateVec( 'double  
list( 10 20 30 40 50 ) ) )  
  
histogramWf=histogram2D(W00 10 "standard" nil t) :(histogram wave)  
densityWf =densityEstimateWaveform(histogramWf) :(density estimator waveform  
wave)  
plot densityWf (you can plot this wave)
```


disableAllChecks

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

Description

Disables all device checks.

Note: This function can be used only in an Analog Design Environment L session. When the [enableDeviceChecking](#) environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Value Returned

<code>t</code>	Successful operation.
<code>nil</code>	Returns <code>nil</code> and prints an error if the simulator is not Spectre.

Example

```
disableAllChecks()
```

disableChecks

```
disableChecks(  
    t_check1  
    [ t_check2 t_check3... ]  
)  
=> l_checks / nil
```

Description

Disables device checks. Only enabled device checks appear in Spectre input deck.

Note: This function can be used only in an Analog Design Environment L session. When the [enableDeviceChecking](#) environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Argument

t_check1 - t_checkn Names of checks to be disabled.

Value Returned

<i>l_checks</i>	Returns list of checks passed.
<i>nil</i>	Returns <code>nil</code> and prints an error if the simulator is not Spectre.

Example

```
disableChecks("check1" "check2" "check3")
```

disableDeviceChecking

```
disableDeviceChecking(  
    )  
=> no / nil
```

Description

Disables device checking. It is the same as turning off the *Enable Device Checking* checkbox in the *Device Checking Setup* form. No checks are written to Spectre input file. No device checking happens.

Note: This function can be used only in an Analog Design Environment L session. When the enableDeviceChecking environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Value Returned

<code>no</code>	Returns <code>no</code> if device checking was already disabled.
<code>nil</code>	Returns <code>nil</code> if device checking was already disabled.
	The command is not successful and prints an error.

Example

```
disableDeviceChecking()
```

displayChecks

```
displayChecks(  
    ?resultsDir t_resultsDir  
    [ t_check1 t_check2 t_check3... ]  
)  
=> nil
```

Description

Displays device checks. If no name is provided, this function displays all checks that have been added so far. If no device checks have been added and a results directory is set by using the *openResults()* command, it prints device checks from the `asserts.info.asserts` file in that results directory. If the results directory is provided, it displays device checks found in the `asserts.info.asserts` file in that results directory.

Note: This function can be used only in an Analog Design Environment L session. When the *enableDeviceChecking* environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Arguments

`?resultsDir t_resultsDir`

Directory containing the PSF files (results).
When specified, this argument will only be used internally and will not alter the current results directory which was set by the *openResults()* command.

`t_check1 - t_checkn`

Names of the checks to be displayed.

Value Returned

`nil`

Returns `nil`.

enableAllChecks

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

Description

Enables all device checks.

Note: This function can be used only in an Analog Design Environment L session. When the [enableDeviceChecking](#) environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Value Returned

<code>t</code>	Successful operation.
<code>nil</code>	Prints an error if the simulator is not Spectre.

Example

```
enableAllChecks ()
```

enableChecks

```
enableChecks (
    t_check1 - l_checkn
)
=> l_checks / nil
```

Description

Enables device checks. Only enabled device checks appear in Spectre input deck.

Note: This function can be used only in an Analog Design Environment L session. When the [enableDeviceChecking](#) environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Argument

t_check1 - t_checkn Names of checks to be enabled.

Value Returned

<i>l_checks</i>	Returns list of checks passed.
<i>nil</i>	Returns <code>nil</code> and prints an error if the simulator is not Spectre.

Example

```
enableChecks ("check1" "check2" "check3")
```

enableDeviceChecking

```
enableDeviceChecking(  
    )  
=> yes / nil
```

Description

Enables device checking. It is the same as turning on the *Enable Device Checking* check box in the *Device Checking Setup* form.

Note: This function can be used only in an Analog Design Environment L session. When the enableDeviceChecking environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Value Returned

<code>yes</code>	Returns <code>yes</code> if the command is successful.
<code>nil</code>	The command is not successful and prints an error.

Examples

```
enableDeviceChecking()
```

setDevCheckOptions

```
setDevCheckOptions(  
    [ ?analysis s_analysis ]  
    [ ?start t_start ]  
    [ ?stop t_stop ]  
    [ ?severity t_severity ]  
    [ ?enableAll t_enableAll ]  
    [ ?disableAll t_disableAll ]  
)
```

Description

This function can be used to set various parameters of checklimit statements. User can override the severity specified on individual asserts, provide the time interval during which the checking should be done (applies only to transient analysis), or enable/disable all device checks for the given analysis.

Note: This function can be used only in an Analog Design Environment L session. When the enableDeviceChecking environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Arguments

<code>?analysis <i>s_analysis</i></code>	Name of analysis.
<code>?start <i>t_start</i></code>	Time at which device checking will start in a transient analysis. Not applicable to other analysis.
<code>?stop <i>t_stop</i></code>	Time at which device checking will stop in a transient analysis. Not applicable to other analysis.
<code>?severity <i>t_severity</i></code>	If specified, this severity level will override the severity specified for individual asserts. Valid values: Warning, Notice, Error, None
<code>?enableAll <i>g_enableAll</i></code>	If true, all checks are enabled during this analysis.
<code>?disableAll <i>g_disableAll</i></code>	If true, all checks are disabled during this analysis.

Value Returned

`nil` Returns `nil`.

printViolations

```
printViolations(  
    [ ?output t_output ]  
    [ ?checks l_checks ]  
    [ ?devices l_devices ]  
    [ ?models l_models ]  
    [ ?primitives l_primitives ]  
    [ ?resultsDir t_resultsDir ]  
)  
=> t / nil
```

Description

Prints a summary of all violations. Format of the output is similar to that of the output from the *Print* button on the *Violations Display* form. The `resultsDir` argument can be used to print violations from the different results directory.

Note: This function can be used only in an Analog Design Environment L session. When the [enableDeviceChecking](#) environment variable is set to `t`, the function can be used for an Analog Design Environment XL session too.

Arguments

<code>?output t_output</code>	Prints the violations to the specified file. If not specified, the violations are printed in the CIW.
<code>?checks l_checks</code>	Prints the violations for the specified checks. If not specified, violations for all the checks are printed.
<code>?devices l_devices</code>	Prints the violations for the specified devices. If not specified, the violations for all the devices are printed.
<code>?models l_models</code>	Prints the violations for the specified device models. If not specified, the violations for all the device models are printed.
<code>?primitives l_primitives</code>	Prints the violations for the specified primitive types or models of the specified primitive type. If not specified, the violations for all the primitives are printed.
<code>?resultsDir t_resultsDir</code>	

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

r Specifies the directory containing the PSF files (results). When specified, this argument is used internally and does not alter the current results directory which was set by the *openResults()* command.

Value Returned

t The violations are printed.
nil The violations are not present for the design.

Examples

The following example prints all the violations in the CIW.

```
printViolations()  
=> Violations for Check1 :  
During circuit read in  
Instance    Param  Value  Remark  
/R1         r      20K    Warning:Exceeded upper limit 10K.  
  
Violations for Check2 :  
During circuit read in  
Parameter   Value   Remark  
tem         27     Warning:Exceeded upper limit 20.
```

The following example prints the violations for Check2 in the CIW.

```
printViolations(?checks "Check2")  
=>Violations for Check2 :  
During circuit read in  
Parameter   Value   Remark  
tem         27     Warning:Exceeded upper limit 20.
```

The following example prints all the violations to the `printviolation` file.

```
printViolations(?output "./simulation/ampTest/printviolation")  
=> t
```

The following example prints all the violations to the `printviolation` file with results loaded from the `./simulation2/ampTest/spectre/schematic` directory.

```
printViolations(?output "./simulation/ampTest/printviolation" ?resultsDir "./simulation2/ampTest/spectre/schematic")  
=> t
```

Note: You can change the default simulation directory using the `projectDir` environment variable.

captabSummary

```
captabSummary(  
    [ ?resultsDir t_resultsDir ]  
    [ ?analysis o_analysis ]  
)
```

Description

Prints a summary of the capacitance table from the specified results directory.

Arguments

<i>t_resultsDir</i>	Directory containing the PSF files (results).
<i>o_analysis</i>	Name of the analysis for the capacitance table.

Examples

```
captabSummary(?resultsDir "./simulation/psf" ?analysis "finalTimeCapinfo.captab")
```

Prints a summary of `finalTimeCapinfo.captab` from `./simulation/psf` directory.

evmOFDM

```
evmOFDM(  
    o_waveform1  
    o_waveform2  
    e_sigStandard  
    n_tstart  
    e_modulationType  
    n_fftSize  
    n_PrefixLength  
    n_SymbolPeriod  
    [ ?packet n_packetLength ]  
    [ ?modifier e_modifier ]  
    [ ?skiplength n_skiplength ]  
    [ ?idx n_idx ]  
    [ ?getval g_getval ]  
)  
=> constellation o_waveform evm_value / nil
```

Description

Processes the I and Q waveform outputs from the hb-envlp simulation run to calculate the Error Vector Magnitude (EVM) and plot the I versus Q scatterplot (constellation). EVM is a useful measurement to describe the overall signal amplitude and phase modulated signal quality. It is based on a statistical error distribution normalized from an ideal digital modulation. Orthogonal Frequency Division Multiplexing (OFDM) is a modern high throughput modulation scheme widely used in wireless signals, such as 802.11a,g,n, where EVM measurement is useful. The EVM is calculated by detecting the I and Q signal levels corresponding to constellation points of each modulation type (can be BPSK, QPSK, 16QAM, and 64QAM) and calculating the difference between the signal level and the ideal signal level.

Note: This method is supported for families of waveforms.

Arguments

<i>o_waveform1</i>	The waveform for the I signal.
<i>o_waveform2</i>	The waveform for the Q signal.
<i>e_sigStandard</i>	The standard of the signal.

Valid values: 802_11a, 802_11g_ERP,
802_11n_20M_Mix, 802_11n_20M_Legacy,
802_11n_20M_Green, 802_11n_40M_Mix,
802_11n_40M_Legacy, 802_11n_40M_Green,
802_11ac

Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

<code>n_tstart</code>	The start time for the first non-zero point of the signal.
<code>e_modulationType</code>	The modulation type of data carriers.
<code>n_fftSize</code>	The points of FFT, such as 64 for 802.11a.
<code>n_PrefixLength</code>	The length of cyclic prefix, number of samples.
<code>n_SymbolPeriod</code>	The symbol period, such as 4us for long GI 802.11a.
<code>?packet n_packetLength</code>	The length of one packet, can be null if there is only one packet in wave.
<code>?modifier e_modifier</code>	The type of showing EVM value, "Percent" or "dB20".
<code>?skiplength n_skiplength</code>	Used under customer mode, time length between start and data symbol.
<code>?idx n_idx</code>	The index of symbol used for channel estimation.
<code>?getval g_getval</code>	Based on the passed Boolean value it returns the following: ■ if <code>nil</code>, it returns EVM value ■ if <code>t</code>, it returns constellation waveform Default is <code>nil</code> .

Value Returned

<code>o_waveform</code>	The waveform object representing the EVM value computed from input waveforms.
<code>evm_value</code>	The EVM value if <code>getVal()</code> is true.
<code>nil</code>	Function call unsuccessful.

Examples

```
evmOFDM(real(harmonic(v("/net9" ?result "envlp_fd") '1)) imag(harmonic(v("/net9"
?result "envlp_fd") '1)) "802_11a" 4e-06 'QAM64 64 16 4e-06 ?modifier "Percent"
?packet 0.000173)
```

Calculates the EVM value in term of percent with packet length as 173u and FFT as 64-points.

relxOption

```
relxOption(  
    [ s_option1 g_optionValue1 ]....[ s_optionN g_optionValueN ]  
)  
=> undefined / nil
```

Description

Enables you to specify the RelXpert options, along with values, to be used by the simulator.

Arguments

<i>s_optionN</i>	Name of the RelXpert option.
<i>g_optionValueN</i>	Value of the RelXpert option.

Value Returned

undefined	Undefined.
nil	The function is not successful.

Examples

```
relxOption( 'enableRelxpert' t )
```

Enables Relxpert.

asiAddModelLibSelection

```
asiAddModelLibSelection(  
    { o_session | o_tool }  
    t_modelLibFile  
    t_section  
)  
=> t / nil
```

Description

Adds a model file by adding an entry to the model library file section (corresponding environment variable modelFiles) and calls `asiInvalidateControlStmts`.

Arguments

<i>o_session</i>	Name of the session.
<i>o_tool</i>	Name of the tool.
<i>t_modelLibFile</i>	Name of the model library file. If this field is blank, no model files are added.
<i>t_section</i>	Name of the model library file section.

Value Returned

<i>t</i>	The function is successful.
<i>nil</i>	The function is not successful.

Examples

```
asiAddModelLibSelection (session "/servers/cic_ade/gashish/LAB/lab/Models/  
allModels.scs" "")
```

Adds `allModels.scs` model file and adds an entry to the model library file section.

asiRemoveAllModelLibSelection

```
asiRemoveAllModelLibSelection(  
    { o_session | o_tool }  
)  
=> t
```

Description

Removes all the model files by setting the modelFiles variable to nil. In addition, it also invalidates the asiSendControlStmts flowchart step by calling asiInvalidateControlStmts.

Arguments

<i>o_session</i>	Name of the session.
<i>o_tool</i>	Name of the tool.

Value Returned

t	All model files deleted.

Examples

Removes all the model files for *session*.

```
asiRemoveAllModelLibSelection (session)
```


Virtuoso ADE SKILL Reference

Functions to Customize Simulator Integration

Functions to Access Simulation Results

This topic lists the SKILL functions that let you access simulation results.

These functions can be classified in the following categories:

- [Data Access Functions](#)
- [Selection Functions](#)
- [Waveform Data Object Functions](#)
- [Direct Plot Functions for RF Analyses](#)
- [OCEAN Script Functions](#)

Data Access Functions

This topic lists the SKILL functions that let you define data access functions and data mapping functions.

[asiDefineDataAccessFunction](#)

[asiDefineDataMappingFunction](#)

[asiGetCalcResultsDir](#)

[asilnit<yourSimulator>DataAccessFunction](#)

[VAR](#)

[DATA](#)

[VS](#)

[QP](#)

[OPT](#)

[MP](#)

[NG](#)

[VN](#)

[VN2](#)

[VNP](#)

[VNPP](#)

[VPD](#)

[VF](#)

[VT](#)

[IS](#)

[IT](#)

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

IF

IDC

VDC

SIMULATOR

asiDefineDataAccessFunction

```
asiDefineDataAccessFunction(  
    o_tool  
    s_dataType  
    s_dataFunction  
)  
=> s_dataFunction
```

Description

Redefines a data access function.

Typically you do not need to redefine a data access function.

Arguments

<i>o_tool</i>	Simulation tool object
<i>s_dataType</i>	Data type for the data access function. Valid Values: VT, VF, VS, IT, IF, IS, VDC, IDC, VTRE, OP, OPT, MP, VN2, VNP, or VNPP
<i>s_dataFunction</i>	Name of the data access function that you write to access the data from the Parameter Storage Format (PSF) file.

Value Returned

<i>s_dataFunction</i>	Returns the name of the data access function.

Examples

Calls the XYZVT function, which gets the VT information from the PSF file.

```
asiDefineDataAccessFunction( tool 'VT' 'XYZVT' )  
    procedure( XYZVT(specifier dataDir simData)  
        let((wave daf)  
            daf = asiGetDataAccessFunction(asiGetTool('analog') 'VT')  
            wave = apply( daf list(specifier dataDir simData))  
            wave  
        )  
    )
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Alternatively, use `asiDefineDataAccessFunction` to achieve the same functionality. This would be helpful for user defined data access types. For example, the existing code would work with VT, IT etc., but the code below would work with a new analysis called `sp`.

```
procedure( XYZDataAccessSp(specifier dataDir simData)
  let((wave)
    wave = asiGetDrlData("sp" specifier dataDir)
    wave
  )
)
procedure( XYZMapNetNameSp( dataDir specifier )
  let((
    specifier = parseString((concat "s" car(specifier) "_" cadr(specifier)))
    specifier
  )
)
```

asiDefineDataMappingFunction

```
asiDefineDataMappingFunction(  
    o_tool  
    s_dataType  
    s_function  
)  
=> s_function
```

Description

Defines the data mapping functions.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Arguments

<i>o_tool</i>	Simulation tool object
<i>s_dataType</i>	Datatype for the data access function. Valid Values: VT, VF, VS, IT, IF, IS, VDC, IDC, VTRE, OP, OPT, MP, VN2, VNP, and VNPP
<i>s_function</i>	Name of the data mapping function that you write to convert the Cadence SPICE PSF name to your name. Typically, you use one of the following: ■ <code>asi<yourSimulator>MapNetName</code> ■ <code>asi<yourSimulator>MapTerminalName</code> ■ <code>asi<yourSimulator>MapInstanceName</code> Callback parameter list: (<i>s_dataDir</i> <i>l_specifier</i>) <i>s_dataDir</i> specifies the data directory, and <i>l_specifier</i> is a list containing the name of the item to look for in the PSF file.

Value Returned

<i>s_function</i>	Returns the name of the data mapping function.

Examples

Calls the `XYZMapNetName` function to convert the *Spectre*-like net name to the net name in the PSF file of the XYZ simulator. For example, you might use this function to convert lower case characters to uppercase, or you might convert the hierarchical delimiter (^) to the hierarchical delimiter for your simulator.

```
asiDefineDataMappingFunction( tool 'VT' asiXYZMapNetName )
procedure( asiXYZMapNetName( dataDir specifier )
    ; replace ^ by . in net names...
    specifier = asiMapNetName( dataDir specifier )
    rplaca(
        specifier
        artStrSubstitute( car( specifier ) "^" "." )
    )
)
```


asiGetCalcResultsDir

```
asiGetCalcResultsDir(  
    )  
=> t_DataDir / nil
```

Description

Returns the results directory currently used by calculator functions.

Arguments

None

Value Returned

<i>t_DataDir</i>	Returns the results directory currently used by calculator functions.

<i>nil</i>	Returns <i>nil</i> if no results are selected.

Examples

Returns the current results directory that is being used by calculator functions.

```
asiGetCalcResultsDir()
```

asiInit<yourSimulator>DataAccessFunction

```
asiInit<yourSimulator>DataAccessFunction(  
    o_tool  
)  
=> t
```

Description

Calls the procedures that modify your data access routines.

Note: You must write `asiInit<yourSimulator>DataAccessFunction`, where `<yourSimulator>` is the name of your simulator. Do not include the angle brackets (<>).

Arguments

<code>o_tool</code>	Simulation tool object

Value Returned

<code>t</code>	Returns <code>t</code> when your procedures are called.

Note: You must write this procedure to return `t`.

Examples

Creates the procedure that calls the functions to modify the data access routines for the XYZ simulator.

```
procedure( asiInitXYZDataAccessFunction(tool)  
    <insert your code>  
    t  
)
```

All functions described in the following section accept an optional argument called *data-directory*. If this argument is not provided, open results are used by the function. For example, `VT("out")` will return the transient voltage at the "out" from the currently open results. If the data-directory is provided, this argument will only be used internally and will not alter the currently selected results.

The Ocean commands `openResults` and `selectResult` do not have any effect on these functions. In case multiple analyses of the same type are present, the first found occurrence of that analysis will be used.

VAR

```
VAR (
    t_variableNname
    [ t_dataDir ]
)
=> n_number / nil
```

Description

Returns the value of the specified design variable.

Arguments

<i>t_variableName</i>	Name of the variable.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>n_number</i>	Returns the value of the specified design variable.
<i>nil</i>	Returns <i>nil</i> if the design variable was not found. Prints an error message if the data-directory does not contain valid PSF data.

Examples

Returns the value of the variable R1.

```
VAR ("R1")
=> 1000.0
```

Returns the value of the variable *temp* in the specified directory.

```
VAR ("temp" "./simulation/test/spectre/schematic-save")
=> 27.0
```

DATA

```
DATA(  
    t_netName  
    t_analysis  
    [ t_dataDir ]  
)  
=> o_data / nil
```

Description

This is a basic data access function. It returns data for the specified node and analysis.

Arguments

<i>t_netName</i>	Net name
<i>t_analysis</i>	Name of the analysis.
<i>t_dataDir</i>	Directory containing the PSF files (results). When specified, this argument will only be used internally and will not alter the currently selected results. The default is the currently selected results directory.

Value Returned

<i>o_data</i>	The waveform for specified net and analysis.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

Returns the data for the specified node and analysis.

```
DATA("net1" "tran-tran")  
=> srrWave:49610776  
  
plot(DATA("net1" "ac-ac"))  
=> t (plot window comes up)
```

VS

```
VS (
    t_netName
    [ t_dataDir ]
)
=> o_wave / nil
```

Description

Returns dc sweep waveform for the specified net.

Arguments

<i>t_netName</i>	Net name
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>o_wave</i>	The dc sweep waveform for specified net.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
plot(VS("out"))
=> t (plot window comes up)
value(VS("net14") 4)
=> 1.3 (value at sweep variable = 4)
```

OP

```
OP (
    t_instanceName
    t_parameterName
    [ t_dataDir ]
)
=> n_number / nil
```

Description

Returns the value of the operating point parameter for the specified instance.

Arguments

<i>t_instanceName</i>	Instance name
<i>t_parameterName</i>	Parameter name
<i>t_dataDir</i>	Directory containing the PSF files (results). When specified, this argument will only be used internally and will not alter the currently selected results. The default is the currently selected results directory.

Value Returned

<i>n_number</i>	Returns value of the specified operating point.
<i>nil</i>	Returns <i>nil</i> if the design variable was not found.

Examples

```
OP("R1" "pwr")
=> 0.001
OP("R1" "pwr" "./simulation/test/spectre/schematic-save")
=> 0.002
```

OPT

```
OPT(  
    t_instanceName  
    t_parameterName  
    [ t_dataDir ]  
)  
=> n_number / nil
```

Description

Returns the transient operating point for the specified instance parameter.

Arguments

<i>t_instanceName</i>	Instance name
<i>t_parameterName</i>	Parameter name
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>n_number</i>	Returns the value of the specified operating point.
<i>nil</i>	Returns nil is the design variable was not found.

Examples

Returns the transient operating point for the instance parameter `pwr`.

```
OPT("R7" "pwr")  
=>0.0
```

Returns the transient operating point for the instance parameter `pwr` in the specified directory.

```
OPT("R7" "pwr" "./simulation/test/spectre/schematic-save")  
=>0.00025
```

MP

```
MP (
    t_instanceName
    t_parameterName
    [ t_dataDir ]
)
=> n_number / nil
```

Description

Returns the value of the specified model parameter for the given instance.

Arguments

<i>t_instanceName</i>	Instance name
<i>t_parameterName</i>	Parameter name
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results. The default is the currently selected results directory.

Value Returned

<i>n_number</i>	Returns value of the specified model parameter.
<i>nil</i>	Returns nil is the design variable was not found.

Examples

Returns the value of the specified model parameter `vpb` for the instance `I8.MP3`.

```
MP ("I8.M3" "vpb")
=> -0.13
```


NG

```
NG (
    [ t_dataDir ]
)
=> nw_noiseGain / nil
```

Description

Returns the noise gain waveform.

Arguments

<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>nw_noiseGain</i>	Returns the noise gain waveform.
<i>nil</i>	Returns <i>nil</i> if no data was found

Examples

```
NG ()
=> srrWave:34428932
NG("./simulation/test/spectre/schematic-save")
=> srrWave:23847792
```

VN

```
VN(  
    [ t_dataDir ]  
)  
=> nw_noise / nil
```

Description

Returns the noise waveform specified in V/sqrt (Hz).

Arguments

<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>nw_noise</i>	Returns the noise waveform.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
VN()  
=> srrWave:82374923  
VN("./simulation/test/spectre/schematic-save")  
=> srrWave:23749823
```

VN2

```
VN2 (
    [ t_dataDir ]
)
=> nw_noise / nil
```

Description

Returns the noise waveform in V²/Hz.

Arguments

<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results. The default is the currently selected results directory.

Value Returned

<i>nw_noise</i>	Returns the noise waveform.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
VN2 ()
=> srrWave:82374923
VN2("./simulation/test/spectre/schematic-save")
=> srrWave:23749823
```

VNP

```
VNP (
    t_name
    [ t_dataDir ]
)
=> g_value / nil
```

Description

Returns any single level noise parameter available in the PSF database.

Arguments

<i>t_name</i>	Name of the noise parameter.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results. The default is the currently selected results directory.

Value Returned

<i>g_value</i>	Value of the specified noise parameter.
<i>nil</i>	Specified noise parameter not found.

Examples

```
VNP("F")
=> srrWave:49610808
plot(VNP("NF"))
=> t (plot window comes up)
```

VNPP

```
VNPP (
    t_name
    t_param
    [ t_dataDir ]
)
=> g_value / nil
```

Description

VNPP accesses any double level noise parameter available in the PSF database.

Arguments

<i>t_name</i>	Name of the component.
<i>t_param</i>	Name of the noise parameter.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>g_value</i>	Value of the specified noise parameter.
<i>nil</i>	Specified noise parameter not found.

Examples

```
VNPP("R1" "fn")
=> srrWave:49610992
```

VPD

```
VPD(  
    t_net1  
    t_net2  
    [ t_dataDir ]  
)  
=> wave / nil
```

Description

Returns the waveform representing phase difference between voltages at the two nets.

Arguments

<i>t_net1</i>	Name of the 1st net.
<i>t_net2</i>	Name of 2nd net.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>wave</i>	Phase difference between voltages at the two nets.
<i>nil</i>	No data was found.

Examples

```
VPD("net1" "net2")  
=>srrWave:49610872
```

VF

```
VF(  
    t_netName  
    [ t_dataDir ]  
)  
=> wave / nil
```

Description

Returns the waveform representing the ac sweep net voltage.

Arguments

<i>t_netName</i>	Name of the net.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>wave</i>	Returns the AC sweep voltage for specified net.
<i>nil</i>	Returns <i>nil</i> if no data was found

Examples

```
VF("out")  
=> srrWave:78782738  
VF("out" "./simulation/test/spectre/schematic-save")  
=> srrWave:43985992
```

VT

```
VT(  
    t_netName  
    [ t_dataDir ]  
)  
=> wave / nil
```

Description

Returns the transient waveform for the specified net.

Arguments

<i>t_netName</i>	Name of the net.
<i>t_dataDir</i>	Directory containing the PSF files (results). When specified, this argument will only be used internally and will not alter the currently selected results. The default is the currently selected results directory.

Value Returned

<i>wave</i>	Returns the transient waveform for specified net.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
fam = VT("/out")  
=> srrWave:78782738  
drIsParamWave( fam )  
=> t
```


IS

```
IS (
    t_terminal
    [ t_dataDir ]
)
=> g_data / nil
```

Description

Returns waveform representing the dc sweep terminal current.

Arguments

<i>t_terminal</i>	Name of the terminal.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>g_data</i>	Returns the dc sweep current for specified terminal.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
IS ("/V0/PLUS")
=> srrWave:49090909

IS ("/V1/PLUS" "./simulation/test/spectre/schematic-save")
=> srrWave:40349095
```

IT

```
IT(  
    t_terminal  
    [ t_dataDir ]  
)  
=> data / nil
```

Description

Returns waveform representing the transient sweep terminal current.

Arguments

<i>t_terminal</i>	Name of the terminal.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>data</i>	Returns the transient current for specified terminal.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
IT("/V0/PLUS")  
=> srrWave:48729090  
IT("/V1/PLUS" "./simulation/test/spectre/schematic-save")  
=> srrWave:40345345
```

IF

```
IF(  
    t_terminal  
    [ t_dataDir ]  
)  
=> data / nil
```

Description

Returns the waveform representing the ac sweep terminal current.

Arguments

<i>t_terminal</i>	name of the terminal.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>data</i>	Returns the ac current for specified terminal.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
IF("/V0/PLUS")  
=> srrWave:49610872  
  
IF("/V1/PLUS" "./simulation/test/spectre/schematic-save")  
=> srrWave:44510345
```

IDC

```
IDC (
    t_terminal
    [ t_dataDir ]
)
=> data / nil
```

Description

Returns the waveform representing the DC terminal current.

Arguments

<i>t_terminal</i>	Name of the terminal.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>data</i>	Returns the dc current for specified terminal.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
IDC ("/V0/PLUS")
=> 3.143

IDC ("/V1/PLUS" "./simulation/test/spectre/schematic-save")
=> 4.23
```

VDC

```
VDC (
    t_netname
    [ t_dataDir ]
)
=> data / nil
```

Description

Returns the dc voltage for the specified net.

Arguments

<i>t_netname</i>	Name of the net.
<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>data</i>	Returns the dc voltage for specified net.
<i>nil</i>	Returns <i>nil</i> if no data was found.

Examples

```
VDC("net15")
=> 3.143

VDC("out" "./simulation/test/spectre/schematic-save")
=> 4.23
```

SIMULATOR

```
SIMULATOR(  
    [ t_dataDir ]  
)  
=> t_name / nil
```

Description

Returns the name of simulator.

Arguments

<i>t_dataDir</i>	Directory containing the PSF files(results). When specified, this argument will only be used internally and will not alter the currently selected results.The default is the currently selected results directory.

Value Returned

<i>t_name</i>	Name of the simulator.
<i>nil</i>	Returns <i>nil</i> if could not determine the simulator.

Examples

```
SIMULATOR()  
=> "spectre"  
  
SIMULATOR("./simulation/test/Spectre/schematic")  
=> "spectre"
```

Selection Functions

Selection functions let you specify schematic data in a type-in field. You give the designer the option of clicking on an item in the schematic window and having the information for that option automatically fill in the field. Typically, selection functions are used as callbacks in code that creates form fields and options.

The following example shows how you might use one of the selection functions as part of some code that creates a *net1* button with an associated type-in field.

```
hiCreateStringField(  
    ?name 'net1  
    ?prompt "Net Name"  
    ?value ""  
)  
hiCreateButton(  
    ?name 'netSelect1  
    ?buttonText "Select"  
    ?callback "asiSelectNet('net1 ?prompt \"Select net1\")"  
)
```

When the designer clicks on the button, the `asiSelectNet` function displays the *"Select net1"* prompt on the schematic window. The designer can click a net in the schematic window to automatically fill in the type-in field.

Note: Currently there is an *hi* limitation—if the debugger is installed for the session, the prompts are not displayed.

The next example shows how you might use one of the analysis-specific selection functions as part of some code that creates an *outVsrc* button with an associated type-in field. (This example does not show all the code for adding a noise analysis—it only shows how you might use the `asiSelectAnalysisSource` function.)

```
asiAddAnalysis( tool  
    ?name 'noise  
    ?prompt "Noise"  
    asiCreateAnalysisField(  
        ?name 'outVsrc  
        ?prompt "Output Voltage Source"  
    )  
    asiCreateAnalysisField(  
        ?name 'selectOutVsrc  
        ?prompt "Select"  
        ?type 'button  
        ?callback "asiSelectAnalysisSource( 'noise  
            'outVsrc \"Select output voltage  
            source ...\" )"  
    )  
)
```

When the designer clicks on the *Select* button, the `asiSelectAnalysisSource` function displays the *"Select output voltage source"* prompt on the schematic window. The designer can click a source in the schematic window to automatically fill in the type-in field.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Typically, the functions in this section are for integrators.

This topic lists the following SKILL functions that let designers set up forms.

[asiSelectAnalysisCompParam](#)

[asiSelectAnalysisInst](#)

[asiSelectAnalysisNet](#)

[asiSelectAnalysisSource](#)

[asiSelectInst](#)

[asiSelectNet](#)

[asiSelectSourceInst](#)

[asiSelectTerm](#)

[asiSelectTermNet](#)

Related Topics

[asiSelectNet](#)

[asiSelectAnalysisSource](#)

asiSelectAnalysisCompParam

```
asiSelectAnalysisCompParam(  
    s_analysisName  
    s_instField  
    s_parField  
)  
=> t / nil
```

Description

Lets the user select a component instance from the schematic to pop up a list box containing the parameters for that instance.

The name of the component automatically fills in the associated form field in the Choosing Analyses form. When the user selects a parameter from the list box, the name of the parameter automatically fills in its associated field.

Arguments

<i>s_analysisName</i>	Name of the analysis.
<i>s_instField</i>	Name of the field to contain the selected component.
<i>s_parField</i>	Name of the field to contain the selected parameter.

Value Returned

t	Successful operation
nil	Indicates an error.

Examples

Lets the user select a component instance from the schematic in an AC analysis. The name of the component fills in the *device* field, and a list box appears showing all the parameters for that component. Then the user can select a parameter from the list box to fill in the *deviceParam* field.

```
asiCreateAnalysisField(  
    ?name          'device  
    ?prompt        "Component Name"  
)  
asiCreateAnalysisField(  
    ?name          'select
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

```
        ?prompt      "Select Component"
        ?type        'button
        ?callback     "asiSelectAnalysisCompParam( 'ac 'device
                      'deviceParam )"
    )
asiCreateAnalysisField(
    ?name            'deviceParam
    ?prompt          "Parameter Name"
)
```

asiSelectAnalysisInst

```
asiSelectAnalysisInst(  
    s_analysis  
    s_field  
    [ t_prompt ]  
)  
=> t / nil
```

Description

Lets the user select an instance from the schematic to be used as input for the specified field for the specified analysis. If the user selects any other object, the system ignores the selection and beeps.

Arguments

<i>s_analysis</i>	Analysis type
<i>s_field</i>	Name of the analysis field.
<i>t_prompt</i>	Prompt string to display in the scheSuccessful operationmatic window. Default Value: "Select component..."

Value Returned

<i>t</i>	Successful operation
<i>nil</i>	Indicates an error.

Examples

Lets the user select an instance from the schematic to use as input for the *component* field for a DC analysis.

```
asiSelectAnalysisInst( 'dc 'component )
```

asiSelectAnalysisNet

```
asiSelectAnalysisNet(  
    s_analysis  
    s_field  
    [ t_prompt ]  
)  
=> t / nil
```

Description

Lets the user select a net from the schematic to be used as input for the specified field for the specified analysis. If the user selects any other object, the system ignores the selection and beeps.

Arguments

<i>s_analysis</i>	Analysis type
<i>s_field</i>	Name of the analysis field.
<i>t_prompt</i>	Prompt string to display in the schematic window. Default Value: "Select net..."

Value Returned

t	Successful operation
nil	Indicates an error.

Examples

Lets the user select a net from the schematic to use as input for the *outputNode* field for a *noise* analysis.

```
asiSelectAnalysisNet( 'noise 'outputNode)
```

Lets the user select a net from the schematic to use as input for the *posNode* field for an *xf* analysis.

```
asiSelectAnalysisNet( 'xf 'posNode)
```

asiSelectAnalysisSource

```
asiSelectAnalysisSource(  
    s_analysis  
    s_field  
    [ t_prompt ]  
)  
=> t / nil
```

Description

Lets the user select a source from the schematic to be used as input for the specified field for the specified analysis. If the user selects any other object, the system ignores the selection and beeps.

Note: The property source should be set to *t* in the simInfo of the component to be selected with this function.

Arguments

<i>s_analysis</i>	Analysis type
<i>s_field</i>	Name of the analysis field.
<i>t_prompt</i>	Prompt string to display in the schematic window. Default Value: "Select source..."

Value Returned

<i>t</i>	Successful operation
nil	Indicates an error.

Examples

Lets the user select a source from the schematic to use as input for the *outVsrc* field for a noise analysis.

```
asiSelectAnalysisSource( 'noise 'outVsrc "Select output voltage source ..." )  
procedure( almCreateSimInfo_<comp>_<yourSimulator>()  
    '( nil  
        netlistProcedure      comp  
        otherParameters      ( fundname noisefile  
                               FNpairs F1 N1 F2 N2 F3 N3 F4 N4 F5 N5  
                               F6 N6 F7 N7 F8 N8 F9 N9 F10 N10 )
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

```
instParameters      ( dc mag phase type xfmag pacmag pacphase val0
                    vall period delay rise fall width tc1 tc2
                    tnom )
termOrder           ( PLUS MINUS )
termMapping         ( nil PLUS \:p MINUS "(FUNCTION minus(root('PLUS')))" )
propMapping         ( nil dc vdc mag acm phase acp val0 v1 vall v2
                    period per delay td rise tr fall tf
                    width pw xfmag xfm pacmag pacm
                    pacphase pacp type srcType )
componentName       vsource
source             t
)
)
```

asiSelectInst

```
asiSelectInst(  
    s_field  
    [ ?prompt t_prompt ]  
    [ ?form r_form ]  
    [ ?tab l_tab ]  
)  
=> t / nil
```

Description

Lets you select an instance from the schematic. If you select any other object, the system beeps and ignores the selection.

Arguments

<code>s_field</code>	Name of the field to be filled in with the selected value.
<code>?prompt t_prompt</code>	Prompt string to display in the schematic window. Default Value: "Select instance..."
<code>?form r_form</code>	Form that contains the field to be filled in with the selected value. The default is the form returned by <code>hiGetCurrentForm</code> .
<code>?tab l_tab</code>	List containing the following: ■ Name of the tab, which is created using <code>hiCreateTabField</code>. ■ Page displayed on the specified tab on which the string field exists. Note: Use this argument only in case the specified form has multiple tabs.

Value Returned

<code>t</code>	Successful operation
<code>nil</code>	Indicates an error.

Examples

Lets you select an instance from the schematic to use as input for the *inst1* field of the current form. It is used when the form does not contain any tabs.

```
asiSelectInst( 'inst1 )
```

Lets you select an instance *inst1* which is present in *page1* of `TabField`. It is used when a form contains multiple tabs.

```
asiSelectInst('inst1 ?tab list('TabField 'page1))
```

Related Topics

[hiCreateTabField](#)

[hiGetCurrentForm](#)

asiSelectNet

```
asiSelectNet(  
    s_field  
    [ ?prompt t_prompt ]  
    [ ?form r_form ]  
    [ ?tab l_tab ]  
)  
=> t / nil
```

Description

Lets you select a net from the schematic. If you select any other object, the system ignores the selection and beeps.

Arguments

<i>s_field</i>	Name of the field to be filled in with the selected value.
<i>?prompt t_prompt</i>	Prompt string to display in the schematic window. Default Value: "Select net..."
<i>?form r_form</i>	Form that contains the field to be filled in with the selected value. The default is the form returned by <code>hiGetCurrentForm</code> .
<i>?tab l_tab</i>	List containing the following: ■ Name of the tab, which is created using <code>hiCreateTabField</code>. ■ Page displayed on the specified tab on which the string field exists. Note: Use this argument only in case the specified form has multiple tabs.

Value Returned

<i>t</i>	Successful operation
<i>nil</i>	Indicates an error.

Examples

Lets you select a net from the schematic to use as input for the *net1* field of the current form. Use this when the form does not contain any tabs.

```
asiSelectNet( 'net1 '
```

Lets you select a net from the schematic to use as an input for the *net1* field which is present in *page1* of *TabField*. Use this when a form contains multiple tabs.

```
asiSelectNet('net1 ?tab list('TabField 'page1'))
```

Related Topics

[hiCreateTabField](#)

[hiGetCurrentForm](#)

asiSelectSourceInst

```
asiSelectSourceInst(  
    s_field  
    [ ?prompt t_prompt ]  
    [ ?form r_form ]  
    [ ?tab l_tab ]  
)  
=> t / nil
```

Description

Lets you select a source instance on the schematic. If you select any other object, the system beeps and ignores the selection.

Arguments

<code>s_field</code>	Name of the field to be filled in with the selected value.
<code>?prompt t_prompt</code>	Prompt string to display in the schematic window. Default Value: "Select source..."
<code>?form r_form</code>	Form that contains the field to be filled in with the selected value. The default is the form returned by <code>hiGetCurrentForm</code> .
<code>?tab l_tab</code>	List containing the following: ■ Name of the tab, which is created using <code>hiCreateTabField</code>. ■ Page displayed on the specified tab on which the string field exists. Note: Use this argument only in case the specified form has multiple tabs.

Value Returned

<code>t</code>	Successful operation
<code>nil</code>	Indicates an error.

Examples

Lets you select a source instance from the schematic to use as input for the *srcInst* field of the current form. Use this when the form does not contain any tabs.

```
asiSelectSourceInst( 'srcInst ?prompt "Select voltage source or current source" )
```

Lets you select a source instance from the schematic to use as input for the *srcInst* field present in `page1` of `TabField`. Use this when a form contains multiple tabs.

```
asiSelectSourceInst( 'srcInst ?tab list('TabField 'page1) ?prompt "Select voltage source or current source")
```

Related Topics

[hiCreateTabField](#)

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

hiGetCurrentForm

asiSelectTerm

```
asiSelectTerm(  
    s_field  
    [ ?prompt t_prompt ]  
    [ ?form r_form ]  
    [ ?tab l_tab ]  
)  
=> t / nil
```

Description

Lets you select an instance terminal from the schematic. If you select any other object, the system ignores the selection and beeps.

Arguments

<code>s_field</code>	Name of the field to be filled in with the selected value.
<code>?prompt t_prompt</code>	Prompt string to display in the schematic window. Default Value: "Select instance terminal..."
<code>?form r_form</code>	Form that contains the field to be filled in with the selected value. The default is the form returned by <code>hiGetCurrentForm</code> .
<code>?tab l_tab</code>	List containing the following: ■ Name of the tab, which is created using <code>hiCreateTabField</code> ■ Page displayed on the specified tab on which the string field exists. Note: Use this argument only in case the specified form has multiple tabs.

Value Returned

<code>t</code>	Successful operation
<code>nil</code>	Indicates an error.

Examples

Lets you select the instance terminal from the schematic to use as input for the *term* field of the current form. Use this when a form does not contain any tabs.

```
asiSelectTerm( 'term )
```

Lets you select an instance terminal from the schematic to use as an input for the *term* field which is present in *page1* of *TabField*. Use this when a form contains multiple tabs.

```
asiSelectTerm('term ?tab list('TabField 'page1))
```

Related Topics

[hiCreateTabField](#)

[hiGetCurrentForm](#)

asiSelectTermNet

```
asiSelectTermNet(  
    s_field  
    [ ?prompt t_prompt ]  
    [ ?form r_form ]  
    [ ?tab l_tab ]  
)  
=> t / nil
```

Description

Lets you select either an instance terminal or a net. If you select any other object, the system beeps and ignores the selection.

Arguments

<code>s_field</code>	Name of the field to be filled in with the selected value.
<code>?prompt t_prompt</code>	Prompt string to display in the schematic window. Default Value: "Select terminal or net..."
<code>?form r_form</code>	Form that contains the field to be filled in with the selected value. The default is the form returned by <code>hiGetCurrentForm</code> .
<code>?tab l_tab</code>	List containing the following: ■ Name of the tab, which is created using <code>hiCreateTabField</code>. ■ Page displayed on the specified tab on which the string field exists. Note: Use this argument only in case the specified form has multiple tabs.

Value Returned

<code>t</code>	Successful operation
<code>nil</code>	Indicates an error.

Examples

Lets you select an instance terminal or net from the schematic to use as input for the *termNet* field of the current form. Use this when a form does not contain any tabs.

```
asiSelectTermNet('termNet')
```

Lets you select an instance terminal or net from the schematic to use as an input for the *termNet* field which is present in `page1` of `TabField`. Use this when a form contains multiple tabs.

```
asiSelectTermNet('termNet' ?tab list('TabField' 'page1'))
```

Related Topics

[hiCreateTabField](#)

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

hiGetCurrentForm

Waveform Data Object Functions

Waveform data objects are used in simulation results manipulation and display. Waveform data objects are represented by the special type *srrWaveID*. Each ID uniquely identifies a waveform data object. A waveform data object has two data-vectors, the X and the Y data vector. Similar to waveform data objects, data vectors are represented by the special type *srrVecID*. Each ID uniquely identifies a data vector.

Note: SKILL functions, such as *declare()* and *makeVector()*, return a vector object, which is conceptually similar to but structurally different from the vector returned by public APIs, such as *drCreateVec()*. The former is a (SKILL) language-supported data structure. The latter is an application-supported data structure, for example, the data vector used in functions. *srrWaveXXXX* public APIs require the application-supported vector type, not the SKILL vector type or array.

Data Values

All the values in a data vector must be of the same data type. However, the X and Y vectors in a waveform data object can be of different data types. The data types and their symbolic representation that are supported for Data Vectors in SKILL are:

- 'intlong for integer numbers
- 'double for double-precision floating-point numbers
- 'doublecomplex for complex numbers
- 'string for strings

When you create a new waveform data object, you must first create the X and Y vectors with *drCreateVec*. You can use *drCreateWaveform* with the X and the Y vectors as arguments to create a waveform.

This topic lists all the SKILL functions that let you create, modify, and analyze waveform data objects.

[drAddElem](#)

[drGetElem](#)

[drSetElem](#)

[drCreateVec](#)

[drCreateEmptyWaveform](#)

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

drCreateWaveform

drGetWaveformXType

drGetWaveformXVec

drGetWaveformYType

drGetWaveformYVec

drPutWaveformXVec

drPutWaveformYVec

drIsDataVector

drIsParamWave

drIsWaveform

drType

drVectorLength

famAddValue

famCreateFamily

famGetSweepName

famGetSweepValues

famIsFamily

famMap

famValue

drAddElem

```
drAddElem(  
    o_vec  
    g_value  
)  
=> t / nil
```

Description

Puts *g_value* after the last element in the data vector *o_vec*. *g_value* must have the same data type as the data vector.

Arguments

<i>o_vec</i>	Data vector to which a new element <i>g_value</i> will be appended.
<i>g_value</i>	New element to be added at end of data vector <i>o_vec</i> .

Value Returned

<i>t</i>	Returns <i>t</i> if element is added successfully.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

```
vec = drCreateVec('double 5)  
drAddElem(vec 10.0)
```

drGetElem

```
drGetElem(  
    o_vec  
    x_index  
)  
=> g_result
```

Description

Returns the x_index^{th} element of the data vector *drVecID*, assuming a zero-based index.

Arguments

<i>o_vec</i>	Target data vector
<i>x_index</i>	Target element of data vector <i>drVecID</i> .

Value Returned

<i>g_result</i>	Returns the contents of element <i>x_index</i> of data vector <i>o_vec</i> .

Examples

```
vec = drCreateVec( 'double '(1 2 3 4 5))  
drGetElem(vec 2)
```

drSetElem

```
drSetElem(  
    o_vec  
    x_index  
    g_value  
)  
=> t / nil
```

Description

Replaces the *x_index*th element of the data vector *o_vec* with *g_value*. *g_value* must have the same data type as the data vector.

Arguments

<i>o_vec</i>	Data vector to which a new element <i>g_value</i> will be appended.
<i>x_index</i>	The index for which the value is replaced.
<i>g_value</i>	New element to be added at end of data vector <i>o_vec</i> .

Value Returned

<i>t</i>	Returns <i>t</i> if the element is replaced.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

```
vec = drCreateVec('double 5)  
drSetElem(vec 0 10.0)
```

drCreateVec

```
drCreateVec(  
    s_dataType  
    [ x_length]  
    [ l_values ]  
)  
=> o_vec / nil
```

Description

Creates a new data vector.

Arguments

<i>s_dataType</i>	Data type. Possible values are described in Data Values .
<i>x_length</i>	Length of the vector. The value of this argument must be greater than 0. When using this argument, you cannot use the <i>l_values</i> argument.
<i>l_values</i>	List of values. When using this argument, you cannot use the <i>x_length</i> argument.

Value Returned

<i>o_vec</i>	Returns the created vector if successful.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

```
vec = drCreateVec('double 5)
```


drCreateEmptyWaveform

```
drCreateEmptyWaveform(  
    )  
    => o_waveform
```

Description

Creates an empty waveform data object.

Arguments

None

Value Returned

<i>o_waveform</i>	Returns the waveform object created.

Examples

```
drCreateEmptyWaveform()
```

drCreateWaveform

```
drCreateWaveform(  
    o_xvec  
    o_yvec  
)  
=> o_wave / nil
```

Description

Creates a waveform data object with the vectors specified.

Arguments

<i>o_xvec</i>	X vector
<i>o_yvec</i>	Y vector

Value Returned

<i>o_wave</i>	Returns the created waveform if successful.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

```
wave = drCreateWaveform( drCreateVec( 'double 5 ) drCreateVec( 'double 5 ) )
```

drGetWaveformXType

```
drGetWaveformXType (
    o_wave
)
=> s_dataType
```

Description

Returns the X vector data type of the waveform data object *o_wave*.

Arguments

<i>o_wave</i>	Waveform data object.

Value Returned

<i>s_dataType</i>	Returns the symbol indicating X vector data type as described Data Values .

Examples

```
wave = drCreateWaveform( drCreateVec( 'double 5 ) drCreateVec( 'double 5 ) )
drGetWaveformXType( wave )
```

drGetWaveformXVec

```
drGetWaveformXVec (
    o_wave
)
=> o_vec
```

Description

Returns the X vector object ID of the waveform data object *o_wave*.

Arguments

<i>o_wave</i>	Waveform data object.

Value Returned

<i>o_vec</i>	Returns the X vector object ID.

Examples

```
wave = drCreateWaveform( drCreateVec( 'double 5 ) drCreateVec( 'double 5 ) )
drGetWaveformXVec( wave )
```

drGetWaveformYType

```
drGetWaveformYType (
    o_wave
)
=> s_dataType
```

Description

Returns the Y vector data type of the waveform data object *o_wave*.

Arguments

<i>o_wave</i>	Waveform data object.

Value Returned

<i>s_dataType</i>	Returns the symbol indicating Y vector data type as described in Data Values .

Example

```
wave = drCreateWaveform( drCreateVec( 'double 5 ) drCreateVec( 'double 5 ) )
drGetWaveformYType( wave )
```

drGetWaveformYVec

```
drGetWaveformYVec (
    o_wave
)
=> o_vec
```

Description

Returns the Y vector object ID of the waveform data object *o_wave*.

Arguments

<i>o_wave</i>	Waveform data object.

Value Returned

<i>o_vec</i>	Returns the Y vector object ID.

Examples

```
wave = drCreateWaveform( drCreateVec( 'double 5 ) drCreateVec( 'double 5 ) )
drGetWaveformYVec( wave )
```

drPutWaveformXVec

```
drPutWaveformXVec (
    o_wave
    o_vec
)
=> t / nil
```

Description

Puts the data vector *o_vec* into the waveform data object *o_wave*. *o_vec* is the X vector of the waveform data object. If the *o_wave* already contains a Y vector, the length of *o_vec* must be the same as that of the Y vector.

Arguments

<i>o_wave</i>	Waveform data object.
<i>o_vec</i>	The new X data vector.

Value Returned

<i>t</i>	Returns <i>t</i> if data vector is added successfully.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

```
wave = drCreateEmptyWaveform()
vec = drCreateVec( 'double ' (1 2 3 4 5))
drPutWaveformXVec( wave vec )
```

drPutWaveformYVec

```
drPutWaveformYVec(  
    o_wave  
    o_vec  
)  
=> t / nil
```

Description

Puts the data vector *o_vec* into the waveform data object *o_wave*. *o_vec* is the Y vector of the waveform data object. If the *o_wave* already contains a X vector, the length of *o_vec* must be the same as that of the X vector.

Arguments

<i>o_wave</i>	Waveform data object.
<i>o_vec</i>	The new Y data vector.

Value Returned

<i>t</i>	Returns <i>t</i> if data vector is added successfully.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

```
wave = drCreateEmptyWaveform()  
vec = drCreateVec( 'double ' (1 2 3 4 5))  
drPutWaveformYVec( wave vec )
```


drIsDataVector

```
drIsDataVector(  
    g_value  
)  
=> t / nil
```

Description

Returns *t* when *g_value* is a valid *drVector* data object and *nil* if it is not one.

Arguments

<i>g_value</i>	Value checked.

Value Returned

<i>t</i>	Returns <i>t</i> when <i>g_value</i> is a <i>drVector</i> data object.
<i>nil</i>	Returns <i>nil</i> if <i>g_value</i> is not a <i>drVector</i> data object.

Examples

```
wave = drCreateEmptyWaveform()  
drIsDataVector( wave )  
=> nil
```

drIsParamWave

```
drIsParamWave(  
    g_value  
)  
=> t / nil
```

Description

Returns *t* when *g_value* is a family, for example a parametric wave.

Arguments

<i>g_value</i>	Value checked.

Value Returned

<i>t</i>	Returns <i>t</i> when <i>g_value</i> is a family.
<i>nil</i>	Returns <i>nil</i> if <i>g_value</i> is not a family.

Examples

Evaluates any net expression after running a Parametric Analysis, such as:

```
fam = VT("/out")  
drIsParamWave( fam )  
=> t  
  
wave = drCreateEmptyWaveform()  
drIsParamWave( wave )  
=> nil
```

drIsWaveform

```
drIsWaveform(  
    g_value  
)  
=> t / nil
```

Description

Returns *t* when *g_value* is a waveform data object and *nil* if it is a family.

Arguments

<i>g_value</i>	Value checked.

Value Returned

<i>t</i>	Returns <i>t</i> when <i>g_value</i> is a waveform data object.
<i>nil</i>	Returns <i>nil</i> if <i>g_value</i> is not a waveform data object.

Examples

```
wave = drCreateEmptyWaveform()  
drIsWaveform( wave )  
=> nil
```

drType

```
drType (
    o_vec
)
=> s_type
```

Description

Returns the data type of the data vector *o_vec*.

Arguments

<i>o_vec</i>	The vector for which the data type is returned.

Value Returned

<i>s_type</i>	Returns a symbolic representation of the type of the data vector. The values are described Data Values .

Examples

```
vec = drCreateVec( 'double ' (1 2 3 4 5))
drType( vec )
```

drVectorLength

```
drVectorLength(  
    o_vec  
)  
=> x_length
```

Description

Returns the length of the data vector *o_vec*.

Arguments

<i>o_vec</i>	Target data vector

Value Returned

<i>x_length</i>	Returns the length of data vector <i>o_vec</i> .

Examples

```
vec = drCreateVec( 'double ' (1 2 3 4 5))  
drVectorLength( vec )
```

famAddValue

```
famAddValue(  
    o_family  
    g_sweepValue  
    g_value  
)  
=> o_family
```

Description

Adds a waveform associated with *sweepValue* to a family if values are specified for both *sweepValue* and *value*.

Arguments

<i>o_family</i>	The family to receive the new value.
<i>g_sweepValue</i>	The sweep value with which to associate the new waveform.
<i>g_value</i>	The new waveform to add.

Value Returned

<i>o_family</i>	The modified family.

Examples

```
fam = famCreateFamily( "prf" 'double )  
=> srrWave:17518616  
  
famAddValue( fam -15 wave15 )  
=> srrWave:17518616  
  
famAddValue( fam -10 wave10 )  
=> srrWave:17518616  
  
famAddValue( fam -5 wave5 )  
=> srrWave:17518616
```

famCreateFamily

```
famCreateFamily(  
    s_sweepName  
    s_varType  
)  
=> o_family
```

Description

Creates a empty data structure called a family. When filled with data, it is suitable to be plotted as a family of curves. Each waveform has a name given by *s_varName* and the values are added with `famAddValue`.

Arguments

<i>s_sweepName</i>	The string name of the sweep variable.
<i>s_varType</i>	One of the following variable types: 'double, 'doublecomplex, 'float, 'intlong, 'signal, 'string. The <i>varType</i> of 'float is equivalent to 'double and 'signal is equivalent to 'string.

Value Returned

<i>o_family</i>	An empty family with the variable label <i>s_var</i> .

Examples

```
fam = famCreateFamily( "prf" 'double )  
=> srrWave:17518616  
  
famAddValue( fam -15 wave15 )  
=> srrWave:17518616  
  
famAddValue( fam -10 wave10 )  
=> srrWave:17518616  
  
famAddValue( fam -5 wave5 )  
=> srrWave:17518616
```

Related Topics

[famAddValue](#)

famGetSweepName

```
famGetSweepName (
    o_family
    [ x_dim ]
)
=> s_sweepName
```

Description

Returns the name of the sweep variable of the parametric waveform and the dimension supplied.

Arguments

<i>o_family</i>	A family name
<i>x_dim</i>	An integer that specifies how deep the function should go before returning the sweep name.

Value Returned

<i>s_sweepName</i>	The string name of the sweep variable.

Examples

```
fam1 = famCreateFamily( "prf" 'double )
=> srrWave:29335584

fam2 = famCreateFamily( "prf" 'double )
=> srrWave:29335592

fam3 = famCreateFamily( "ofreq" 'double )
=> srrWave:29335600

famAddValue( fam3 999M fam1)
=> srrWave:29335600

famAddValue( fam3 1G fam2)
=> srrWave:29335600

famGetSweepName( fam3 )
=> "ofreq"

famGetSweepName( fam3 0 )
=> "ofreq"

famGetSweepName( fam3 1 )
=> "prf"
```


famGetSweepValues

```
famGetSweepValues(  
    o_family  
)  
=> l_values
```

Description

Returns the values of the sweep variable of the family specified. The returned list is sorted in increasing order.

Arguments

<i>o_family</i>	A family name

Value Returned

<i>l_values</i>	A list of values of the sweep variable.

Examples

```
foreach( v famGetSweepValues( fam )  
printf( "%L\n" v))
```

famIsFamily

```
famIsFamily(  
    g_arg  
)  
=> t / nil
```

Description

Checks whether the argument specified is a family with at least one waveform.

Arguments

<i>g_arg</i>	Any expression

Value Returned

<i>t</i>	Returns <i>t</i> when <i>g_arg</i> is a family.
<i>nil</i>	Returns <i>nil</i> if <i>g_arg</i> is not a family.

Examples

```
famIsFamily( 3 )  
=> nil  
  
fam1 = famCreateFamily( "prf" 'double )  
=> srrWave:50152744  
  
famIsFamily( fam1 )  
=> nil  
  
famAddValue( fam1 -10 drCreateEmptyWaveform() )  
=> srrWave:50152744  
  
famIsFamily( fam1 )  
=> t
```

famMap

```
famMap(  
    s_func  
    o_family  
    [ @rest t_args ]  
)  
=> o_result
```

Description

Applies a function with a set of arguments to each member of a family of waveforms.

Arguments

<i>s_func</i>	A function specified as a string or symbol.
<i>o_family</i>	A family
<i>@rest t_args</i>	

Value Returned

<i>o_result</i>	A family with a structure similar to the input family.

Examples

```
;; After running parametric pnoise analysis  
wave1 = getData( "out" ?result 'pnoise )  
srrWave:50152592  
ocnPrint( wave1 )  
  
freq (Hz)  
quotient (getD      quotient (getD  
prf      -10      -9  
901M      2.39482n      2.6192n  
902M      2.04681n      2.1431n  
903M      1.84256n      1.89187n  
904M      1.75846n      1.78815n  
905M      1.71612n      1.73575n  
  
=> t  
  
ocnPrint( famMap( 'quotient wave1 2.0 ))  
freq (Hz)      quotient (getD      quotient (getD  
prf      -10      -9  
  
901M      1.19741n      1.3096n  
902M      1.02341n      1.0715n  
903M      0.92128n      0.94594n
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

```
904M          0.87923n          0.89408n
905M          0.85806n          0.86788n
=> t
;; This is equivalent to ocnPrint( wave1 | 2.0 )
```

famValue

```
famValue(  
    o_family  
    g_sweepValue  
)  
=> o_waveformOrFamily
```

Description

Returns the waveform whose *sweepName* has the value specified using *sweepValue*.

Arguments

<i>o_family</i>	A family of waveforms.
<i>o_sweepValue</i>	The value of sweep to retrieve. This must be an exact match with the value specified using <i>famAddValue</i> .

Value Returned

<i>o_waveformOrFamily</i>	The waveform or family whose sweep value was specified in the argument.

Examples

```
wave1  
srrWave:29298888  
  
wave2  
srrWave:29298896  
  
famStr = famCreateFamily( "name" 'string )  
=> srrWave:29298920  
  
famAddValue( famStr "one" wave1 )  
=> srrWave:29298920  
  
famAddValue( famStr "two" wave2 )  
=> srrWave:29298920  
  
famValue( famStr "one")  
=> srrWave:29298968  
;; although it is a differen waveform it is equivalent.  
  
equal( wave1 famValue( famStr "one" ))  
=> t  
  
famValue( famStr "three" )  
=> nil
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

```
famNum = famCreateFamily( "prf" 'double )
=> srrWave:29299008

famAddValue(famNum 1.0 wave1)
=> srrWave:29299008

famAddValue( famNum 2.0 wave2 )
=> srrWave:29299008

famValue( famNum 1.5 )
=> srrWave:29299056    ;; interpolated waveform is returned
```

Related Topics

[famAddValue](#)

Direct Plot Functions for RF Analyses

The direct plot functions calculate the simulation result of some pre-defined analysis. This topic lists all SKILL functions that let you calculate simulation results for RF circuits only.

[drplMcpValue](#)

[drplWrIsAcprValue](#)

[drplEvmWrIs](#)

[drplACPRWithMask](#)

[drplEvmBpsk](#)

[drplPacVolGnExpDen](#)

[drplJitter](#)

[drplRFJc](#)

[drplRFJcc](#)

[drplParamSweepRFJc](#)

[drplParamSweepRFJcc](#)

[drplRFValueAt](#)

[drplSwpHp](#)

[drplSwpSp](#)

[drplSwpYp](#)

[drplSwpZm](#)

[drplSwpZp](#)

[drplSprobeHPM](#)

[drplSprobePortSPData](#)

[drplSprobePortVSWR](#)

[drplSprobeYPM](#)

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

[drplSprobeZM](#)

[drplSprobeZPM](#)

[drplGetMeas](#)

[drplGetSpectrum](#)

drplMcpValue

```
drplMcpValue(  
    s_probe  
)  
=> f_power
```

Description

Returns the main channel power value.

Arguments

<i>s_probe</i>	The name of the probe for which the main channel power needs to be calculated.

Value Returned

<i>f_power</i>	The main channel power value.

Examples

Returns the main channel power for the WPRB0 probe:

```
drplMcpValue('WPRB0')  
=> 6.87651
```

drplWrlsAcprValue

```
drplWrlsAcprValue(  
    s_probe  
    n_idx  
)  
=> f_acpr
```

Description

Returns the adjacent channel power ratio value.

Arguments

<i>s_probe</i>	The name of the probe for which the adjacent channel power ratio needs to be calculated.
<i>n_idx</i>	The index of symbol used for channel estimation.

Value Returned

<i>f_acpr</i>	The adjacent channel power ratio.

Examples

Returns the adjacent channel power ratio for the WPRB0 probe:

```
drplWrlsAcprValue('WPRB0 ?idx 1')  
=> -25.63478
```

drplEvmWrIs

```
drplEvmWrIs(  
    s_probe  
    g_percent  
    g_sweepValue  
)  
=> o_waveform
```

Description

Displays the error vector magnitude value on the constellation plot.

Arguments

<i>s_probe</i>	The name of the probe for which the error vector magnitude needs to be calculated.
<i>g_percent</i>	Specify whether to select the percent format or not.
<i>g_sweepValue</i>	The value of sweep to retrieve.

Value Returned

<i>o_waveform</i>	Returns the waveform object representing the error magnitude.

Examples

Returns the error vector magnitude for the WPRB0.mea probe:

```
drplEvmWrIs("WPRB0.mea" ?percent t ?sweepValue nil)  
=> srrWave:0x340d42a0
```

drplACPRWithMask

```
drplACPRWithMask(  
    t_acprw  
    t_sig  
)  
=> o_waveform / nil
```

Description

This function plots `acpr` and `spectrum` masks if you select one of standards defined in Channel Definitions on `envlp` result Direct Plot form. These masks confirm that the `acpr` reaches the mask requirement defined in communication standards.

Arguments

<code>t_acprw</code>	Name of the <code>acpr</code> wave.
<code>t_sig</code>	The signal standard, such as <code>802_11a</code> .

Value Returned

<code>o_waveform</code>	Returns the waveform of <code>acpr</code> and mask.
<code>nil</code>	Returns <code>nil</code> , if result <code>envlp_fd</code> does not exist or power spectral density can not be calculated.

Examples

Plots `rRvalue` on the Y-axis.

```
drplACPRWithMask(db10(psd(bbb(real(harmonic(v("/net9" ?result "envlp_fd") '1))  
imag(harmonic(v("/net9" ?result "envlp_fd") '1)) 0 0.00016 12800 ?windowSize 64  
?windowName "Cosine4" ?detrending "none")) "802_11a"))
```

drplEvmBpsk

```
drplEvmBpsk(  
    o_waveform1  
    o_waveform2  
    n_tDelay  
    n_sampling  
    b_autoLevelDetect  
    n_voltage  
    n_offset  
    b_normalize  
    [ ?percent b_percent ]  
)  
=> o_waveform / nil
```

Description

Processes the I and Q waveform outputs from the transient simulation run to calculate the Error Vector Magnitude (EVM) and plot the I versus Q scatterplot. EVM is a useful measurement to describe the overall signal amplitude and phase modulated signal quality. It is based on a statistical error distribution normalized from an ideal digital modulation. Binary Phase Shift Keying (BPSK) is a typical modulation scheme where EVM is useful. The EVM is calculated by detecting the I and Q signal levels corresponding to the two possible I and Q symbol combinations, and calculating the difference between the signal level and the ideal signal level. Compared to other types of phase shift keying, such as QPSK, QAM16 and QAM64, BPSK has lowest bit error rate for the same signal to noise ratio.

Note: This function is not supported for family of waveforms.

Arguments

<i>o_waveform1</i>	The waveform for I signal.
<i>o_waveform2</i>	The waveform for Q signal.
<i>n_tDelay</i>	The start time for the first valid symbol. This can be obtained from the <i>Waveform Viewer</i> window by recording the time of the first minimum or maximum of the valid symbol (whichever is earlier), on the selected signal stream.
<i>n_sampling</i>	A period for the symbol. Each period is represented by a data rate. The data rate at the output is determined by the particular modulation scheme being used.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

<i>b_autoLevelDetect</i>	Indicates that you want the amplitude, <i>n_voltage</i> , and DC offset, <i>n_offset</i> , to be calculated automatically. Amplitude is calculated by averaging the rectified voltage level of the signal streams. DC offset is calculated by averaging the sum of an equal number of positive and negative symbols in each signal stream. These values are used to determine the EVM value. If the <i>b_autoLevelDetect</i> is <i>nil</i> , the values for <i>n_voltage</i> and <i>n_offset</i> are required. Valid values: <i>t</i> , <i>nil</i> Default value: <i>t</i>
<i>n_voltage</i>	The amplitude of the signal.
<i>n_offset</i>	The DC offset value.
<i>b_normalize</i>	An option to see the scatter plot normalized to the ideal values +1 and -1 (for example, when superimposing scatter plots from different stages in the signal flow, where the levels may be different, but you want to see relative degradation or improvement in the scatter). This option does not affect the calculation of the EVM number. Valid values: <i>t</i> , <i>nil</i> Default value: <i>nil</i>
<i>?percent b_percent</i>	Specifies whether to print the average EVM in percentage or in dB scale. If the value is set to <i>t</i> , average EVM is printed in percentage, otherwise, average EVM is printed in dB scale. Valid values: <i>t</i> , <i>nil</i> Default value: <i>nil</i>

Value Returned

<i>o_waveform</i>	Returns a waveform object representing the EVM value computed from the waveforms.
<i>nil</i>	Returns <i>nil</i> and prints the error message if the function is unsuccessful.

Examples

Calculates the EVM value when *b_autoLevelDetect* is set to *t* . In this case, values are not specified for *n_voltage* and *n_offset*.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

```
drplEvmBpsk( v("samp_out_Q"), v("samp_out_I") 1.5u, 181.81n, t, nil, nil, nil)
```

Calculates the EVM value when `b_autoLevelDetect` is set to `nil`. In this case, values are specified for `n_voltage` and `n_offset`.

```
drplEvmBpsk( v("samp_out_Q"), v("samp_out_I") 1.5u, 181.81n, nil, 1.3, 0, nil)
```

drplPacVolGnExpDen

```
drplPacVolGnExpDen(  
    t_denSigStr  
    rh  
    t_name  
)  
=> o_waveform / nil
```

Description

Plots the PAC voltage wave.

Arguments

<i>t_denSigStr</i>	The defined signal string.
<i>rh</i>	The reference harmonic.
<i>t_name</i>	Name.

Value Returned

<i>o_waveform</i>	Returns the waveform of PAC voltage.
<i>nil</i>	Returns <i>nil</i> , otherwise.

Examples

```
drplPacVolGnExpDen("v(\"/VLO\" ?result \"hbac\")" ' (0) nil)
```


drplJitter

```
drplJitter(  
  [ ?result t_result ]  
  [ ?resultsDir t_resultsDir ]  
  [ ?freq n_freq ]  
  [ ?k n_k ]  
  [ ?unit t_unit ]  
  [ ?ber g_ber ]  
  [ ?event n_event ]  
)  
=> value / nil
```

Description

Calculates jitter from the result of pnoise analysis, where *noisetype* is set as `pmjitter`.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Arguments

<code>?result t_result</code>	Name of the result of pnoise analysis. By default, the name for the result of pnoise jitter analysis is <code>pnoise_pmjitter</code> .
<code>?resultsDir t_resultsDir</code>	Path to the results directory.
<code>?freq n_freq</code>	Frequency at which you want to calculate jitter. This value is used in case of an autonomous circuit only. Specify it as <code>nil</code> , in case of a driven circuit.
<code>?k n_k</code>	The number of cycles. The default value of <code>k</code> is 1.
<code>?unit t_unit</code>	Unit to measure jitters. Possible values are <code>Second</code> , <code>UI</code> , and <code>ppm</code> .
<code>?ber n_ber</code>	The value of BER (Bit Error Rate) when the signal level is peak-to-peak.
<code>?event n_event</code>	Index of an event in the result. The jitter value is calculated for each event.

Value Returned

<code>value</code>	Returns the jitter value.
<code>nil</code>	Returns <code>nil</code> , if the result name is incorrect and jitter cannot be calculated.

Examples

Returns a jitter value.

```
drplJitter(?result "pnoise_pmjitter" ?unit "Second" ?event 3 ?k 1)
```

drplRFJc

```
drplRFJc(  
    [ ?from n_from ]  
    [ ?to n_to ]  
    [ ?k n_k ]  
    [ ?multiplier n_multiplier ]  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
    [ ?unit t_unit ]  
    [ ?ber g_ber ]  
    [ ?event S_event ]  
)  
=> value / nil
```

Description

Calculates cycle jitter from the result of a single event of pnoise analysis where *noisetype* is set as *pmjitter*, or from the result of pnoise analysis of oscillators where *noiseout* is set as [*pm*].

Arguments

<code>?from <i>n_from</i></code>	The lower frequency limiter.
<code>?to <i>n_to</i></code>	The upper frequency limiter.
<code>?k <i>n_k</i></code>	The number of cycles.
<code>?multiplier <i>n_multiplier</i></code>	The frequency multiplier. By default, it is set to 1.
<code>?result <i>t_result</i></code>	Name of the result of pnoise jitter analysis.
<code>?resultsDir <i>t_resultsDir</i></code>	Path to the results directory.
<code>?unit <i>t_unit</i></code>	Unit to measure jitters. Possible values are <i>Second</i> , <i>UI</i> , and <i>ppm</i> .
<code>?ber <i>g_ber</i></code>	The value of BER (Bit Error Rate) when the signal level is peak-to-peak.
<code>?event <i>S_event</i></code>	Index ID of an event.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Value Returned

<i>value</i>	Returns the value of cycle jitter.
<i>nil</i>	Returns <i>nil</i> , if the result name is incorrect and jitter cannot be calculated.

Examples

Returns the cycle jitter value.

```
drplRFJc( ?from 100 ?to 500000000 ?k 1 ?result "pnoise_pmjitter" ?unit "Second"
?event 0 )
```

drplRFJcc

```
drplRFJcc(  
    [ ?from n_from ]  
    [ ?to n_to ]  
    [ ?k n_k ]  
    [ ?multiplier n_multiplier ]  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
    [ ?unit t_unit ]  
    [ ?ber g_ber ]  
    [ ?event S_event ]  
)  
=> value / nil
```

Description

Calculates cycle-to-cycle jitter from the result of pnoise analysis for a single event. where *noisetype* is set as *pmjitter*, or from the result of pnoise analysis of oscillators where *noiseout* is set as *[pm]*.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Arguments

<code>?from <i>n_from</i></code>	The lower frequency limiter.
<code>?to <i>n_to</i></code>	The upper frequency limiter.
<code>?k <i>n_k</i></code>	The number of cycles.
<code>?multiplier <i>n_multiplier</i></code>	The frequency multiplier. By default, it is set to 1.
<code>?result <i>t_result</i></code>	Name of the result of pnoise jitter analysis.
<code>?resultsDir <i>t_resultsDir</i></code>	Path to the results directory.
<code>?unit <i>t_unit</i></code>	Unit to measure jitters. Possible values are <code>Second</code> , <code>UI</code> , and <code>ppm</code> .
<code>?ber <i>g_ber</i></code>	The value of BER (Bit Error Rate) when the signal level is peak-to-peak.
<code>?eventList <i>S_eventList</i></code>	Index ID of an event.

Value Returned

<code>value</code>	Returns the value of cycle jitter.
<code>nil</code>	Returns <code>nil</code> , if the result name is incorrect and jitter cannot be calculated.

Examples

Returns cycle-to-cycle jitter value.

```
drplRFJcc( ?from 100 ?to 50000000 ?k 1 ?result "pnoise_pmjitter" ?unit "Second"
?event 0)
```

drplParamSweepRFJc

```
drplParamSweepRFJc (
    [ ?from n_from ]
    [ ?to n_to ]
    [ ?k n_k ]
    [ ?multiplier n_multiplier ]
    [ ?result t_result ]
    [ ?resultsDir t_resultsDir ]
    [ ?unit t_unit ]
    [ ?ber g_ber ]
    [ ?eventList S_eventList ]
)
=> o_waveform / nil
```

Description

Calculates cycle jitter for parametric sweep with multiple events from the result of pnoise analysis where *noisetype* is set as *pmjitter*.

Arguments

<code>?from <i>n_from</i></code>	The lower frequency limiter.
<code>?to <i>n_to</i></code>	The upper frequency limiter.
<code>?k <i>n_k</i></code>	The number of cycles.
<code>?multiplier <i>n_multiplier</i></code>	The frequency multiplier. By default, it is set to 1.
<code>?result <i>t_result</i></code>	Name of the result of pnoise analysis.
<code>?resultsDir <i>t_resultsDir</i></code>	Path to the results directory.
<code>?unit <i>t_unit</i></code>	Unit to measure jitters. Possible values are <i>Second</i> , <i>UI</i> , and <i>ppm</i> .
<code>?ber <i>g_ber</i></code>	The value of BER (Bit Error Rate) when the signal level is peak-to-peak.
<code>?eventList <i>S_eventList</i></code>	List of index ID for each event of parameter sweep.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
<i>nil</i>	Returns <i>nil</i> , if the result name is incorrect and jitter cannot be calculated.

Examples

Returns a waveform that shows sweep variable on the X-axis and jitter value plotted on the Y-axis.

```
drplParamSweepRFJc (?from 1000000 ?to 1000000000 ?k 1 ?multiplier 1 ?result  
"pnoise_pmjitter" ?unit "Second" ?eventList '("-10.0 (1 1.25143e-10)" "-10.0  
(2 6.57854e-10)" "-9.0 (1 1.27759e-10)" "-9.0 (2 6.5234e-10)"))
```


drplParamSweepRFJcc

```
drplParamSweepRFJcc(  
    [ ?from n_from ]  
    [ ?to n_to ]  
    [ ?k n_k ]  
    [ ?multiplier n_multiplier ]  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
    [ ?unit t_unit ]  
    [ ?ber g_ber ]  
    [ ?eventList S_eventList ]  
)  
=> o_waveform / nil
```

Description

Calculates cycle-to-cycle jitter for parametric sweep with multiple events from the result of pnoise analysis where *noisetype* is set as *pmjitter*.

Arguments

<i>?from n_from</i>	The lower frequency limiter.
<i>?to n_to</i>	The upper frequency limiter.
<i>?k n_k</i>	The number of cycles.
<i>?multiplier n_multiplier</i>	The frequency multiplier. By default, it is set to 1.
<i>?result t_result</i>	Name of the result of pnoise analysis.
<i>?resultsDir t_resultsDir</i>	Path to the results directory.
<i>?unit t_unit</i>	Unit to measure jitters. Possible values are <i>Second</i> , <i>UI</i> , and <i>ppm</i> .
<i>?ber g_ber</i>	The value of BER (Bit Error Rate) when the signal level is peak-to-peak.
<i>?eventList S_eventList</i>	List of index ID for each event of parameter sweep.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
<i>nil</i>	Returns <i>nil</i> , if the result name is incorrect and jitter cannot be calculated.

Examples

Returns cycle-to-cycle jitter value and its corresponding waveform. This waveform shows the sweep variable on the X-axis and jitter value is plotted on the Y-axis.

```
drplParamSweepRFJc ?from 1000000 ?to 1000000000 ?k 1 ?multiplier 1 ?result  
"pnoise_pmjitter" ?unit "Second" ?eventList '("-10.0 (1 1.25143e-10)" "-10.0  
(2 6.57854e-10)" "-9.0 (1 1.27759e-10)" "-9.0 (2 6.5234e-10)")
```

drplRFValueAt

```
drplRFValueAt(  
    o_waveform  
    g_xValue  
)  
=> g_yValue / nil
```

Description

Returns the Y-axis value corresponding to the specified X-axis value on the given waveform.

Arguments

<i>o_waveform</i>	The waveform object.
<i>g_xValue</i>	The X-axis value at which you want to find the corresponding Y-axis value.

Value Returned

<i>g_yValue</i>	Returns the Y-axis value corresponding to the specified X-axis value.
<i>nil</i>	Returns <i>nil</i> if there is an error.

Examples

When the X-axis value is 200 KHz, the function above returns 37.5n, which is the Y-axis value of the Output Noise waveform from pnoise analysis.

```
drplRFValueAt(rfOutputNoise("v/sqrt(Hz)" ?result "pnoise") "200K")  
=> 37.5n
```

ddrplPssHarmThd

```
drplPssHarmThd(  
    o_waveform  
    x_harmonicNumber  
)  
=> x_THD / nil
```

Description

Calculates the total harmonic distortion (THD) in percentage for each waveform of frequency sweep. This function is available only for single-tone harmonic balance (hb) and single-tone periodic steady-state (pss) analyses.

THD is expressed as a percentage and calculated using the following formula:

$$\text{THD} = \frac{\sqrt{V_2^2 + V_3^2 + \dots + V_n^2}}{V_1} \times 100$$

where V_1 is the RMS value of the fundamental frequency voltage, and $V_2, V_3, \dots, V_n$ are the RMS values from second harmonic through last harmonic frequencies.

Arguments

<i>o_waveform</i>	Waveform or a family of waveforms. The <i>x</i> vector of the waveform must be frequency. For example, <code>vh('hb' / IFp" nil)</code> .
<i>x_harmonicNumber</i>	The harmonic number.

Value Returned

<i>x_THD</i>	THD value in percentage.
<i>nil</i>	THD value cannot be calculated because of an error.

Examples

Returns THD values for waveform `vh` at the specified harmonics 1 and 2. Note that `vh` represents the voltage across the net `/IFp` at all harmonics in the frequency domain of a single-tone harmonic balance analysis, `hb`.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

```
openResults("./simulation/RFworkshop/mixer_testbench/maestro/results/maestro/
ExplorerRun.0/psf/RFworkshop_mixer_testbench_1/psf")
; Opens simulation results of a single-tone hb analysis stored in the specified
directory.
results()
;Lists the results available in the currently open results directory.
harmonicList(?resultsDir "./simulation/RFworkshop/mixer_testbench/maestro/
results/maestro/ExplorerRun.0/psf/RFworkshop_mixer_testbench_1/psf" ?result
"hb_fi")
;Returns a list of available harmonics for the specified result of the results
directory.
selectResult('hb_fi)
;Selects the hb_fi result from the results directory.
outputs()
;Lists the outputs available in the selected result.
drplPssHarmThd(vh('hb "/IFp" nil) 1)
;Returns the THD value for waveform vh at the specified harmonic 1. Note that vh
represents the voltage across the net /IFp at all harmonics in the frequency domain
of a single-tone harmonic balance analysis, hb.
drplPssHarmThd(vh('hb "/IFp" nil) 2)
;Returns the THD value for waveform vh at the specified harmonic 2.
=>
"./simulation/RFworkshop/mixer_testbench/maestro/results/maestro/ExplorerRun.0/
psf/RFworkshop_mixer_testbench_1/psf"
=>
(hb_td hb_fd hb_fi model instance
  output designParamVals primitives subckts variables
)
=>
(0 1 2 3 4
  5 6 7 8
)
=> stdobj@0x33f31e00
=>
("/LOn" "/LOp" "/IFn" "/IFp" "/RF"
  "/VDDA1" "/VLO" "/VbiasLO" "/I19/net059" "I19.R2.R0.PLUSHEAD"
  "I19.R2.R0.MINUSHEAD" "I19.R1.R0.PLUSHEAD" "I19.R1.R0.MINUSHEAD" "/PORT0/PLUS"
"/PORT1/PLUS"
  "/PORT3/PLUS" "/V0/PLUS" "/V3/PLUS" "/largeTone_harm_index"
)
=> 53.35389
=>202.5374
```

drplSwpHp

```
drplSwpHp(  
    n_iIndex  
    n_jIndex  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
)  
=> o_waveform / nil
```

Description

Returns the hybrid matrix for sweep port SP analysis.

Arguments

<i>n_iIndex</i>	Index of the first port.
<i>n_jIndex</i>	Index of the second port.
?result <i>t_result</i>	Name of the result of SP analysis.
?resultsDir <i>t_resultsDir</i>	Path to the results directory.

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
nil	Returns nil, if the result name is incorrect and jitter cannot be calculated.

Examples

Returns a waveform with frequency on the X-axis and the hybrid matrix on the Y-axis.

```
drplSwpHp( 1 1 ?result "sp" )
```

drplSwpSp

```
drplSwpSp(  
    n_iIndex  
    n_jIndex  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
)  
=> o_waveform / nil
```

Description

Returns the S-parameter waveform for sweep port SP analysis.

Arguments

<i>n_iIndex</i>	Index of the first port.
<i>n_jIndex</i>	Index of the second port.
?result <i>t_result</i>	Name of the result of SP analysis.
?resultsDir <i>t_resultsDir</i>	Path to the results directory.

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
nil	Returns nil.

Examples

Returns a waveform with frequency on the X-axis and the S-parameter waveform on the Y-axis.

```
drplSwpSp( 1 1 ?result "sp" )
```

drplSwpYp

```
drplSwpYp(  
    n_iIndex  
    n_jIndex  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
)  
=> o_waveform / nil
```

Description

Returns the admittance matrix for sweep port SP analysis.

Arguments

<i>n_iIndex</i>	Index of the first port.
<i>n_jIndex</i>	Index of the second port.
?result <i>t_result</i>	Name of the result of SP analysis.
?resultsDir <i>t_resultsDir</i>	Path to the results directory.

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
nil	Returns nil.

Examples

Returns a waveform with frequency on the X-axis and the admittance matrix on the Y-axis.

```
drplSwpYp( 1 2 ?result "yp" )
```


drplSwpZm

```
drplSwpZm(  
    n_iIndex  
    n_jIndex  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
)  
=> o_waveform / nil
```

Description

Returns the port input impedance waveform for sweep port SP analysis.

Arguments

<i>n_iIndex</i>	Index of the first port.
<i>n_jIndex</i>	Index of the second port.
?result <i>t_result</i>	Name of the result of SP analysis.
?resultsDir <i>t_resultsDir</i>	Path to the results directory.

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
nil	Returns nil.

Examples

Returns a waveform with frequency on the X-axis and the port input impedance matrix on the Y-axis.

```
drplSwpZm( 2 2 ?result "sp")
```

drplSwpZp

```
drplSwpZp(  
    iIndex  
    jIndex  
    [ ?result t_result ]  
    [ ?resultsDir t_resultsDir ]  
)  
=> o_waveform / nil
```

Description

Returns the impedance matrix for sweep port SP analysis.

Arguments

<i>iIndex</i>	Index of the first port.
<i>jIndex</i>	Index of the second port.
<i>?result t_result</i>	Name of the result of SP analysis.
<i>?resultsDir t_resultsDir</i>	Path to the results directory.

Value Returned

<i>o_waveform</i>	Returns the waveform of cycle jitter.
<i>nil</i>	Returns <i>nil</i> .

Examples

Returns a waveform with frequency on the X-axis and the impedance matrix on the Y-axis.

```
drplSwpZp(2 2 ?result "sp")
```

drplSprobeHPM

```
drplSprobeHPM(  
    g_inst  
    x_index1  
    x_index2  
)  
=> o_waveform / nil
```

Description

Returns the waveform for h-parameter data, for the specified probe instance.

Arguments

<i>g_inst</i>	The probe instance along with the left or right port name. The format for specifying this argument is <i>sprobeInstance.port</i> , where <i>sprobeInstance</i> represents the probe instance name, while <i>port</i> specifies the port name.
<i>x_index1</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.
<i>x_index2</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.

Value Returned

<i>o_waveform</i>	Waveform object representing the h-parameters.
<i>nil</i>	Specified port indexes are invalid.

Examples

Specifies the results directory and selects the `sp_sprobe` analysis.

```
openResults("../psf")  
selectResults('sp_sprobe) sp sp
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Returns the waveform for h-parameters when the input and output port index are set to 1 and 2 respectively.

```
drplSprobeHPM("I8.left" 1 2)
```

drplSprobePortSPData

```
drplSprobePortSPData(  
    g_inst  
    x_index1  
    x_index2  
)  
=> o_waveform / nil
```

Description

Returns the waveform for the S-parameter data for the probe instance with port.

Arguments

<i>g_inst</i>	The probe instance with port, format is like <code>sprobeInstance.port</code> . The format for specifying this argument is <code>sprobeInstance.port</code> , where <code>sprobeInstance</code> represents the probe instance name, while <code>port</code> specifies the port name.
<i>x_index1</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.
<i>x_index2</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.

Value Returned

<i>o_waveform</i>	Waveform object representing the S-parameters.
<i>nil</i>	Unable to get the data from the waveform database.

Examples

Specifies the results directory and selects the `sp_sprobe` analysis.

```
openResults("../psf")  
selectResults('sp_sprobe')
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Returns the waveform for y-parameters when the input and output port index are set to 1 and 2 respectively.

```
drplSprobePortSPData("I8.left" 1 2)
```

Related Topics

[sp](#)

drplSprobePortVSWR

```
drplSprobePortVSWR(  
    g_inst  
    x_index1  
)  
=> o_waveform / nil
```

Description

Returns the waveform for the voltage standing wave ratio, for the specified probe instance.

Arguments

<i>g_inst</i>	The probe instance along with the left or right port name. The format for specifying this argument is <i>sprobeInstance.port</i> , where <i>sprobeInstance</i> represents the probe instance name, while <i>port</i> specifies the port name.
<i>x_index1</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.

Value Returned

<i>o_waveform</i>	Waveform object representing the voltage standing wave ratio.
<i>nil</i>	Unable to get the data from the waveform database.

Examples

Specifies the results directory and selects the *sp_sprobe* analysis.

```
openResults("../psf")  
selectResults('sp_sprobe)
```

Returns the waveform for the voltage standing wave ratio when the port index are set to 1.

```
drplSprobePortVSWR("I8.left" 1)
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Related topics

[VSWR](#)

drplSprobeYPM

```
drplSprobeYPM(  
    g_inst  
    x_index1  
    x_index2  
)  
=> o_waveform / nil
```

Description

Returns the waveform for y-parameter data, for the specified probe instance.

Arguments

<i>g_inst</i>	The probe instance along with the left or right port name. The format for specifying this argument is <i>sprobeInstance.port</i> , where <i>sprobeInstance</i> represents the probe instance name, while <i>port</i> specifies the port name.
<i>x_index1</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.
<i>x_index2</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.

Value Returned

<i>o_waveform</i>	Waveform object representing the y-parameters.
<i>nil</i>	Specified port indexes are invalid.

Examples

Specifies the results directory and selects the `sp_sprobe` analysis.

```
openResults("../psf")  
selectResults('sp_sprobe')
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Returns the waveform for y-parameters when the input and output port index are set to 1 and 2 respectively.

```
drplSprobeYPM("I8.left" 1 2)
```

Related Topics

[yym](#)

drplSprobeZM

```
drplSprobeZM(  
    g_inst  
    x_index1  
)  
=> o_waveform / nil
```

Description

Returns the waveform for the port input impedance, for the specified probe instance.

Arguments

<i>g_inst</i>	The probe instance along with the left or right port name. The format for specifying this argument is <i>sprobeInstance.port</i> , where <i>sprobeInstance</i> represents the probe instance name, while <i>port</i> specifies the port name.
<i>x_index1</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.

Value Returned

<i>o_waveform</i>	Waveform objecting representing the port input impedance.
<i>nil</i>	Unable to get the data from the waveform database.

Examples

Specifies the results directory and selects the `sp_sprobe` analysis.

```
openResults("../psf")  
selectResults('sp_sprobe)
```

Returns the waveform for the port input impedance when the port index are set to 1.

```
drplSprobeZM("I8.left" 1)
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Related Topics

zm

drplSprobeZPM

```
drplSprobeZPM(  
    g_inst  
    x_index1  
    x_index2  
)  
=> o_waveform / nil
```

Description

Returns the waveform for z-parameter data, for the specified probe instance.

Arguments

<i>g_inst</i>	The probe instance along with the left or right port name. The format for specifying this argument is <i>sprobeInstance.port</i> , where <i>sprobeInstance</i> represents the probe instance name, while <i>port</i> specifies the port name.
<i>x_index1</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.
<i>x_index2</i>	Port index for sp simulation. By default, the port index is not specified. Valid values: Available port index, such as 1, 2.

Value Returned

<i>o_waveform</i>	Waveform object representing the z-parameters.
<i>nil</i>	Specified port indexes are invalid.

Examples

Specifies the results directory and selects the `sp_sprobe` analysis.

```
openResults("../psf")  
selectResults('sp_sprobe')
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Returns the waveform for y-parameters when the input and output port index are set to 1 and 2 respectively.

```
drplSprobeZPM("I8.left" 1 2)
```

Related Topics

[zpm](#)

drplGetMeas

```
drplGetMeas(  
    t_measurement  
)  
=> o_waveform
```

Description

Returns the value of the specified measurement for simulation results of wireless envelope analysis. These measurements can be one of the following: error vector magnitude (EVM), bit error rate (BER), or frequency shift keying error (FSKerr). The x- and y-axis vectors of the waveform are frame index and frame values, respectively

Arguments

<i>t_measurement</i>	EVM, BER, or FSKerr for a wireless probe (wprobe).

Value Returned

<i>o_waveform</i>	A waveform object that represents the values of the specified measurement.

Examples

The following examples shows how to plot the waveform objects that represent the bit error rate and error vector magnitude of wireless probes, WPRB1 and WPRB2 from the simulation results of the wireless envelope analysis.

```
openResults("/home/user/envlpsim/WirelessLib/  
wireless_LTE_W_Modulator_and_DAC_12_bit/maestro/results/maestro/ExplorerRun.0/  
psf/Wireless/psf")  
;Opens the simulation results of wireless envelope analysis stored in the specified  
directory.  
results()  
;Lists the results available in the currently open results directory.  
selectResult('envlp_fd')  
;Selects the results result 'envlp_fd'.  
w1=drplGetMeas("WPRB1.ber")  
;Creates a waveform object that represents the bit error rate of the wireless probe  
WPRB1.  
w2=drplGetMeas("WPRB1.evm")
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

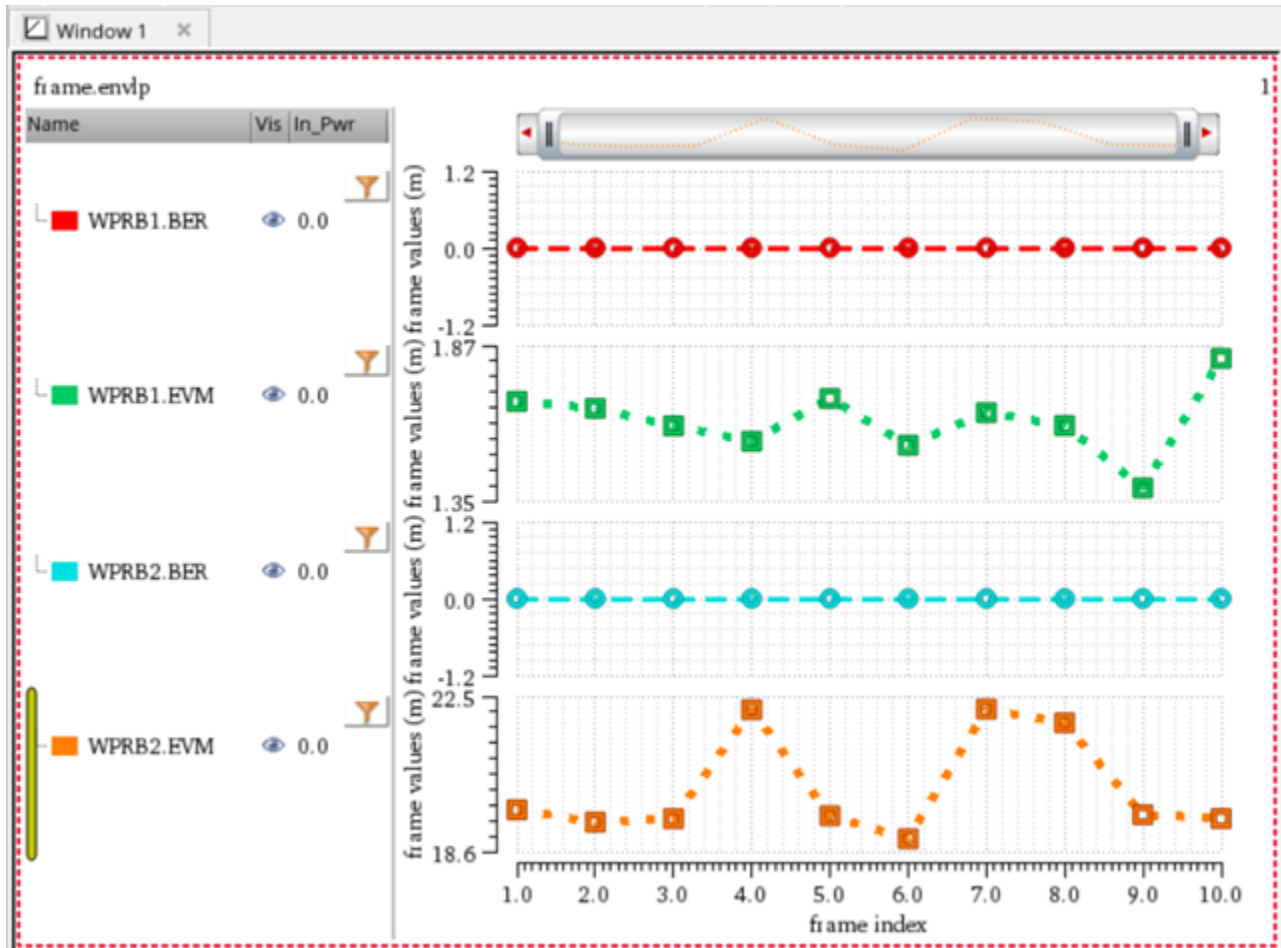
```
;Creates a waveform object that represents the error vector magnitude of the
wireless probe WPRB1.
w3=drplGetMeas("WPRB2.ber")
;Creates a waveform object that represents the bit error rate of the wireless probe
WPRB2.
w4=drplGetMeas("WPRB2.evm")
;Creates a waveform object that represents the bit error vector magnitude of the
wireless probe WPRB2.
awvCreatePlotWindow()
;Creates a Waveform window.
awvPlotWaveform(
    awvGetCurrentWindow()
    list(w1 w2 w3 w4)
    ?expr list("WPRB1.BER" "WPRB1.EVM" "WPRB2.BER" "WPRB2.EVM")
    ?color list("y1" "y66" "y12" "y6")
    ?index list(1 2 3 4)
    ?lineType list("line" "line" "line" "line")
    ?lineStyle list("dash" "dot" "dash" "dot")
    ?lineThickness list("thick" "extraThick" "thick" "extraThick")
    ?showSymbols list(t t t t)
    ?dataSymbol list(4 5 4 5)
    ?stripNumber list(1 2 3 4)
)
;Plots waveforms "WPRB1.BER", "WPRB1.EVM", "WPRB2.BER", and "WPRB2.EVM" in the
strips 1, 2, 3, and 4 of the current Waveform window.

=> "/home/user/envlpsim/WirelessLib/wireless_LTE_W_Modulator_and_DAC_12_bit/
maestro/results/maestro/ExplorerRun.0/psf/Wireless/psf"
=>
(envlp_fd conste binary fwrls ccdf
    avgwrls param model instance output
    designParamVals primitives subckts variables
)
=> stdobj@0x37d23068
=> srrWave:0x3b45d020
=> srrWave:0x3b45d030
=> srrWave:0x3b45d040
=> srrWave:0x3b45d050
=> window:3
```


Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

=> t



drplGetSpectrum

```
drplGetSpectrum(  
    t_wirelessProbe  
)  
=> o_waveform / nil
```

Description

Creates a waveform object that represents the spectrum of the signal to which the specified wireless probe is connected. The x- and y-axis vectors of the waveform are `frequency` and `dBv/Hz`, respectively.

You must have run a wireless envelope simulation before using this function.

Arguments

<code>t_wirelessProbe</code>	Name of the wireless probe which is connected the signal whose spectrum you want to plot.

Value Returned

<code>o_waveform</code>	A waveform object, representing the spectrum of the signal to which the specified wireless probe is connected.
<code>nil</code>	Cannot create the waveform object because of an error.

Examples

The following example shows how to plot spectrum of the signals to which the wireless probes, WPRB1 and WPRB2 are connected.

```
openResults("/home/user/envlpsim/WirelessLib/  
wireless_LTE_W_Modulator_and_DAC_12_bit/maestro/results/maestro/ExplorerRun.0/  
psf/Wireless7psf")  
;Opens the simulation results of wireless envelope analysis stored in the specified  
directory.  
results()  
;Lists the results available in the currently open results directory.  
selectResult('envlp_fd')  
;Selects the results result 'envlp_fd'.  
w1=drplGetSpectrum("WPRB1")
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

;Creates a waveform object that represents the spectrum of the net (signal) to which the wireless probe WPRB1 is connected.

```
w2=drplGetSpectrum("WPRB2")
```

;Creates a waveform object that represents the spectrum of the net (signal) to which the wireless probe WPRB2 is connected.

```
awvCreatePlotWindow()
```

;Creates a Waveform window.

```
awvPlotWaveform(  
    awvGetCurrentWindow()  
    list(w1 w2)  
    ?expr list("WPRB1_spectrum" "WPRB2_spectrum")  
    ?color list("y1" "y66")  
    ?index list(1 2)  
    ?lineType list("line" "line")  
    ?lineStyle list("solid" "solid")  
    ?lineThickness list("thick" "thick")  
    ?showSymbols list(t t)  
    ?dataSymbol list(4 5)  
    ?stripNumber list(1 2)  
)
```

;Plots waveforms "WPRB1_spectrum" and "WPRB1_spectrum" in the strip number 1 and 2 of the current Waveform window.

```
=>"/home/user/envlpsim/WirelessLib/wireless_LTE_W_Modulator_and_DAC_12_bit/  
maestro/results/maestro/ExplorerRun.0/psf/Wireless/psf"
```

```
=>
```

```
(envlp_fd conste binary fwrls ccdf  
    avgwrls param model instance output  
    designParamVals primitives subckts variables  
)
```

```
=> stdobj@0x375d9068
```

```
=> srrWave:0x3ad23ac0
```

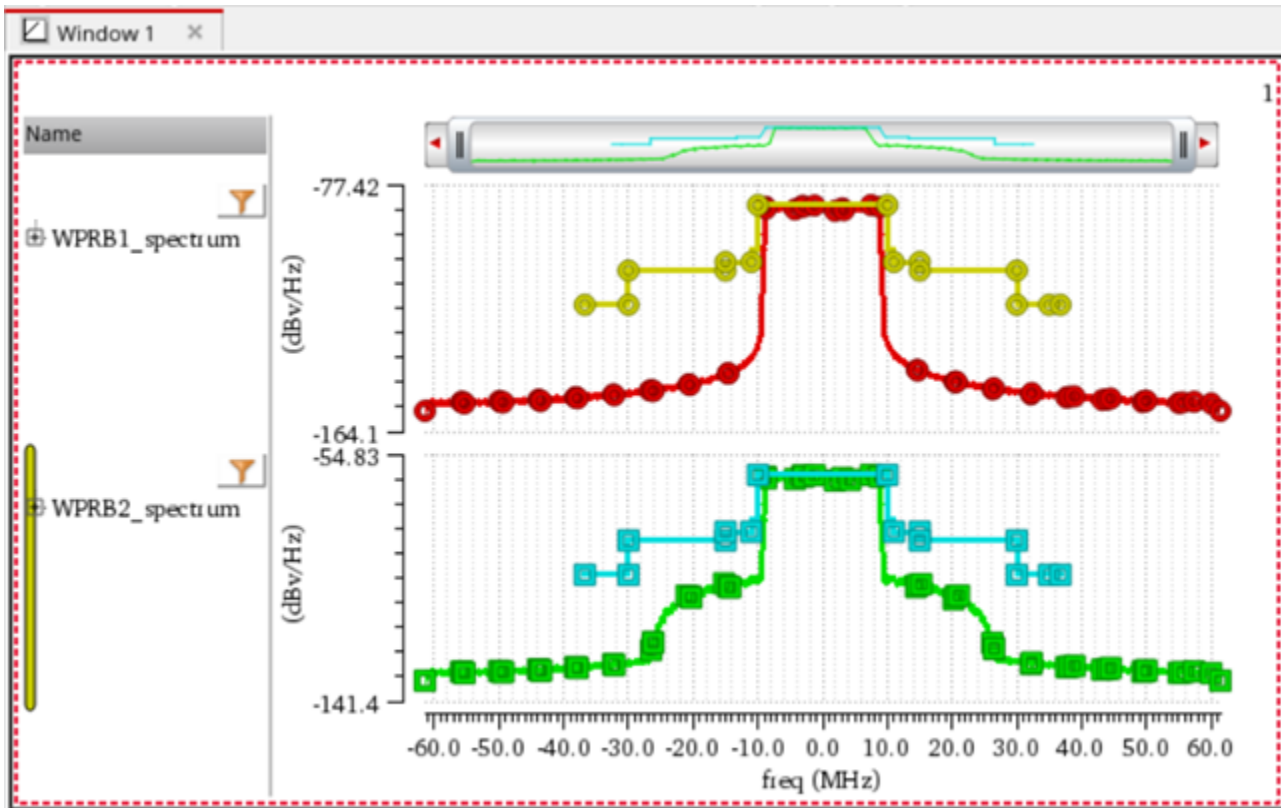
```
=> srrWave:0x3ad22590
```

```
=> window:3
```

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

=> t



OCEAN Script Functions

This topic lists the SKILL functions that let you open, edit, and save an OCEAN script.

[asiOpenOceanScript](#)

[asiWriteOceanScript](#)

[asiCloseOceanScript](#)

asiOpenOceanScript

```
asiOpenOceanScript(  
    t_filename  
)  
=> g_fptr
```

Description

Opens the specified file for writing the OCEAN script. If the file exists, it asks for your permission to overwrite it.

Arguments

<i>t_filename</i>	Name of the file or directory to be opened.

Value Returned

<i>g_fptr</i>	Returns the file pointer.

Examples

Opens the file oceanScript.

```
asiOpenOceanScript("/usr/mnt3/user/simulation/ckt/spectre/schematic/netlist/  
oceanScript")
```

asiWriteOceanScript

```
asiWriteOceanScript(  
    p_filePointer  
    o_session  
    [ ?noRun g_noRun ]  
    [ ?fullKey g_fullKey ]  
    [ ?calledFromCorners g_calledFromCorners ]  
)  
=> t / nil
```

Description

Writes the OCEAN script.

Arguments

<i>p_filePointer</i>	Name of the file pointer, which was opened by calling <code>asiOpenOceanScript</code> .
<i>o_session</i>	Current session.
<i>?noRun g_noRun</i>	Flag to determine if the <code>run()</code> command and any post-processing plots should be written to the OCEAN script. For example, this flag may be used with a tool that has its own <code>run()</code> command.
<i>?fullKey g_fullKey</i>	Flag to determine if the default values should be written to the OCEAN script.
<i>?calledFromCorners g_calledFromCorners</i>	Flag to determine whether or not the model file information is to be written to the OCEAN script generated from the Corners tool. By default, it is set to <code>nil</code> , so no model file information is written to the OCEAN script generated from the Corners tool.

Value Returned

<i>t</i>	Returns <i>t</i> if the OCEAN script was written successfully.
<i>nil</i>	Returns <i>nil</i> if there was an error.

Virtuoso ADE SKILL Reference

Functions to Access Simulation Results

Examples

Writes the current session's commands to the OCEAN script if the .cdsenv variable `saveDefaultsFlag` is nil. Otherwise, writes all commands (used as well as unused) to the OCEAN script.

```
saveDefaultsFlag=envGetVal("asimenv.misc" "saveDefaultsToOCEAN" )
asiWriteOceanScript( fp session ?fullKey saveDefaultsFlag )
```

Writes the current session's commands to the OCEAN script.

```
asiWriteOceanScript( fp session)
```

Writes the OCEAN script without adding `run()` and plots to the end.

```
asiWriteOceanScript( fp session ?noRun t)
```


asiCloseOceanScript

```
asiCloseOceanScript(  
    p_filePointer  
)  
=> t / nil
```

Description

Closes the OCEAN script.

Arguments

<i>p_filePointer</i>	File pointer to the OCEAN script file.

Value Returned

t	Successful operation
nil	Indicates an error.

Examples

Closes the specified file.

```
asiCloseOceanScript(fpPtr)
```

Simulation Environment Functions

The simulation environment functions are used by the simulation environment to build and maintain the information and interfaces used by the environment. Most of these functions require the `session` argument. You can obtain this argument by using the `sevSession()` function or by calling these functions from a `.menus` file.

sevGetSessionType

```
sevGetSessionType(  
    t_session  
)  
=> t_toolName / nil
```

Description

Returns the name of the tool associated with the given session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t_toolName</i>	Name of the tool in which the given session is open. Possible values: adel, adexl, explorer, and assembler.
<i>nil</i>	Indicates an error.

Examples

The following example returns the name of the tool in which the session `sevsess` is open.

```
sess = (axlGetWindowSession (hiGetCurrentWindow))  
sevsess = (axlGetToolSession sess "AC1")  
sevGetSessionType(sevsess)  
=> assembler
```

sevSetMainWindowPulldownMenus

```
sevSetMainWindowPulldownMenus (  
    l_menus  
)  
=> t / nil
```

Description

Sets the menus for the Virtuoso Analog Simulation Environment window.

Arguments

<i>l_menus</i>	The list of menu names

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetMTSMODE

```
sevSetMTSMODE (
    o_session
    g_mtsMode
    [ g_noPopups ]
)
=> t / nil
```

Description

Enables or disables MTS mode.

Arguments

<i>o_session</i>	The simulation environment session object.
<i>g_mtsMode</i>	Enables or disables MTS mode.
<i>g_noPopups</i>	Controls whether the function pops up a warning message or not when the config view is in edit mode.

Value Returned

<i>t</i>	MTS mode is set successfully.
<i>nil</i>	MTS mode cannot be set because of an error.

Example

The following example sets MTS mode.

```
sess = (axlGetWindowSession (hiGetCurrentWindow))
sdb = (axlGetMainSetupDB sess)
(foreach testname (cadr (axlGetTests sdb) )
  (printf "Test %s\n" testname)
  sevsess = (axlGetToolSession sess testname)
  mtsmode = (sevMTSMODE sevsess)
  (printf "\tMTS mode = %L\n" mtsmode)
  ;Example setting MTS mode t
  (unless mtsmode
    (sevSetMTSMODE sevsess t)
  )
)
```

sevMTSMode

```
sevMTSMode (  
    o_session  
)  
=> t / nil
```

Description

Returns the status of MTS mode.

Arguments

<i>o_session</i>	The simulation environment session object.

Value Returned

<i>t</i>	MTS mode is enabled.
<i>nil</i>	MTS mode is disabled.

sevMTSOptions

```
sevMTSOptions(  
    o_session  
)  
=> t / nil
```

Description

Invokes the MTS Options form if MTS mode is enabled. A dialog box appears if the MTS option is disabled.

Arguments

<i>o_session</i>	The simulation environment session object.

Value Returned

<i>t</i>	Displays the MTS Options form.
<i>nil</i>	Does not display the MTS Options form.

sevOpenXterm

```
sevOpenXterm(  
    t_dirPath  
)  
=> t / nil
```

Description

Opens an terminal window at the given path.

Arguments

<i>t_dirPath</i>	The absolute path to which the terminal window should open.

Value Returned

<i>t</i>	Terminal window opens.
<i>nil</i>	The path is not accessible.

Example

The following example opens the terminal window in the `/home/user/debugPath` directory.

```
dirPath = "/home/user/debugPath"  
(sevOpenXterm dirPath)  
=> t
```


sevSetSchematicPulldownMenus

```
sevSetSchematicPulldownMenus(  
    l_menus  
)  
=> t / nil
```

Description

Sets the simulation menus on the schematic window.

Arguments

<i>l_menus</i>	The list of menu entries

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetTypeInWindowPulldownMenus

```
sevSetTypeInWindowPulldownMenus (  
    l_menus  
)  
=> t / nil
```

Description

Sets the menu on the simulator type in window.

Arguments

<i>l_menus</i>	The list of menu entries

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetMenuItemLists

```
sevSetMenuItemLists(  
    l_lists  
)  
=> t / nil
```

Description

Creates the menu item lists.

Arguments

<i>l_lists</i>	The list of menu entries

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevAddMenuItemLists

```
sevAddMenuItemLists(  
    l_lists  
)  
=> t / nil
```

Description

Adds menu items to an existing menu item list.

Arguments

<i>l_lists</i>	The list of menu entries

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevDirectPlotMenu

```
sevDirectPlotMenu(  
    t_session  
    l_items  
)  
=> t / nil
```

Description

Creates the direct plot menu item list.

Arguments

<i>t_session</i>	The simulation environment session.
<i>l_items</i>	The list of menu entries.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevEnvironment

```
sevEnvironment(  
    t_session  
)  
=> o_session / nil
```

Description

Displays the OASIS session object tied to the simulation environment session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>o_session</i>	Returns the OASIS session object
<i>nil</i>	there is no OASIS session object currently tied to the simulation environment.

sevNoEnvironment

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

Description

Indicates whether an OASIS environment is tied to the simulation environment.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSaveState

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

Description

Displays the *Saving State* form that lets you save the current state of the simulation environment. You can use this function for only an ADE L or ADE XL session. You cannot save the state of a maestro session.

Arguments

<code>t_session</code>	The simulation environment session.

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

sevLoadState

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

Description

Displays the *Loading State* form that lets you load a saved state into the current simulation environment. In a maestro cellview, you can use this functions to load an adexl state.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

Related Topics

[Migrating ADE L or ADE XL Cellviews to ADE Assembler](#)

sevSaveOceanScript

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

Description

Displays the Save Ocean Script to File form that lets you save an OCEAN script that will regenerate the current session to the specified file.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditOptions

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

Description

Displays the Editing Session Options form that lets you edit the options for the given simulation environment session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevOpenSchematic

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

Description

Displays a schematic window for the design that is tied to the simulation environment.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevMenuItems

```
sevMenuItems(  
    t_session  
    t_name  
)  
=> t / nil
```

Description

Displays the menu item list that corresponds to the named menu item.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_name</i>	The name of menu item from the menu item list.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevReset

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

Description

Resets the simulation environment session to its default values.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevQuit

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

Description

Quits the simulation session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevCreateMainWindow

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

Description

Creates the main simulation environment window. If a window is already open for the specified session, the function brings it to the top.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevChooseSimulator

```
sevChooseSimulator(  
    t_session  
    [ g_disableProjectDir ]  
)  
=> t / nil
```

Description

Displays the Choosing Simulator/Directory/Host form that lets you choose the simulator you want to use, the run directory, and the host machine.

Arguments

<i>t_session</i>	The simulation environment session.
<i>g_disableProjectDir</i>	Optional argument that disables the <i>Project Directory</i> field in the Choosing Simulator/Directory/Host form. Valid Values: <i>t</i> , disables the <i>Project Directory</i> field; <i>nil</i> , enables the <i>Project Directory</i> field. Default value: <i>nil</i>

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevChooseTemperature

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

Description

Displays the Setting Temperature form that lets you set the simulation temperature.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevMpuTool

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

Description

Displays the Setting Model Path form that lets you select the paths to the model files.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevChooseEnvironmentOptions

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

Description

Displays the Environment Options form that lets you select the environment options for the simulation environment session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditStimulus

```
sevEditStimulus(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Displays the specified stimulus in a window for editing.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of stimulus - analog or digital.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevNonMixedSignal

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

Description

Indicates whether the session is running a mixed signal simulation.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The session is running a mixed-signal simulation.
<i>nil</i>	The session is running an analog simulation.

sevEditSimulationFile

```
sevEditSimulationFile(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Displays the specified file in a window for editing.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of simulation file to edit. Possible values are <code>include</code> , <code>model</code> , and <code>stimulus</code> .

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevChooseDesign

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

Description

Displays the Choosing Design form that lets you select the design to simulate.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditSelectedAnas

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

Description

Displays the Choosing Analyses form that lets you select and edit the analyses.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditSelectedVars

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

Description

Displays the Editing Design Variables form that lets you edit the simulation variables and their values.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditSelectedOuts

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

Description

Displays the Outputs assistant that lets you edit the simulation outputs.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevChangeOutsOnSchematic

```
sevChangeOutsOnSchematic(  
    t_session  
    t_setType  
    [ ?selectionMode t_selectionMode ]  
)  
=> t / nil
```

Description

Sets up for selection of selected output types from the schematic.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_setType</i>	The type of outputs to change: save, march, or plot.
<i>?selectionMode t_selectionMode</i>	The selection mode for selecting outputs.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSaveOptions

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

Description

Displays the Save Options form that lets you select what voltages and currents should be automatically saved in the simulation.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevDeleteSelectedAnas

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

Description

Deletes the analyses that are currently selected in the *Analysis* listbox.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoAnaSelections

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

Description

Indicates whether any analyses are selected in the *Analysis* listbox.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	No analyses are selected in the analysis listbox.
<i>nil</i>	There are analyses selected in the analysis listbox.

sevActivateSelectedAnas

```
sevActivateSelectedAnas(  
    t_session  
    g_active  
)  
=> t / nil
```

Description

Enables or disables the selected analyses.

Arguments

<i>t_session</i>	The simulation environment session.
<i>g_active</i>	Boolean (<i>t / nil</i>) to either activate or deactivate the selected analyses.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevDeleteSelectedVars

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

Description

Deletes any variables that are selected in the Variables listbox.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoVarSelections

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

Description

Indicates whether any variables are selected in the Variables listbox.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	No variables are selected in the <i>Variables</i> listbox.
<i>nil</i>	Variables are selected in the <i>Variables</i> listbox.

sevFindSelectedVars

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

Description

Highlights the device on the schematic where the selected variable is used.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevCopyCellViewVariables

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

Description

Copies the cell view variables and their values into the simulation environment.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevCopyVariablesToCellView

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

Description

Copies the simulation session variables and their values into the cellview.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevDeleteSelectedOuts

```
sevDeleteSelectedOuts(  
    t_session  
    [ l_listbox ]  
)  
=> t / nil
```

Description

Deletes the selected items from the outputs list box.

Arguments

<i>t_session</i>	The simulation environment session.
<i>l_listbox</i>	List of items to be deleted from the outputs list box.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevExportOutputsToTxt

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

Description

Exports the outputs specified in the Output listbox to a text file. This text file can be edited and imported back to ADE.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevImportOutputsFromTxt

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

Description

Imports the outputs saved in a text file to the output list box in the main ADE window.

Ensure that you retain the format of the file as it was exported from *ADE Output* list box using `sevExportOutputsToTxt()`.

Arguments

<code>t_session</code>	The simulation environment session.

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

sevExportOutputsToCSV

```
sevExportOutputsToCSV(  
    t_sevSession  
)  
=> t / nil
```

Description

Exports the outputs specified in the ADE setup to the CSV file.

Arguments

<i>t_sevSession</i>	The simulation environment session. For more information, refer to sevSession .

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevExportOutputsToFile

```
sevExportOutputsToFile(  
    t_sevSession  
)  
=> t / nil
```

Description

Exports the output from the ADE setup to the text or CSV file, which is specified by the `.cdsenv` variable `outputsImportExportVersion`. If the value of `outputsImportExportVersion` variable is greater than 1.0, then the function generates the output to the CSV file, else it generates the output to the text file.

You can set the value of the `.cdsenv` variable `outputsImportExportVersion` using the command: `asimenv.misc outputsImportExportVersion float value`.

Arguments

<code>t_sevSession</code>	The simulation environment session. For more information, refer to sevSession .

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

Example

```
envSetVal("asimenv.misc" "outputsImportExportVersion" 'float 1.1)  
sevExportOutputsToFile(sevSession)  
=> t
```

sevImportOutputsFromCSV

```
sevImportOutputsFromCSV(  
    t_sevSession  
)  
=> t / nil
```

Description

Imports the outputs from the CSV file to the ADE setup.

Arguments

<i>t_sevSession</i>	The simulation environment session. For more information, refer to sevSession .

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

Example

The following example generates the CSV file with the outputs specified in the ADE L session *sevSession*.

```
sevImportOutputsFromCSV(sevSession)  
=> t
```

sevImportOutputsFromFile

```
sevImportOutputsFromFile(  
    t_sevSession  
)  
=> t / nil
```

Description

Imports the output to the ADE setup from the text or CSV file, which is specified by the `.cdsenv` variable `outputsImportExportVersion`. If the value of `outputsImportExportVersion` variable is greater than 1.0, then the function imports the outputs from the CSV file, else it imports the outputs from the text file.

You can set the value of the `.cdsenv` variable `outputsImportExportVersion` using the command: `asimenv.misc outputsImportExportVersion float value`.

Arguments

<code>t_sevSession</code>	The simulation environment session. For more information, refer to sevSession .

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

Example

The following example imports outputs from the CSV file in the ADE L session `sevSession`.

```
envSetVal("asimenv.misc" "outputsImportExportVersion" 'float 1.1)  
sevImportOutputsFromFile(sevSession)  
=> t
```

sevNoOutSelections

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

Description

Indicates whether any output is selected in the Outputs listbox.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	No outputs are selected.
<i>nil</i>	An output is selected.

sevRemovePlotWindow

```
sevRemovePlotWindow(  
    t_session  
    w_windowID  
)  
=> l_plotWindows / nil
```

Description

Removes the specified plot window from the list of plot windows owned by the given simulation environment session.

Arguments

<i>t_session</i>	The simulation environment session.
<i>w_windowID</i>	Window ID of a plot window owned by the given session.

Value Returned

<i>l_plotWindows</i>	Returns a list of remaining plot window IDs owned by the given session.
<i>nil</i>	Indicates an error.

sevSetPropertyForSelectedOuts

```
sevSetPropertyForSelectedOuts (  
    t_session  
    t_property  
    g_value  
)  
=> t / nil
```

Description

Sets the property to the specified value on the selected outputs.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_property</i>	The property name to set on the selected outputs.
<i>g_value</i>	The value to set the property to.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSimulator

```
sevSimulator(  
    t_session  
)  
=> t_simulatorName
```

Description

Displays the name of the simulator used in the session as a string.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t_simulatorName</i>	Returns the name of the simulator in the session.

sevRunEngine

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

Description

Runs a simulation from the simulation environment session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevStopEngine

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

Description

Stops the currently running simulation that is tied to the session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevIsContinuable

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

Description

Indicates whether the currently run simulation can be continued from its stopping point.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetEngineOptions

```
sevSetEngineOptions(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Displays the Engine Options form for the selected type for editing.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of engine options to edit.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNetlistFile

```
sevNetlistFile(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Creates the selected type of netlist file.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of netlist file to create. For socket netlisting, you may choose between <code>createRaw</code> , <code>displayRaw</code> , <code>createFinal</code> , or <code>displayFinal</code> . For direct netlisting, the options are <code>create</code> , <code>display</code> , and <code>recreate</code> .

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevOpenEncap

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

Description

Opens the command type in window that lets you enter commands directly to the simulator.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevViewSimulatorOutput

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

Description

Displays the simulation output log.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoOutputLog

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

Description

Indicates whether the simulation output log exists.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	No simulation output log exists.
<i>nil</i>	The simulation output log does exist.

sevConvergence

```
sevConvergence(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Displays the form for setting up the selected type of convergence aid.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of convergence aid to set up. The value can be any of these: 'storeRestore, 'transientStoreRestore, 'nodeSet, 'initialCondition, or 'forceNode.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevNoResults

```
sevNoResults(  
    t_session  
    [ t_type ]  
)  
=> t / nil
```

Description

Indicates whether the specified results exist.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of results to check for. The type of results to check existence of. The values can be any of these: 'dc', 'dc_op', 'dc_sens', 'dc_op_sens', 'ac', 'ac_sens', 'noise', 'tran', 'tran_op', 'tran_ic', 'tran_sens', 'xf', 'sparam', 'spss.td.pss', 'spss.fd.pss', 'td.pss', 'td.envlp', 'tran.pss', 'fd.pss', 'fd.envlp', 'fd.pdisto', 'fi.pdisto', 'tdr', 'hb', 'four', 'disto', 'model', 'instance', 'output', 'design_variables', 'summary', 'hb_noise', 'osc'.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoPlottableOutputs

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

Description

Indicates whether there are any plottable outputs.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	there are no plottable expressions.
<i>nil</i>	there are plottable expressions.

sevCircuitCond

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

Description

Displays the Circuit Conditions form that lets you set or display any special circuit conditions.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoDesign

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

Description

Indicates whether a design is tied to the simulation environment.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetSimDataDir

```
sevSetSimDataDir(  
    t_session  
    t_dir  
)  
=> t / nil
```

Description

Loads the results for the specified simulation environment session and results directory.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_dir</i>	The complete path of the results directory where the data is stored. The path should include the name of the results directory.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

Example

Loads the results from the results directory, *myResults*, at the path *~/simulation/ampTest/spectre* for the *sevSession1* session.

```
sevSetSimDataDir( 'sevSession1 "~/simulation/ampTest/spectre/myResults" )  
=> t
```

sevSaveResults

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

Description

Displays the *Save Results* form that lets you save the current results.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSelectResults

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

Description

Displays the Select Results form that lets you select and load a previously saved set of results.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevDeleteResults

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

Description

Displays the Delete Results form that lets you delete a set of previously saved results.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditPlottingOptions

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

Description

Displays the Setting Plotting Options form that lets you edit the plotting and printing options.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevPlotAllOutputs

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

Description

Plots all enabled plottable outputs.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoPlottableSignals

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

Description

Indicates whether there are any plottable outputs.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevPlotSignals

```
sevPlotSignals(  
    t_session  
    t_type  
    [ ?disableRedraw g_disableRedraw ]  
)  
=> t / nil
```

Description

Plots the specified type of signals that exist as outputs.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of signals to plot. The value can be any of these: 'tran, 'ac, 'dc, or 'noise.
<i>?disableRedraw g_disableRedraw</i>	Flag to disable redraw of plots.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEvaluateAndPlotExpressions

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

Description

Evaluates and plots all the output expressions.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoPlottableExpressions

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

Description

Indicates whether there are any plottable expressions.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevPrintResults

```
sevPrintResults(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Displays the Print Results form that lets you print the selected type of simulation results.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of results to print. Possible values are: 'dcOpPoints, 'tranOpPoints, 'modelParameters, 'sensitivities, 'noiseParameters, 'noiseSummary, 'dcNodeVoltages, or 'tranNodeVoltages.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevRetrieveFromEffectiveCDF

```
sevRetrieveFromEffectiveCDF(  
    t_sevSession  
)  
=> t / nil
```

Description

Retrieves data from the effective CDF. This function should be used before the `sevAnnotateResults` function.

Arguments

<i>t_sevSession</i>	The simulation environment session. For more information, see sevSession .

Value Returned

<i>t</i>	The function call is successful.
<i>nil</i>	Indicates an error.

Example

The following example calls the `sevRetrieveFromEffectiveCDF` function to retrieve the data from the effective CDF and then calls `sevAnnotateResults` function to annotate the selected results.

```
cellId = ddGetObj( "analogLib" "res" )  
opPointLabels="v i"  
cdfId=cdfCreateUserCellCDF( cellId )  
cdfId->opPointLabelSet = opPointLabels  
sevRetrieveFromEffectiveCDF(sevSession(hiGetCurrentWindow()))  
sevAnnotateResults( sevSession(hiGetCurrentWindow()) 'dcOpPoints)
```

sevAnnotateResults

```
sevAnnotateResults(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Annotates the selected results to the schematic window.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of results to annotate to the schematic. The possible values are: 'componentParameters, 'modelParameters, 'dcOpPoints, 'tranOpPoints, 'netNames, 'dcNodeVoltages, 'tranNodeVoltages, 'defaults, 'dcTermCurrents, 'tranTermCurrents, 'pinNames, or 'voltageLevels.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

Example

The following example calls the `sevRetrieveFromEffectiveCDF` function to retrieve the data from the effective CDF and then calls `sevAnnotateResults` function to annotate the selected results.

```
cellId = ddGetObj( "analogLib" "res" )  
opPointLabels="v i"  
cdfId=cdfCreateUserCellCDF( cellId )  
cdfId->opPointLabelSet = opPointLabels  
sevRetrieveFromEffectiveCDF(sevSession(hiGetCurrentWindow()))  
sevAnnotateResults( sevSession(hiGetCurrentWindow()) 'dcOpPoints)
```

sevRegisterPcellsForAnnotation

```
sevRegisterPcellsForAnnotation(  
    l_LibCellNames  
)  
=> l_LibCellNames/ nil
```

Description

Returns a list of library and cell name pairs for the registered Pcells.

Arguments

<i>l_LibCellNames</i>	The list of library and cell name pairs for the Pcells to be registered.

Value Returned

<i>l_LibCellNames</i>	The list of library and cell name pairs for the Pcells, if registered successfully.
<i>nil</i>	The call is unsuccessful.

Example

The following example shows how you can use this function to get a list of registered Pcells:

```
sevRegisterPcellsForAnnotation(list("tsmcN5/analog_cmos_cell" "tsmcN5/  
hp_analog_cell" "tsmcN5_ArrayLib/mosfet_StackGate"))  
  
=> ("tsmcN5/analog_cmos_cell" "tsmcN5/hp_analog_cell" "tsmcN5_ArrayLib/  
mosfet_StackGate")
```

sevGetRegisteredPcellsForAnnotation

```
sevGetRegisteredPcellsForAnnotation(  
    )  
=> l_LibCellNames/ nil
```

Description

Returns the list of library and cell name pairs of the registered Pcells to be used in the operating point annotation flow.

Arguments

None

Value Returned

<i>l_LibCellNames</i>	Returns the list of library and cell name pairs for the registered Pcells.
<i>nil</i>	The call is unsuccessful.

Example

The following example shows how you can use this function to get a list of registered Pcells:

```
sevRegisterPcellsForAnnotation(list("tsmcN5/analog_cmos_cell" "tsmcN5/  
hp_analog_cell" "tsmcN5_ArrayLib/mosfet_StackGate"))  
=> ("tsmcN5/analog_cmos_cell" "tsmcN5/hp_analog_cell" "tsmcN5_ArrayLib/  
mosfet_StackGate")
```

sevParametricTool

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

Description

Opens the Parametric analysis tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevCornersTool

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

Description

Opens the Corners tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevMonteCarloTool

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

Description

Opens the Analog Statistical Analysis tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevOptimizationTool

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

Description

Opens the Optimization tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevOpenCalculator

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

Description

Opens the Calculator.

Arguments

None

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevOpenDRLBrowser

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

Description

Opens the Browse Project Hierarchy window.

Arguments

None

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevOpenPlotWindow

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

Description

Opens a plot window.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevOpenPrintWindow

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

Description

Opens a print window.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevOpenJobMonitor

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

Description

Opens the Job monitor.

Arguments

None

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevlcon

```
sevlcon(  
    t_name  
)  
=> o_icon / nil
```

Description

Displays the icon object corresponding to the specified name.

Arguments

<i>t_name</i>	The name of the icon for the fixed menu/icon bar. This depends on the created icons. The existing Cadence simulation environment icons are: 'circuit, 'emitterFollower, 'ruler, 'alpha, 'analyses, 'acdc, 'dcTranAc, 'acTranDc, 'knobPanel, 'variables, 'pin, 'outputs, 'pencilEraser, 'scissors, 'vcrPlay, 'trafficGoSmall, 'trafficGo, 'trafficYellow, 'vcrStop, 'trafficStopSmall, 'trafficStop, 'waveform, 'fx, or 'sine.

Value Returned

<i>o_icon</i>	Returns the icon object that can be used in the icon strip.
<i>nil</i>	The named icon cannot be found.

sevDeleteSelections

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

Description

Deletes the selected items in any of the simulation environment list boxes.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevWhatsNew

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

Description

Opens the Whats New window for the simulation environment.

Arguments

None

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevStartSession

```
sevStartSession(  
    [ ?design g_design ]  
    [ ?lib g_lib ]  
    [ ?cell g_cell ]  
    [ ?view g_view ]  
    [ ?schematic g_schematic ]  
)  
=> t / nil
```

Description

Starts the Virtuoso Analog Simulation Environment session tied to the specified design. It will try the design first, then lib/cell/view, and finally it will try the schematic. If none of these is specified, it will start a skeleton session that will not be able to do anything until a design has been tied to the session. The lib/cell/view arguments must be specified, otherwise they are ignored.

Arguments

<code>?design <i>g_design</i></code>	The design object. The default value is <code>nil</code> .
<code>?lib <i>g_lib</i></code>	The library name. The default value is <code>nil</code> .
<code>?cell <i>g_cell</i></code>	The cell name. The default value is <code>nil</code> .
<code>?view <i>g_view</i></code>	The view name. The default value is <code>nil</code> .
<code>?schematic <i>g_schematic</i></code>	The schematic object. The default value is <code>nil</code> .

Value Returned

<code>t</code>	The call is successful.
<code>nil</code>	The call is unsuccessful.

sevAboutTool

```
sevAboutTool(  
    t_toolname  
)  
=> t / nil
```

Description

Generates and displays the standard About DFII window for the tool with the tool name in the message.

Arguments

<i>t_toolname</i>	Name of the tool calling <code>sevAboutTool</code> .

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevEditModels

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

Description

Displays the Model Library Setup form that lets you edit the Spectre direct model libraries.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetupStimuli

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

Description

Displays the Setup Analog Stimuli form that lets you specify the circuit stimuli.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetupSimulationFiles

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

Description

Displays the Simulation Files Setup form that lets you specify the simulation files and paths for Spectre direct simulation.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

t	The call is successful.
nil	The call is unsuccessful.

sevNetlistAndRun

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

Description

Forces the circuit to create a netlist and then run the simulation.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevRun

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

Description

Runs a simulation using the current netlist.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNetlistAndDebug

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

Description

Forces the circuit to netlist and runs the AHDL debugger.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevDebug

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

Description

Runs the AHDL debugger using the current netlist.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevLMGTool

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

Description

Opens the Transmission Line Modeler tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevPKGTool

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

Description

Opens the RFIC Package Modeler tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevKmodelTool

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

Description

Opens the RIFC Modeler for Cierito SPW tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevPCMTool

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

Description

Opens the Spiral Inductor Modeler tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevBPMTTool

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

Description

Opens the Bond Pad Modeler tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevBALMTool

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

Description

Opens the Transformer Modeler tool.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevActiveSelectedAna

```
sevActiveSelectedAna(  
    o_session  
)  
=> l_analyses / nil
```

Description

Returns the list of selected analyses that are currently enabled.

Arguments

<i>o_session</i>	The simulation environment session.

Value Returned

<i>l_analyses</i>	List of the selected, enabled analyses.
<i>nil</i>	There are no analyses selected in the analysis list box or all the selected analyses are not enabled.

sevNonActiveSelectedAna

```
sevNonActiveSelectedAna(  
    o_session  
)  
=> l_analyses / nil
```

Description

Returns the list of selected analyses that are currently not enabled.

Arguments

<i>o_session</i>	The simulation environment session.

Value Returned

<i>l_analyses</i>	The list of selected, non-enabled analyses.
<i>nil</i>	There are no analyses selected in the analysis list box or all the selected analyses are enabled.

sevSession

```
sevSession(  
    o_entity  
)  
=> t_session / nil
```

Description

Displays the session ID of the simulation environment such as the ADE L window, the schematic window associated with ADE L, or a form launched from ADE L.

Arguments

<i>o_entity</i>	The object with which the simulation environment session is associated. For example, form or window.

Value Returned

<i>t_session</i>	The session ID of the current simulation environment window.
<i>nil</i>	There is no simulation environment session object currently tied to the entity.

Examples

When you specify `hiGetCurrentWindow()` as the object, the function returns the session ID of the current ADE L window.

```
sevSession(hiGetCurrentWindow())  
=> sevSession2
```

When you specify `hiGetCurrentForm()` as the object, the function returns the session ID of the ADE L window associated with the currently opened form.

```
sevSession(hiGetCurrentForm())  
=> sevSession2
```

When you specify the *windowID* as the object, the function returns the session ID associated with the *windowID*.

```
sevSession(window(8))  
=> sevSession2
```

sevSetTopSaveDir

```
sevSetTopSaveDir(  
    o_session  
)  
=> t
```

Description

This function sets the ADE L Save State directory for the simulation environment session.

Arguments

<i>o_session</i>	The simulation environment session.
<i>t_dir</i>	The directory path to be set as the ADE L Save State directory.

Value Returned

<i>t</i>	Successful operation.

sevTopSaveDir

```
sevTopSaveDir(  
    o_session  
)  
=> t_dir
```

Description

Displays the current ADE L Save State directory path.

Arguments

<i>o_session</i>	The simulation environment session.

Value Returned

<i>t_dir</i>	Returns the ADE L Save State directory path.

sevDisplayViolations

```
sevDisplayViolations(  
    t_sevSession  
)  
=> o_form
```

Description

Displays the violations form. This function can be used only with an Analog Design Environment L session.

Arguments

<i>t_sevSession</i>	The simulation environment session.

Value Returned

<i>o_form</i>	Displays the violations form.

sevNoViolationsFound

```
sevNoViolationsFound(  
    s_sevSession  
)  
=> t / nil
```

Description

Determines if any violation file has been found in the results.

Arguments

<i>s_sevSession</i>	The simulation environment session.

Value Returned

<i>t</i>	No violation files are found in the results.
<i>nil</i>	Violations files are found in the results.

sevParasiticsDisplayed

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

Description

Determines whether the *Show Parasitics* menu will be disabled when the DC Operating Point results are available.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The <i>Show Parasitics</i> menu is disabled.
<i>nil</i>	The <i>Show Parasitics</i> menu is not disabled.

sevParasiticsNotDisplayed

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

Description

Determines if the *Hide Parasitics* menu will be disabled when the DC Operating Point results are available.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The <i>Hide Parasitics</i> menu is disabled.
<i>nil</i>	The <i>Hide Parasitics</i> menu is not disabled.

sevDevChecking

```
sevDevChecking(  
    t_sevSession  
)  
=> o_form / nil
```

Description

Displays the Analog Design Environment Device Checking Setup form.

This function can be used only with an Analog Design Environment L session.

Arguments

<i>t_sevSession</i>	The simulation environment session.

Value Returned

<i>o_form</i>	Returns the form object and displays the Device Checking Setup form if the call is successful.
<i>nil</i>	The call is unsuccessful.

Example

The following example displays the Analog Design Environment Device Checking Setup form.

```
sevDevChecking(sevSession)  
=> formStruct@0x38930e8
```

sevSetSolver

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

Description

Displays the *Choose Solver* form, which lets you select a solver.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	Displays the solver form if the call is successful.
<i>nil</i>	The call is unsuccessful.

sevSetConnectModules

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

Description

Displays the Connect Rules form that allows you to select built-in or user-defined connect rules.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevInvokeNCBrowse

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

Description

Displays the *NCBrowse* window.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevInvokeSimvision

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

Description

Displays the *SimVision Waveform* window.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevInvokeSimvisionDebugger

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

Description

Displays the *SimVision Debugger* interface with GUI options during an AMS session.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevNoLog

```
sevNoLog(  
    t_session  
    t_type  
)  
=> t / nil
```

Description

Checks if the specified log file exists for the AMS interface.

Arguments

<i>t_session</i>	The simulation environment session.
<i>t_type</i>	The type of log file.
	Valid Values: <code>compiler</code> , <code>elaborator</code> , and <code>simulator</code> .

Value Returned

<i>t</i>	The specified log file is not found.
<i>nil</i>	The specified log file is found.

sevViewNetlisterLog

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

Description

Displays the AMS netlister log file.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevViewCompilerLog

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

Description

Displays the AMS simulation compiler log file. The log file can be either `ncvlog.log` or `ncvhdl.log` depending on the contents of the AMS design.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevViewElabLog

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

Description

Displays the AMS simulation elaborator log file.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevViewNcverilogLog

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

Description

Displays the AMS simulation NcVerilog log file.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevViewSimLog

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

Description

Displays the AMS simulator log file.

Arguments

<i>t_session</i>	The simulation environment session.

Value Returned

<i>t</i>	The call is successful.
<i>nil</i>	The call is unsuccessful.

sevReturnVariablesWithEmptyValues

```
sevReturnVariablesWithEmptyValues(  
    o_session  
)  
=> t_string / nil
```

Description

Returns the set of variables in a session with empty values.

Arguments

<i>o_session</i>	The simulation environment session.

Value Returned

<i>t_string</i>	Returns a string containing variable names separated by comma. For example, "Var1, Var2".
nil	Unsuccessful operation

sevAddExpression

```
sevAddExpression(  
    o_session  
    t_expressionname  
    t_expression  
)  
=> t / nil
```

Description

Takes the name and expression as a string and adds a corresponding output in ADE session.

Arguments

<i>o_session</i>	A valid session passed as a string or symbol.
<i>t_expressionname</i>	Name of the expression to be added
<i>t_expression</i>	Expression passed as a string.

Value Returned

<code>list(nil errorString)</code>	Addition of expression failed.
<code>list(t outputStruct)</code>	Expression added successfully.

Example

The following example adds the specified expression to the output in ADE session.

```
sevAddExpression("sevSession1" "a" "1+1")
```

sevGetExpressions

```
sevGetExpressions (
    o_session
    [ ?axlTestName t_axlTestName ]
    [ ?namedOnly g_namedOnly ]
)
=> l_list
```

Description

Returns all the expressions in the specified ADE/ADEXL session.

Arguments

<i>o_session</i>	The simulation environment session.
<i>?axlTestName t_axlTestName</i>	Name of the test. If passed, method returns expressions to the corresponding test.
<i>?namedOnly g_namedOnly</i>	If passed, return only named expressions.

Value Returned

<i>l_list</i>	List of name value pairs.

Example

Returns all the named expressions corresponding to the test `aaa` in `sevSession1`.

```
sevGetExpressions("sevSession1" ?axlTestName "aaa" ?namedOnly t)
```

sevDeleteSelectedSubckts

```
sevDeleteSelectedSubckts(  
    t_sevSession  
    [ l_instanceList ]  
)  
=> t / nil
```

Description

Deletes the subcircuit instances selected in the *Save By Subckt Instances* pane of the simulation window.

Arguments

<i>t_sevSession</i>	The simulation environment session. For more information, see sevSession .
<i>l_instanceList</i>	List of subcircuit instances to be deleted from the <i>Save By Subckts Instances</i> pane.

Value Returned

<i>t</i>	The selected subcircuit instances are deleted.
<i>nil</i>	The specified simulation environment session object is invalid, or no instance is selected in the <i>Save By Subckt Instances</i> pane of the simulation window.

Examples

The following example deletes all the selected subcircuit instances in the simulation environment session, *sevSession1*:

```
sevSession(hiGetCurrentWindow())  
=> sevSession1  
  
sevDeleteSelectedSubckts('sevSession1')  
=> t
```


sevDeleteSelectedOpPoints

```
sevDeleteSelectedOpPoints(  
    t_sevSession  
    [ l_instanceList ]  
)  
=> t / nil
```

Description

Deletes the operating point instances selected in the *Save Operating Points* pane of the simulation window.

Arguments

<i>t_sevSession</i>	The simulation environment session. For more information, see sevSession .
<i>l_instanceList</i>	List of operating point instances to be deleted from the <i>Save Operating Points</i> pane.

Value Returned

<i>t</i>	The selected operating point instances are deleted.
<i>nil</i>	The specified simulation environment session object is invalid, or no instance is selected in the <i>Save Operating Points</i> pane of the simulation window.

Example

The following example deletes all the selected operating point instances in the simulation environment session, *sevSession1*.

```
sevSession(hiGetCurrentWindow())  
=> sevSession1  
  
sevDeleteSelectedOpPoints('sevSession1')  
=> t
```

CDF Functions

The Component Description Format (CDF) describes the parameters and their attributes for individual components and libraries of components. The Virtuoso CDF tools let you create, edit, and delete CDF data of components.

The CDF SKILL functions let you create routines to perform operations on multiple components at once, automating your handling of CDF descriptions.

These functions are broadly classified in the following categories.

- **Description creation functions:** These functions operate on CDF descriptions and let you create base and user CDF descriptions for libraries and cells.
 - ❑ [cdfCreateBaseCellCDF](#)
 - ❑ [cdfCreateBaseLibCDF](#)
 - ❑ [cdfCreateParam](#)
 - ❑ [cdfCreateUserCellCDF](#)
 - ❑ [cdfGetBaseLibCDF](#)
- **Description querying functions:** These function let you query existing CDF descriptions. To use the returned value, you must assign it to a variable that you create.
 - ❑ [cdfGetBaseLibCDF](#)
 - ❑ [cdfGetUserLibCDF](#)
 - ❑ [cdfGetLibCDF](#)
 - ❑ [cdfGetBaseCellCDF](#)
 - ❑ [cdfGetUserCellCDF](#)
 - ❑ [cdfGetCellCDF](#)
- **Save functions:** This function lets you save CDF descriptions. If a CDF description already exists, it is overwritten.

Virtuoso ADE SKILL Reference

CDF Functions

- ❑ [cdfSaveCDF](#)

- Dumping and editing functions: These functions let you save a CDF description to a file and edit it.

While editing dump files remember to escape invalid characters while specifying `termOrder` using symbols. All symbols must be preceded by the backslash (\) to make them valid symbols in SKILL.

- ❑ [cdfDump](#)

- ❑ [cdfDumpAll](#)

- Deletion functions: These functions let you delete a CDF description or its parameters.

- ❑ [cdfDeleteCDF](#)

- ❑ [cdfDeleteParam](#)

- Parameter management functions: These functions let you copy, find, and update CDF data and parameters.

- ❑ [aedCopyCDF](#)

- ❑ [aedDeleteCDF](#)

- ❑ [cdfCopyCDF](#)

- ❑ [cdfCopyParam](#)

- ❑ [cdfFindParamByName](#)

- ❑ [cdfRefreshCDF](#)

- ❑ [cdfUpdateInstParam](#)

- Scale factor functions: These functions let you specify scale factors or determine the current scale factors.

- ❑ [cdfDisableScaleFactorRetentionForZero](#)

- ❑ [cdfEditScaleFactors](#)

- ❑ [cdfEnableScaleFactorRetentionForZero](#)

- ❑ [cdfGetUnitScaleFactor](#)

- ❑ [cdfSetUnitScaleFactor](#)

- Edit CDF form invocation functions: These functions let you specify how the Edit CDF form opens. You can use `aedEditCDF` to specify initial library, cell, and type values.

- ☐ [aedEditCDF](#)
- ☐ [cdfGetCustomViaCDF](#)
- ☐ [cdfUpdateCustomViaParam](#)

■ Miscellaneous functions: These functions are not classified in any of the previous categories.

- ☐ [aedEditCDF](#)
- ☐ [cdfFormatFloatString](#)
- ☐ [cdfGetCustomViaCDF](#)
- ☐ [cdfParseFloatString](#)
- ☐ [cdfSyncInstParamValue](#)
- ☐ [cdfUpdateCustomViaParam](#)

■ Remastering functions: These functions let you register or unregister triggers used for remastering.

- ☐ [cdfRegRemasterTrigger](#)
- ☐ [cdfRemasterAction](#)
- ☐ [cdfUnregRemasterTrigger](#)

Related Topics

[CDF SKILL Function Properties](#)

CDF SKILL Function Properties

CDF SKILL functions use or operate on data IDs, parameters, parameter attributes, expressions, and global variables. The following topics describe each of these SKILL properties in the context of SKILL.

Cell and Library Data IDs

Before working on a CDF description, you must specify the data ID of the cell or library. You can get the data ID for the cell or library you are using with the `ddGetObj()` function. With SKILL versions earlier than 4.4, you used the `dmFindLib()` and `dmFindCell()` functions.

When creating the variable for the object ID of a cell, you must specify both the library name and the cell name:

```
test_cell = ddGetObj("analogLib" "schottky")
```

or

```
test_cell = ddGetObj(mylib-> name "schottky")
```

if you have already defined `mylib`.

You must then use `mylib` when a SKILL function requires a `d_id` or `g_libId`, and `test_cell` when a SKILL function requires a `d_id` or `g_cellId`.

You can use this data ID to access information about the description by using the right arrow (`->` or `~>`) operator. For example

```
test_cell-> name
```

returns the ID

```
"schottky"
```

Data Objects

CDF descriptions are represented by `g_cdfDataId` objects, which are SKILL objects that you can manipulate like other SKILL database objects. Just like data IDs, you assign the `g_cdfDataId` of a particular CDF description to a variable that you create. The most common variable name is `g_cdfDataId`. To create a new CDF data object for a new CDF description, use this type of ID assignment:

```
newCellId = cdfCreateUserCellCDF(test_cell)
cdf:25092160
newCellId->type
"userCellData"
```

To access an existing CDF data object for an existing CDF description, use this type of ID assignment:

```
baseCell = cdfGetBaseCellCDF(test_cell)
baseCell->dataFile->value
"bjt"
```

Virtuoso ADE SKILL Reference

CDF Functions

`newCellId` and `baseCell` are arbitrary names that you create.

Although you can add any information to a `g_cdfDataId`, an object uses the following elements to hold the information that it maintains:

- `id`
- `type`
- `parameters`
- `doneProc`
- `formInitProc`
- `Parameters`

id

The database object (cell ID or library ID) to which the `g_cdfDataId` is attached. This element is not editable.

type

There are seven types of `g_cdfDataId`:

<code>baseLibCDF</code>	base-level library CDF
<code>userLibCDF</code>	user-level library CDF
<code>effLibCDF</code>	effective-level library CDF
<code>baseCellCDF</code>	base-level cell CDF
<code>userCellCDF</code>	user-level cell CDF
<code>effCellCDF</code>	effective-level cell CDF
<code>effInstCDF</code>	effective-level instance CDF

This element is not editable.

parameters

List of parameters attached to a `g_cdfDataId`. This element is not editable.

Virtuoso ADE SKILL Reference

CDF Functions

doneProc

An optional procedure name that is used to validate changes to a parameter on a component instance as a part of post-processing. The procedure must take a single argument—the instance that has been modified.

The `doneProc` callback function can be used to modify the `cdfgForm` elements that are editable.

The default value is `nil`. This element is editable.

formInitProc

An optional procedure name. If specified, the procedure is executed when the contents of the CDF are displayed on a form. The default value is `nil`. This element is editable.

This procedure runs when you use the *Add Instance* and *Edit Properties* commands. You can use the procedure for preprocessing CDF data. This procedure must take a single argument, the `g_cdfDataId` being added to the form.

Note: When you modify a parameter value using the `formInitProc`, the Edit Properties Object command for the schematic application is not aware of the modification and does not update the changes made with `formInitProc`. You can avoid this problem by setting the variable `cdfgForm->cdfModified` to the value `t`.

Parameters

The following elements control the display of parameters on a form.

Element	Description	Default Value
<code>fieldWidth</code>	Width of a field	350 pixels
<code>fieldHeight</code>	Height of a field	35 pixels
<code>buttonFieldWidth</code>	Width of a button field	340 pixels
<code>promptWidth</code>	Width of a prompt	175 pixels

CDF Parameters

CDF parameters are represented by `g_cdfParamId` objects, which are SKILL objects that you can manipulate like other SKILL database objects. You can use a `g_cdfParamId` to access information about a parameter by using the right arrow (`->`) operator. Although you

Virtuoso ADE SKILL Reference

CDF Functions

can add any information to a *g_cdfParamId*, the object has elements to hold information that it maintains.

In addition to getting the *g_cdfParamIds* through the *parameters* element of a *g_cdfDataId*, you can also access a parameter by specifying:

g_cdfDataId->*paramName*

paramName is the name of the parameter. However, you must not create any user-defined properties on a *g_cdfDataId* that conflict with the name of a parameter on the *g_cdfDataId*.

Each parameter can use the following properties:

- use
- paramType
- defValue
- value
- prompt
- choices
- units
- editable
- callback
- parseAsNumber
- dontSave
- parseAsCEL
- storeDefault
- display

use

Specifies whether to use this parameter. The `use` attribute is context-specific and is evaluated when necessary. In this element, you can specify parameters that you are not using because of the value of other parameters, or because of the function being used. The `use` element must be a string.

Virtuoso ADE SKILL Reference

CDF Functions

- If the element evaluates to a value other than `nil`, the system uses the parameter.
- If a value for the element exists, the system ignores the value.
- If the element evaluates to `nil`, the system does not use the parameter.
- If you do not specify this element, `t` is assumed, so the system always uses the element.

This element is editable.

paramType

Type of the parameter. Valid values for this field are

`"string"`

`"int"`

`"float"`

`"radio"`

`"cyclic"`

`"Boolean"`

`"button"`

`"netSet"`

The quotation marks are required. CDF checks that the parameter value matches the type specified. You must specify this field.

This element is not editable.

Note: Use the `netSet` type to store inherited connections in CDF. Inherited connections allow you to selectively override global signals in designs created in Virtuoso Schematic Editor. The override information is communicated through net expressions and `netSet` properties.

defValue

Default value of the parameter. You must specify this element.

This element is editable.

value

Current value of the parameter.

This element is editable.

prompt

Prompt that appears on a form field when an application asks for the value of this parameter. The value for this field must be a string. You must specify this element.

This element is editable.

choices

Possible values (a list of strings) of a cyclic or radio type parameter. Do not use this element for other types of parameters.

This element is editable.

units

Unit type of the parameter. You often modify parameters that represent resistance, capacitance, length, time, power, and so forth, and expect to see the parameter value scaled to appropriate units (for example, pF, uM, dBm). Valid values for this element include the following strings:

"resistance"

"capacitance"

"inductance"

"conductance"

"time"

"frequency"

"power"

"powerDB"

"lengthMetric"

"lengthEnglish"

"angle"

"voltage"

"current"

"temperature"

The quotation marks are required. This element determines the unit's suffix and scale factor to use when displaying the parameter value. For example, by setting a parameter's `units` element to `capacitance`, the parameter value is displayed as 5 pF instead of 5e-12.

editable

Specifies whether a parameter is for display only. If set to `nil`, the element specifies a displayed parameter that you cannot change. (Other parameters' callbacks can change the parameter.) Use this element only on int, float, and string type fields.

The *editable* field must be a string and is evaluated when necessary.

- If the element evaluates to a value other than `nil`, the parameter is editable.
- If the element evaluates to `nil`, the parameter is grayed out when displayed as a form field.
- If you do not specify the element, `t` is assumed, meaning that the element is editable.

callback

(Optional) Specifies the SKILL code to be executed whenever the value of a parameter changes. Using `callback`, you can cause any parameter's value to affect any other parameter's value. The value of this element must be a string.

parseAsNumber

Specifies a string type parameter whose value can evaluate to a floating-point number. These types of parameters often occur for circuit-level simulators that allow component parameters to be either numbers or expressions containing variables.

If you specify this element, the system parses the value of the parameter to check if the value is a number.

- If the value is a number, the system converts the string to a floating point number, converts the string to the most efficient notation (taking into account any units field specified), then reconverts the number into a string.
- If the parameter value is not a number, the system does no conversion.

For example, if the parameter value is the string `5.4e-12` and the parameter value has a `units` element of `capacitance` (that is, the suffix is F), the parameter value is converted to the string `5.4p` before being displayed. However, if the parameter value is `c1`, no conversion takes place.

dontSave

Specifies that the parameter cannot be stored in any instance that corresponds to the component.

The value for the `dontSave` element must be a string. The system evaluates the element as necessary. If it evaluates to a value other than `nil`, the parameter is not stored. If you do not specify the element, `nil` is assumed and the parameter value is stored.

parseAsCEL

If set to `yes`, the associated CDF parameter is processed as a CDF Expression Language (CEL) expression. The parameter must be a string. If the expression resolves to a numeric value (the usual case), the `parseAsNumber` flag should also be set to `yes`.

storeDefault

Specifies whether the parameter default value is saved as a property of the instance. All tools based on the Cadence Analog Design Environment software use the CDF to find default values if no property exists on an instance.

If set to `no` (the default) or `don't use`, a property is not saved on the instance when the parameter value is the default. Also, if the default value of the parameter changes, all instances that use the default automatically get the new default value. (To see the change in an open window, you must select *Window – Redraw* from the Cadence menu.)

If set to `yes` and the parameter value is set to the same value as the default, a property is saved on the instance. One disadvantage of this attribute is that if the default value of a parameter changes, the instances that use the default do not automatically change to the new default. If you are a user of the Open Simulation Software (OSS) system, and you used to set this attribute to `yes` because you could only netlist using instance properties, you can now set it to `no` because OSS users now have the option of using CDF.

display

Determines if this parameter is displayed in forms that display CDF parameters, such as the Edit Object Properties form or the Add Instance form. You must enter `t`, `nil`, or a SKILL expression that evaluates to `t` or `nil` in this element to determine if this parameter is to be displayed. If the field evaluates to non-`nil` (the default), the parameter is displayed. If the element evaluates to `nil`, the parameter is not displayed.

Example

You can use the `g_cdfDataId baseCell` from the previous example to examine the parameters in the base-level cell CDF description of the Schottky transistor in the analog library. For example, that cell has a parameter `m`. You can access information about `m` in the following manner:

```
baseCell->m->prompt
"Multiplier"
baseCell->m->editable
nil
baseCell->m->display
"artParameterInToolDisplay('m) "
baseCell->m->paramType
"string"
```

CDF Expressions

CEL (CDF Expression Language) is another name for the Analog Expression Language (AEL) that works with CDF parameters. With AEL you can express a value as a mathematical expression instead of a single, fixed value. In such an expression, symbolic names such as `sheetResistivity` can refer to values that are computed and set on one of the following:

- CDF parameter on the same design component
- CDF parameter on the parent design component

Global Variables

CDF parameter values interact with each other, and one parameter's value can affect the existence of another. This feature is implemented primarily through the `use` and `callback` elements of the parameters section.

The following global variables, which you can access, are set whenever parameter fields are evaluated.

- `cdfgData`
- `cdfgForm`
- `gLabelsNumNotation`

cdfgData

Specifies the CDF data for the component in use. You can use this variable in the `doneProc` or `formInitProc` callback functions to either read or write data for a component. Use the `value` element to get the current value of any parameter in the CDF description.

- For creating an instance, set the variable to the last value you used when you created a component of this type.
- For editing, set the variable to the value for the component you are editing.

For example, you might set the `use` element for the `resistance` parameter to:

```
"cdfgData->resType->value == \"ideal\""
```

implying that the `resistance` parameter should be used only if the resistor type is set to `ideal`. (`==` is an equality test, not an assignment.)

When setting the value of a parameter, set

```
cdfgData->paramName->value = paramValue
```

If you use this setting, in the future you should be able to use the property list editor to edit the parameter values directly without going through the form.

cdfgForm

Specifies the form on which the CDF data is displayed, if there is one. You can modify data stored in `cdfgData` in the CDF `formInitProc`, or by modifying the `cdfgForm` fields with a callback function. When doing this, set the boolean variable `cdfgForm -> cdfModified` to `t`.

gLabelsNumNotation

Displays the `cdsTerm` and `cdsParam` values in different notations, such as scientific or engineering. The syntax of `gLabelsNumNotation` is as follows:

```
gLabelsNumNotation = 'suffix
```

The possible values are `'suffix`, `'scientific`, `'engineering`, `'simple` or `'default`.

The `default` value displays the labels according to the existing setting. For example, the `nmos` symbol shows the engineering notation, `300e-3` for `w`, by default.

To set the number of significant digits, use the `aelPushSignifDigits` function as follows:

```
aelPushSignifDigits(10)
```

Virtuoso ADE SKILL Reference

CDF Functions

Related Topics

CDF Functions

cdfCreateBaseCellCDF

```
cdfCreateBaseCellCDF(  
    g_cellID  
    [ ?doneProc t_doneProc ]  
    [ ?formInitProc t_formInitProc ]  
    [ ?fieldWidth x_fieldWidth ]  
    [ ?fieldHeight x_fieldHeight ]  
    [ ?buttonFieldWidth x_buttonFieldWidth ]  
    [ ?promptWidth x_promptWidth ]  
)  
=> g_cdfDataID / nil
```

Description

Creates a base-level CDF description for a cell. The CDF description is created with no parameters or simulation models.

Note the following:

- You must open the cell in write mode.
- Before using this function, ensure that the base-level CDF description does not already exist for the cell. If the CDF description already exists, this function will not update the existing CDF description.

Arguments

<i>g_cellID</i>	Specifies the cell ID for which the base-level CDF description is to be created.
?doneProc <i>t_doneProc</i>	Specifies an optional SKILL routine that is run to validate any parameter changes on the instantiation form.
?formInitProc <i>t_formInitProc</i>	Specifies an optional SKILL language routine that executes automatically when the component is placed on an instantiation form.
?fieldWidth <i>x_fieldWidth</i>	Specifies the width of a field on the instantiation form. The default width is 350 pixels.
?fieldHeight <i>x_fieldHeight</i>	

Virtuoso ADE SKILL Reference

CDF Functions

Specifies the height of a field on the instantiation form.
The default height is 35 pixels.

?buttonFieldWidth *x_buttonFieldWidth*

Specifies the width of a button on the instantiation form.
The default width is 350 pixels.

?promptWidth *x_promptWidth*

Specifies the width of the prompt on the instantiation form. The default width is 175 pixels.

Value Returned

g_cdfDataID The database object that represents the CDF description for the component.

nil When a library is not specified.

cdfCreateBaseLibCDF

```
cdfCreateBaseLibCDF(  
    g_libID  
    [ ?doneProc t_doneProc ]  
    [ ?formInitProc t_formInitProc ]  
    [ ?fieldWidth x_fieldWidth ]  
    [ ?fieldHeight x_fieldHeight ]  
    [ ?buttonFieldWidth x_buttonFieldWidth ]  
    [ ?promptWidth x_promptWidth ]  
)  
=> g_cdfDataID / nil
```

Description

Creates the Base Library CDF that is applied to all the devices in the library. The CDF description is created with no parameters or *simModels*.

Note the following:

- You must open the library in write mode.
- Before using this function, ensure that the base-level CDF description does not already exist for the library. If the CDF description already exists, this function will not update the existing CDF description.

Arguments

<code>g_libID</code>	This is the library ID.
<code>?doneProc t_doneProc</code>	Lets you specify an optional SKILL routine that is run to validate any parameter changes on the instantiation form.
<code>?formInitProc t_formInitProc</code>	Lets you specify an optional SKILL language routine that executes automatically when the component is placed on an instantiation form.
<code>?fieldWidth x_fieldWidth</code>	Lets you specify the width of a field on the instantiation form. The default width is 350 pixels.
<code>?fieldHeight x_fieldHeight</code>	Lets you specify the height of a field on the instantiation form. The default height is 35 pixels.

Virtuoso ADE SKILL Reference

CDF Functions

?buttonFieldWidth *x_buttonFieldWidth*

Lets you specify the width of a button on the instantiation form. The default width is 350 pixels.

?promptWidth *x_promptWidth*

Lets you specify the width of the prompt on the instantiation form. The default width is 175 pixels.

Value Returned

g_cdfDataID

The database object that represents the CDF description for the component.

nil

A parameter with the same name already exists.

cdfCreateParam

```
cdfCreateParam(  
    g_cdfDataId  
    [ ?name t_name ]  
    [ ?type t_type ]  
    [ ?defValue g_defValue ]  
    [ ?units t_units ]  
    [ ?parseAsNumber t_parseAsNumber ]  
    [ ?choices l_choices ]  
    [ ?prompt t_prompt ]  
    [ ?use t_use ]  
    [ ?display t_display ]  
    [ ?editable t_editable ]  
    [ ?dontSave t_dontSave ]  
    [ ?callback t_callback ]  
    [ ?storeDefault t_storeDefault ]  
    [ ?parseAsCEL t_parseAsCEL ]  
    [ ?description t_description ]  
)  
=> g_cdfDataId / nil
```

Description

Creates a parameter on the specified CDF database object with the specified attributes. The only attributes that are always required are the parameter's name and type. If this parameter description is not overriding an existing base-level parameter definition, you must also specify the default value.

Note the following:

- You cannot override a parameter's type. Also, CDF checks that the effective parameter is consistent and valid despite any overrides.
- If the name of a parameter already exists in the CDF description, the CDF description will not be updated.

Arguments

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
?name <i>t_name</i>	The name of the parameter.
?type <i>t_type</i>	The data type of the parameter.
?defValue <i>g_defValue</i>	The default value of the parameter.

Virtuoso ADE SKILL Reference

CDF Functions

<code>?units <i>t_units</i></code>	The unit suffix and scale factor to use when displaying the parameter value.
<code>?parseAsNumber <i>t_parseAsNumber</i></code>	Specifies whether the parameter can be evaluated to a floating-point number. Use this attribute only for string type parameters that contain numeric data. If <i>t_parseAsNumber</i> is <i>yes</i>, string is converted to a floating-point number, which is then converted the most efficient notation for execution, and then reconverts the floating-point number into a string. If <i>t_parseAsNumber</i> is <i>no</i> (the default value) or <i>don't use</i>, the system does no conversion. Use this setting if the parameter to be defined is a literal that contains numeric characters, such as the name of a file or a model. <i>t_parseAsNumber</i> must be used when the value of <i>t_parseAsCEL</i> is <i>yes</i>.
<code>?choices <i>l_choices</i></code>	The space- or comma-separated list of selections for a <i>cyclic</i> or <i>radio</i> data type. This attribute does not apply to other types of parameters. The choices depend on the following two cases: ■ If each choice is a single word, you can separate the choices with spaces. ■ If any choice is a group of words, you must separate each choice with a comma (,). (When using commas, do not leave extra spaces between choices because these spaces become part of the choice value.) A typical entry in the form field can be <code>choice 1,choice 2,choice 3</code> Notice that there is no space between the number 1 and the comma, or between the comma and the <i>c</i> in <i>choice</i>.
<code>?prompt <i>t_prompt</i></code>	The name for the CDF parameter that is displayed on the <i>Add Instance</i> or the <i>Edit Object Properties</i> form.

Virtuoso ADE SKILL Reference

CDF Functions

<code>?use t_use</code>	Determines if the parameter is to be used. You can enter Cadence SKILL language expression that evaluates to <code>t</code> or <code>nil</code> in this field to determine if this parameter is applicable. When the field evaluates to <code>nil</code> , the system never displays the parameter. When it evaluates to non- <code>nil</code> (the default), then based on the value of <code>t_display</code> , the system displays the parameter.
<code>?display t_display</code>	Determines if this parameter is displayed in forms that display CDF parameters, such as the <i>Edit Object Properties</i> form or the <i>Add Instance</i> form. You must enter <code>t</code> , <code>nil</code> , or a SKILL expression that evaluates to <code>t</code> or <code>nil</code> in this field to determine if this parameter is to be displayed. If the field evaluates to non- <code>nil</code> (the default), the parameter is displayed. If the field evaluates to <code>nil</code> , the parameter is not displayed.
<code>?editable t_editable</code>	Determines if this parameter can be edited in forms that display CDF parameters, such as the Edit Object Properties form or the Add Instance form. In this field, you can enter a SKILL expression that evaluates to <code>t</code> or <code>nil</code> in this field to determine if the parameter is editable. If the field evaluates to non- <code>nil</code> (the default), the parameter is editable. If the field evaluates to <code>nil</code> , the parameter is not editable. (If not editable, the parameter is grayed so that you can see the value but you cannot edit it.) This field is valid only for <code>string</code> , <code>int</code> , and <code>float</code> data types.
<code>?dontSave t_dontSave</code>	Determines if the parameter value is to be saved on the instance. This attribute is for programming use only. Typically, you should set this field to <code>nil</code> and not use it. If the field evaluates to <code>nil</code> (the default), the parameter value is saved as a property on the instance. If the field evaluates to non-<code>nil</code>, the parameter value is not saved as a property on the instance. Note: The <code>t_dontSave</code> attribute overrides the <code>t_storeDefault</code> setting. If both <code>t_storeDefault</code> and <code>t_dontSave</code> are set to <code>yes</code>, the default property is not saved.

Virtuoso ADE SKILL Reference

CDF Functions

?callback *t_callback*

Specifies a SKILL routine to be executed whenever the value of the parameter changes. The value of this optional field must be a string. The entered value can be an entire function to be executed or a call to a function that is defined elsewhere. If you do not enter anything in this field, the system assumes that there is no callback.

The CDF parameter callback is primarily a GUI-based callback. GUI-based callbacks occur when you modify the values in the parameter form fields. A GUI-based callback is active when the CDF parameters are displayed in the Add Instance form or the Edit Object Properties form if you use the *Create Instance* or *Edit Properties* commands.

?storeDefault *t_storeDefault*

Virtuoso ADE SKILL Reference

CDF Functions

Specifies whether to store the default value of a parameter as a parameter attribute on the instance. All tools based on the *Cadence Virtuoso Analog Design* software use the CDF description to find default values if there is no property on an instance.

If set to `no` or `don't use`, the default value that you set for a parameter is not preserved. In this case, if you modify the default value of a parameter, the change is reflected in all the existing instances and new instances of the component.

When the default value of the CDF parameter changes, all the instances including the ones already instantiated are updated with the new default value of the CDF parameter. To see the change in an open window, you must choose *View – Redraw*.

If set to `yes`, a parameter attribute that stores the default value of the parameter is added on the instance. Later if the default value for the parameter is modified, the instances that are already instantiated retain the old default value of the parameter. Only new instantiations have the new default value of the parameter.

One disadvantage of setting this attribute to `yes` is that if the default value of a parameter changes, the existing instances that use the default value do not automatically change to the new default value. That is, they retain the old default value.

Default value: `no`

`?parseAsCEL t_parseAsCEL`

Specifies whether the parameter is processed as a CDF Expression Language (CEL) expression. Use this attribute only for string type parameters.

If `t_parseAsCEL` is `yes`, the parameter is processed as a CEL expression. (Expressions such as `iPar(x)` or `pPar(x)`, indicating inheritance from an instance parameter or a parent parameter, are processed as CEL expressions.)

If `t_parseAsCEL` is `no` (the default), the parameter is not processed as a CEL expression.

Virtuoso ADE SKILL Reference

CDF Functions

`?description t_description`

Lets you specify a tooltip or description for each parameter. The text specified as description will be displayed on the *Edit Object Properties* and, *Create Instance* form, and the *Property Editor* assistant.

Value Returned

<code>g_cdfDataId</code>	The database object that represents the CDF description for the component.
<code>nil</code>	When a parameter with the same name already exists.

Examples

```
cdfParamId = cdfCreateParam(g_cdfDataId
?name "resistorType"
?type "radio"
?prompt "Type:"
?defValue "ideal"
?choices list("ideal" "thin film")
?callback "myModelTypeCB()")
```

Related Topics

[Component Parameters](#)

[Writing Callbacks](#)

[NBSIM Transistor CDF SKILL Description](#)

cdfCreateUserCellCDF

```
cdfCreateUserCellCDF(  
    g_cellId  
    [ ?doneProc t_doneProc ]  
    [ ?formInitProc t_formInitProc ]  
    [ ?fieldWidth x_fieldWidth ]  
    [ ?fieldHeight x_fieldHeight ]  
    [ ?buttonFieldWidth x_buttonFieldWidth ]  
    [ ?promptWidth x_promptWidth ]  
)  
=> g_cdfDataId / nil
```

Description

Creates a user-level CDF description for a cell. The CDF description is created with no parameters or simulation models.

Note: Before using this function, ensure that the user-level CDF description does not already exist for the cell. If the CDF description already exists, this function will not update the existing CDF description.

Virtuoso ADE SKILL Reference

CDF Functions

Arguments

<i>g_cellId</i>	This is the cell ID.
<i>?doneProc t_doneProc</i>	Lets you specify an optional SKILL routine that is run to validate any parameter changes on the instantiation form.
<i>?formInitProc t_formInitProc</i>	Lets you specify an optional SKILL language routine that executes automatically when the component is placed on an instantiation form.
<i>?fieldWidth x_fieldWidth</i>	Lets you specify the width of a field on the instantiation form. The default width is 350 pixels.
<i>?fieldHeight x_fieldHeight</i>	Lets you specify the height of a field on the instantiation form. The default height is 35 pixels.
<i>?buttonFieldWidth x_buttonFieldWidth</i>	Lets you specify the width of a button on the instantiation form. The default width is 350 pixels.
<i>?promptWidth x_promptWidth</i>	Lets you specify the width of the prompt on the instantiation form. The default width is 175 pixels.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<i>nil</i>	When a library is not specified.

cdfCreateUserLibCDF

```
cdfCreateUserLibCDF(  
    g_libId  
    [ ?doneProc t_doneProc ]  
    [ ?formInitProc t_formInitProc ]  
    [ ?fieldWidth x_fieldWidth ]  
    [ ?fieldHeight x_fieldHeight ]  
    [ ?buttonFieldWidth x_buttonFieldWidth ]  
    [ ?promptWidth x_promptWidth ]  
)  
=> g_cdfDataId / nil
```

Description

Creates the user-level library CDF that is applied to all the devices in the library. The user-level CDF can override entries in the base-level CDF. Therefore, a combination of the base-level CDF and the user-level CDF becomes the effective CDF.

The CDF description is created with no parameters or simulation models.

Note: Before using this function, ensure that the user-level CDF description does not already exist for the library. If the CDF description already exists, this function will not update the existing CDF description.

Virtuoso ADE SKILL Reference

CDF Functions

Arguments

<code>g_libId</code>	This is the library ID.
<code>?doneProc t_doneProc</code>	Lets you specify an optional SKILL routine that is run to validate any parameter changes on the instantiation form.
<code>?formInitProc t_formInitProc</code>	Lets you specify an optional SKILL language routine that executes automatically when the component is placed on an instantiation form.
<code>?fieldWidth x_fieldWidth</code>	Lets you specify the width of a field on the instantiation form. The default width is 350 pixels.
<code>?fieldHeight x_fieldHeight</code>	Lets you specify the height of a field on the instantiation form. The default height is 35 pixels.
<code>?buttonFieldWidth x_buttonFieldWidth</code>	Lets you specify the width of a button on the instantiation form. The default width is 350 pixels.
<code>?promptWidth x_promptWidth</code>	Lets you specify the width of the prompt on the instantiation form. The default width is 175 pixels.

Value Returned

<code>g_cdfDataId</code>	The database object that represents the CDF description for the component.
<code>nil</code>	When a library is not specified.

cdfGetBaseLibCDF

```
cdfGetBaseLibCDF(  
    g_libId  
)  
=> g_cdfDataId / nil
```

Description

Returns the base-level CDF description attached to a library. If one is not defined, it returns `nil`.

Arguments

<i>g_libId</i>	Specifies the ID of the library for which you want to view the base-level CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<code>nil</code>	When a library is not specified.

cdfGetUserLibCDF

```
cdfGetUserLibCDF(  
    g_libId  
)  
=> g_cdfDataId / nil
```

Description

Returns the user-level CDF description attached to a library. If one is not defined, it returns `nil`.

Arguments

<i>g_libId</i>	Specifies the ID of the library for which you want to view the user-level CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<code>nil</code>	When a library is not specified.

cdfGetLibCDF

```
cdfGetLibCDF(  
    g_libId  
)  
=> g_cdfDataId / nil
```

Description

Returns the effective CDF description attached to a library. If neither a base- nor user-level CDF description is defined, it returns `nil`. The resulting CDF description represents the overlay of the user-level CDF on the base-level CDF.

Arguments

<i>g_libId</i>	Specifies the ID of the library for which you want to view the CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<code>nil</code>	When a library is not specified.

cdfGetBaseCellCDF

```
cdfGetBaseCellCDF(  
    g_cellId  
)  
=> g_cdfDataId / nil
```

Description

Returns the base-level CDF description attached to a cell. If one is not defined, it returns `nil`.

Arguments

<i>g_cellId</i>	Specifies the ID of the cell for which you want to view the base-level CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<code>nil</code>	When a cell is not specified.

cdfGetUserCellCDF

```
cdfGetUserCellCDF(  
    g_cellId  
)  
=> g_cdfDataId / nil
```

Description

Returns the user-level CDF description attached to a cell. If one is not defined, it returns `nil`.

Arguments

<i>g_cellId</i>	Specifies the ID of the cell for which you want to view the user-level CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<code>nil</code>	When a cell is not specified.

cdfGetCellCDF

```
cdfGetCellCDF(  
    g_cellId  
)  
=> g_cdfDataId / nil
```

Description

Returns the effective CDF description attached to a cell. If neither a base- nor user-level CDF description is defined for the cell or its library, it returns `nil`. The resulting CDF description represents the overlay of the user-level cell CDF on the base-level cell CDF on the user-level library CDF on the base-level library CDF.

Arguments

<i>g_cellId</i>	Specifies the ID of the cell for which you want to view the base-level CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<code>nil</code>	When a base or user-level CDF description is not defined for the specified cell or its library.

cdfGetInstCDF

```
cdfGetInstCDF(  
    d_instId  
)  
=> g_cdfDataId / nil
```

Description

Returns the effective CDF description associated with an instance.

The difference between the instance's effective CDF description and the cell's effective CDF description is that the values of any CDF parameter takes into account the values of the parameters stored on the instance.

Arguments

<i>d_instId</i>	Specifies the ID of the instance for which you want to view the effective CDF description.

Value Returned

<i>g_cdfDataId</i>	The database object that represents the CDF description for the component.
<i>nil</i>	When an instance is not specified.

cdfSaveCDF

```
cdfSaveCDF(  
    g_cdfDataId  
)  
=> t / nil
```

Description

Saves a CDF description to disk. The CDF description is then read in every time you open the cell of the library to which the description is attached. You can save only base-level CDF descriptions.

You must have write permission for the object to which the CDF description is attached to execute this function.

Arguments

<i>g_cdfDataId</i>	Specifies the database object that represents the CDF description to be saved.

Value Returned

<i>t</i>	Saves the specified CDF description.
<i>nil</i>	When you do not have the write permissions for the object to which the CDF description is attached.

cdfDump

```
cdfDump(  
    t_libName  
    t_fileName  
    [ ?cellName t_cellName ]  
    [ ?level s_level ]  
    [ ?edit g_edit ]  
)  
=> t / nil
```

Description

Dumps the CDF description for the specified library and cell into the given file. If a cell is not specified, then only the library CDF description is dumped. The file is created in the current working directory or the directory specified with the file name. If *g_edit* is *t*, a text editor window is automatically opened on *t_fileName*. The default is no editor.

You can also specify the level of the CDF description by using the *?level s_level* argument.

Virtuoso ADE SKILL Reference

CDF Functions

Arguments

<i>t_libName</i>	Specifies the name of the library for which the CDF description is to be saved to the given file.
<i>t_fileName</i>	Specifies the name of the file to which the CDF descriptions for the given library and cell are to be saved.
?cellName <i>t_cellName</i>	Specifies the name of the cell for which the CDF description is to be saved to the given file.
?level <i>s_level</i>	Specifies the level of the CDF description to be displayed. ■ base: Saves the base-level CDF descriptions to the specified file. ■ user: Saves the user-level CDF descriptions to the specified file. This is the default.
?edit <i>g_edit</i>	Specified whether the specified file should be opened in a text editor. ■ t: Opens the specified file in a text editor. ■ nil: The specified file is not opened in a text editor.

Value Returned

<i>t</i>	Dumps the CDF description for the specified library and cell to the given file.
<i>nil</i>	When the specified library or cell is invalid.

Examples

```
cdfDump( "nmos" "tr.mod" ?cellName "pnp" ?level 'base
        ?edit t)
```

cdfDumpAll

```
cdfDumpAll(  
    t_libName  
    t_fileName  
    [ ?level s_level ]  
    [ ?edit g_edit ]  
)  
=> t / nil
```

Description

Dumps the CDF description for the specified library and all its cells into the given file. You can also specify the level of the CDF description by using the `?level s_level` argument. The specified file is created in the current working directory or the directory specified with the filename.

If the `?edit g_edit` argument is set to `t`, the specified file is opened in a text editor window. The default is no editor.

Virtuoso ADE SKILL Reference

CDF Functions

Arguments

<code>t_libName</code>	Specifies the name of the library for which the CDF description is to be saved to the given file.
<code>t_fileName</code>	Specifies the name of the file to which the CDF descriptions for the given library and cell are to be saved.
<code>?level s_level</code>	Specifies the level of the CDF description to be displayed. ■ <code>base</code>: Saves the base-level CDF descriptions to the specified file. ■ <code>user</code>: Saves the user-level CDF descriptions to the specified file. This is the default.
<code>?edit g_edit</code>	Specifies whether the given file should be opened in a text editor. ■ <code>t</code>: Opens the specified file in a test editor. ■ <code>nil</code>: The specified file is not opened in a text editor.

Value Returned

<code>t</code>	Dumps the CDF description for the specified library and cell to the given file.
<code>nil</code>	When the specified library or cell is invalid.

Examples

```
cdfDumpAll("asic" "lib.mod" ?level 'base ?edit t)
```

cdfDeleteCDF

```
cdfDeleteCDF(  
    g_cdfDataId  
)  
=> t / nil
```

Description

Deletes a CDF description, including all attached parameters.

If the CDF description has been saved, the saved versions are also deleted. If this is a base-level CDF description, you must have write permissions for the object to which the CDF description is attached.

Arguments

<i>g_cdfDataId</i>	Specifies the database object that represents the CDF description to be deleted.

Value Returned

t	Deletes the CDF description, including all attached parameters.
nil	When you do not have the write permissions for the object to which the CDF description is attached.

cdfDeleteParam

```
cdfDeleteParam(  
    g_cdfParamId  
)  
=> t / nil
```

Description

Deletes the specified CDF parameter.

If the CDF parameter is attached to a base-level CDF description, you must have the write permissions for the object to which the CDF description is attached.

CDF checks that no invalid parameter descriptions would result from deleting the specified parameter before it is deleted. This would occur if you tried to delete a base-level parameter and a user-level parameter is defined that only partially overrides the base-level description.

Arguments

<i>g_cdfParamId</i>	Specifies the ID of the CDF parameter to be deleted.

Value Returned

<i>t</i>	Deletes the specified CDF parameter.

<i>nil</i>	Returns <i>nil</i> if:

- Invalid parameter descriptions would result from deleting the specified parameter
- You do not have the write permissions for the object to which the CDF description is attached

cdfCopyCDF

```
cdfCopyCDF(  
    g_cellId | g_libId  
    t_dataType  
    g_sourceCdfDataId  
    )  
=> g_cdfDataId / nil
```

Description

Copies the CDF data of the specified type from specified source to a library or cell by creating a new CDF data ID.

The destination must be specified as the ID of a library or a cell. The destination library or cell must not already have a CDF description of the specified CDF data type.

Note the following:

- If you have specified the type `baseCellData` or `baseLibData`, ensure that the destination library or cell has the appropriate write permission.
- You cannot copy effective-level CDF data.

Virtuoso ADE SKILL Reference

CDF Functions

Arguments

<i>g_cellId g_libId</i>	Specifies the ID of the library or cell where the CDF data must be copied.
<i>t_dataType</i>	Specifies the type of CDF data to be copied. ■ <code>baseLibData</code>: Base-level CDF data of the specified library. ■ <code>userLibData</code>: User-level CDF data of the specified library. ■ <code>baseCellData</code>: Base-level CDF data of the specified cell ■ <code>userCellData</code>: User-level CDF data of the specified cell. Note: If you have specified the type <code>baseCellData</code> or <code>baseLibData</code>, ensure that the destination library or cell has the appropriate write permission.
<i>g_sourcecdfDataId</i>	Specifies the source of the CDF data to be copied.

Value Returned

<i>g_cdfDataId</i>	ID of the new CDF data returned if the copy operation was successful.
<i>nil</i>	When the copy operation fails.

Examples

Copies the base-level CDF description of `nbsim` in `analogLib` to `nfet` in `bicmos`.

```
source_cell = ddGetObj( "analogLib" "nbsim" )
srcCDF = cdfGetBaseCellCDF( source_cell )
dest_cell = ddGetObj( "bicmos" "nfet" )
if( destCDF = cdfGetBaseCellCDF(dest_cell) then
    cdfDeleteCDF( destCDF)
)
destCDF = cdfCopyCDF( dest_cell "baseCellData" srcCDF
```

cdfCopyParam

```
cdfCopyParam(  
    g_cdfDataId  
    g_cdfParamId  
)  
=> g_cdfParamId / nil
```

Description

Copies the specified parameter to the given CDF data object.

g_cdfDataId must not already have a parameter with the same name. The parameter and the data ID must not be effective objects.

Arguments

<i>g_cdfDataId</i>	Specifies the ID of the CDF data object where the parameter is to be copied.
<i>g_cdfParamId</i>	Specifies the ID of the CDF parameter to be copied to the CDF data object.

Value Returned

<i>g_cdfParamId</i>	The updated ID of the parameter that was copied to the CDF data object.
<i>nil</i>	When the specified parameter already exists in the specified CDF data object.

cdfFindParamByName

```
cdfFindParamByName(  
    g_cdfDataId  
    t_name  
)  
=> g_cdfParamId / nil
```

Description

Returns the parameter ID of the specified parameter name for the given CDF description, if it exists. If not, it returns `nil`.

Use this function to search for parameters by name.

Arguments

<i>g_cdfDataId</i>	Specifies the ID of the CDF data object where the specified parameter is to be searched.
<i>t_name</i>	Specifies the name of the parameter to be searched.

Value Returned

<i>g_cdfParamId</i>	The ID of the specified parameter.
<code>nil</code>	The specified parameter does not exist in the given CDF data object.

cdfUpdateInstParam

```
cdfUpdateInstParam(  
    d_instId  
)  
=> t / nil
```

Description

Stores the CDF parameters specified in the effective cell CDF of the instance master to the given instance. If a `doneProc` post-processing procedure is specified, the function runs this procedure to validate any updates to the instance. When the ID given is not for an instance or the instance master does not have CDF definition, it returns `nil`.

Arguments

<i>d_instId</i>	Specifies the ID of the instance where the CDF parameters specified in the effective cell CDF of the instance master are to be saved.

Value Returned

<code>t</code>	Saves the CDF parameters specified in the effective cell CDF of the instance master to the given instance
<code>nil</code>	The specified ID is incorrect or the instance master does not have the CDF definition.

cdfRefreshCDF

```
cdfRefreshCDF(  
    g_libId / g_cellId  
)  
=> t / nil
```

Description

Updates the CDF structure in the memory for the specified library or cell Id with the contents stored on the hard disk. Returns `nil` if the CDF structures for the specified library and cell ID are not present in memory.

If there are multiple Virtuoso sessions and you modify and save a CDF in one session, then that CDF in other sessions would be outdated. In such a case, use the `cdfRefreshCDF` function to update the CDF in other sessions.

Note: Use this function with caution. In case multiple users modify the CDF parameters of a common component simultaneously, the saved parameter pointers may become invalid as there is no way of notifying the application after refreshing the CDF parameter values.

Arguments

g_libId or *g_cellId* Specifies the *dd_id* of the library or the cell.

Return Value

<code>t</code>	Updates the CDF structure in the memory for the specified library or cell Id with the contents stored on the hard disk.
<code>nil</code>	The CDF structures for the specified library and cell ID do not exist in the memory

Examples

```
cdfRefreshCDF(ddGetObj("mylib" "nnpnpar"))
```

aedCopyCDF

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

Description

Opens the Copy Component CDF form.

Arguments

None

Return Value

t	Opens the Copy Component CDF form.
nil	The Copy Component CDF form is not opened.

aedDeleteCDF

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

Description

Opens the Delete Component CDF form.

Arguments

None

Return Value

t	Opens the Delete Component CDF form.
nil	The Delete Component CDF form is not opened.

cdfGetUnitScaleFactor

```
cdfGetUnitScaleFactor(  
    t_unitName  
)  
=> t_scaleFactor
```

Description

Displays the current scale factor for the specified unit.

Arguments

<i>t_unitName</i>	Specifies the unit name for which you want to display the scale factor.

Value Returned

<i>t_scaleFactor</i>	Returns the current scale factor for the specified unit name.

Examples

Returns the scale factors for power:

```
cdfGetUnitScaleFactor("power")
```

cdfSetUnitScaleFactor

```
cdfSetUnitScaleFactor(  
    t_unitName  
    t_scaleFactor  
)  
=> t / nil
```

Description

Sets the scale factor for the specified unit.

Arguments

<i>t_unitName</i>	Specifies the unit name for which you want to set the scale factor.
<i>t_scaleFactor</i>	Specifies the scale factor for the specified unit name.

Value Returned

<i>t</i>	Sets the scale factor for the specified unit.
<i>nil</i>	The scale factor is not set.

Examples

Sets the scale factor for the unit `lengthMetric` to `m` (millimeters):

```
cdfSetUnitScaleFactor("lengthMetric" "m")
```

cdfEditScaleFactors

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

Description

Displays the Units Scaling Factors form which can be used to set scaling factors for displaying CDF parameters.

Arguments

None

Value Returned

t	Displays the Units Scaling Factors form.
nil	The Units Scaling Factors form is not opened.

cdfEnableScaleFactorRetentionForZero

```
cdfEnableScaleFactorRetentionForZero(  
    )  
=> t
```

Description

Lets you retain the unit scale factor for a CDF parameter if its specified default value is 0 followed by the scale factor. This is possible only when the scale factor for a unit type is set to `auto`.

By default, in such cases, the value of the CDF Parameter is converted to 0.

Arguments

None

Value Returned

<code>t</code>	Retains the unit scale factor for a CDF parameter if its specified default value is 0 followed by the scale factor.

Examples

The `cdfFormatFloatString` function in the following example returns `0u` when called after the `cdfEnableScaleFactorRetentionForZero` function.

```
cdfEnableScaleFactorRetentionForZero()  
cdfFormatFloatString("0u" "auto") => "0u"
```

Related Topics

[cdfFormatFloatString](#)

cdfDisableScaleFactorRetentionForZero

```
cdfDisableScaleFactorRetentionForZero(  
    )  
=> t
```

Description

Disables the unit scale factor retention for CDF parameters enabled using the `cdfEnableScaleFactorRetentionForZero` function.

Arguments

None

Value Returned

t	Disables the unit scale factor retention for CDF parameters enabled using the <code>cdfEnableScaleFactorRetentionForZero</code> function.

Related Topics

[cdfEnableScaleFactorRetentionForZero](#)

cdfParseFloatString

```
cdfParseFloatString(  
    t_string  
)  
=> nil / d_value / t_string
```

Description

Uses the standard `strtod` (string to double) function to parse the input string. When the input string contains trailing non-numerical characters, the fragment of the string is compared against a supported set of scale factor designators.

Arguments

<i>t_string</i>	Specifies the string to be parsed.

Value Returned

<i>nil</i>	When the input string cannot be parsed as a float value or without a valid scale factor, as shown below:

```
cdfParseFloatString("lg") => nil
```

<i>d_value</i>	Specifies that the input string can be parsed as a float value with or without a valid scale factor, as shown below:

```
cdfParseFloatString("1.0") => 1.0  
cdfParseFloatString("1.0u") => 1e-06
```

<i>t_string</i>	Specifies that the input string does not contain a valid numerical representation for a float value. For example, the input string starting with a non-digit character as shown below:

```
cdfParseFloatString("abcd") => "abcd"
```

Related Topics

NFET Examples

cdfFormatFloatString

```
cdfFormatFloatString(  
    t_string  
    t_scaleFactor  
)  
=> nil / t_val
```

Description

This function formats the input string into a value representation, if possible. It formats the input string using the input scale factor, re-converts the value to a string, and then returns the formatted string value. If the input string cannot be converted, the input string is returned with no change to it.

Arguments

<i>t_string</i>	Specifies a string representing a float value
<i>t_scaleFactor</i>	Specifies a string representing a scale factor

Value Returned

<i>nil</i>	When the given <i>t_scaleFactor</i> is invalid
<i>t_val</i>	Specifies that the input string can be formatted using the input scale factor. Else, the input string is returned without any change to it.

Examples

```
cdfFormatFloatString("123.4" "m") => "123400.0m"  
cdfFormatFloatString("10000" "M") => "0.01M"
```

cdfSyncInstParamValue

```
cdfSyncInstParamValue(  
    d_instId1  
    d_instId2  
)  
=> t / nil
```

Description

This function generates all the CDF parameters for the first instance (*d_instId1*) and updates the second instance (*d_instId2*) with the same values. Both the instances must share the same cell.

Arguments

<i>d_instId1</i>	Specifies the instance for which the CDF parameters are to be generated.
<i>d_instId2</i>	Specifies the instance for which the CDF parameter values are to be updated.

Value Returned

<i>t</i>	Updates the CDF parameter values for the second instance with the same values as that of the first instance.
<i>nil</i>	When the specifies instances do not share the same cell.

cdfUpdateInstSingleParam

```
cdfUpdateInstSingleParam(  
    d_instId  
    t_paramName  
)  
=> t / nil
```

Description

Copies the specified parameter's effective value to the specified instance.

Arguments

<i>d_instId</i>	Specifies the instance to which the given parameter's effective value is to be copied.
<i>t_paramName</i>	Specifies the name of the parameter whose effective value is to be copied to the given instance.

Value Returned

<i>t</i>	Copies the specified parameter's effective value to the given instance.
<i>nil</i>	When the specified parameter or instance does not exist.

aedEditCDF

```
aedEditCDF(  
    [ ?libName t_libraryName ]  
    [ ?cellName t_cellName ]  
    [ ?cdfType t_cdfType ]  
)  
=> t
```

Description

Opens the Edit CDF form to the specified library, cell, and CDF type. The library and cell names must be strings referring to an existing library or cell, and the CDF type must be *effective*, *base*, or *user*.

Arguments

?libName t_libraryName

Specifies the name of the library for which the Edit CDF form is to be opened.

?cellName t_cellName

Specifies the name of the cell for which the Edit CDF form is to be opened.

?cdfType t_cdfType

Specifies the CDF type.

- *effective*: Specifies an effective CDF description.
- *base*: Specifies a base-level CDF description.
- *user*: Specifies a user-level CDF description.

Value Returned

t

Opens the Edit CDF form to the specified library, cell, and CDF type.

nil

When the specified library, cell, or CDF type is invalid.

cdfGetCustomViaCDF

```
cdfGetCustomViaCDF(  
    d_customViaId  
)  
=> g_cdfDataId / nil
```

Description

Returns the effective CDF description associated with a custom via or returns nil. When the custom via's cell or library has a base or user-level CDF defined, it returns the *g_cdfDataId*, otherwise returns nil.

Arguments

<i>d_customViaId</i>	Specifies the ID of the custom via for which the effective CDF description is to be displayed.

Value Returned

<i>g_cdfDataId</i>	Opens the Edit CDF form to the specified library, cell, and CDF type.
nil	When the custom via's cell or library does not have a defined base or user-level CDF description.

cdfUpdateCustomViaParam

```
cdfUpdateCustomViaParam(  
    d_customViaId  
)  
=> t / nil
```

Description

Stores the parameters specified in the effective cell CDF of the custom via in the specified custom via instance. When the specified ID is not for a customVia instance or the instance master does not have a CDF definition, it returns `nil`.

Arguments

<i>d_customViaId</i>	Specifies the ID of the custom via for which the effective CDF description is to be displayed.

Value Returned

<code>t</code>	Opens the Edit CDF form to the specified library, cell, and CDF type.
<code>nil</code>	When the specified ID is not for a custom via instance or the instance master does not have a CDF definition.

cdfRegRemasterTrigger

```
cdfRegRemasterTrigger(  
    t_triggerFunc  
)  
=> l_ids / nil
```

Description

Registers a trigger that is called after instance remastering to set, map, or clear the parameter values on the remastered instance. The registered trigger function is only called when the resetOnRemaster environment variable is set to `t`.

It is possible to register a trigger before defining the function. No capturing of the function is performed and only the most up-to-date implementation is called. Only one trigger can be registered and any existing registration is replaced.

Arguments

t_triggerFunc

The name of the user-specified remastering function.

The function must be defined in the following format:

```
(d_instId d_newMasterCVID) => l_returnDPL
```

Here,

- `d_instId` is the database instance ID being remastered
- `d_newMasterCVID` is the ID of the new master
- `l_returnDPL` is a DPL created by the `cdfRemasterAction` function

The `cdfRemasterAction` SKILL function is used to build the return DPL.

Virtuoso ADE SKILL Reference

CDF Functions

Value Returned

<code>t</code>	The user-specified trigger was registered.
<code>nil</code>	The argument cannot be converted to a string.

Examples

Defines a remaster function. The callback `MyRemasterFunc` specifies:

1. The `formInitProc` of the new master is not called.
2. The `formDoneProc` of the new master is called.
3. The property `p1` is set to the value 1 and the property values `p2` and `p3` are set to the instance value before remastering.

```
(defun MyRemasterFunc (instId newMasterCvId) cdfRemasterAction(?noFormInitCall t
?noFormDoneCall nil ?paramList list(list("p1" 1) "p2" "p3")))
```

Registers the remaster function.

```
cdfRegRemasterTrigger("MyRemasterFunc")
=>t
```

Performs the remastering on an instance.

```
dbRemasterAnyInst(inst new_cv)
=>t
```

Related Topics

[`cdfRemasterAction`](#)

[`cdfUnregRemasterTrigger`](#)

cdfRemasterAction

```
cdfRemasterAction(  
    [ ?noFormInitCall g_noFormInitCall ]  
    [ ?noFormDoneCall g_noFormDoneCall ]  
    [ ?paramList g_paramList ]  
    [ ?paramCallbacks g_paramCallbacks ]  
  
    )  
=> l_returnDPL
```

Description

Helper function for the CDF remastering callback function that is registered using `cdfRegRemasterTrigger`. The callback function is expected to return a SKILL disembodied property list (DPL) having specific named properties. The `cdfRemasterAction` function constructs this DPL from its key arguments.

The callback function must be defined in the following format:

```
(d_instId d_newMasterCVID) => l_returnDPL
```

Here,

- `d_instId` is the database instance ID being remastered
- `d_newMasterCVID` is the ID of the new master
- `l_returnDPL` is a DPL created by the `cdfRemasterAction` function

Arguments

`?noFormInitCall` *g_noFormInitCall*

Boolean that specifies whether the CDF form initialization procedure of the instance should be called during property value application. The procedure is called only if the value is `nil`, and not called if the value is `t`. Other boolean trues such as 1 or "some value" are an error.

The default is `nil`.

Virtuoso ADE SKILL Reference

CDF Functions

`?noFormDoneCall` *g_noFormDoneCall*

Boolean that specifies whether the CDF form done procedure of the instance should be called during property value application. The procedure is called only if the value is `nil`, and not called if the value is `t`. Other boolean trues such as `1` or "some value" are an error.

The default is `nil`.

`?paramList` *g_paramList*

List of parameter name-value pairs in the following format:

```
(list t_paramName g_paramValue)
```

For example:

```
?paramList list(list("l" 10e-9) list("model"
"newModel") "m")
```

Here, running `cdfRemasterAction` with the above example ensures that the remastered instance has three properties, where `l` is set to `10e-9`, `model` is set to `newModel`, and `m` is set to the value on the instance before remastering.

When `?paramList` is set to `nil`, no properties are preserved or mapped.

The default is `nil`.

Virtuoso ADE SKILL Reference

CDF Functions

`?paramCallbacks` *g_paramCallbacks*

Controls the parameter callbacks executed on the remastered instance.

Possible values are:

- `'none`: Does not run any parameter callbacks.
- `'all`: Calls the callbacks for all the parameters of the new master cell CDF. Here, callbacks will be called for CDF parameters which are part of `paramList` first followed by callbacks of the rest of the new master cell CDF parameters.
- `'paramList`: Runs only the callbacks for parameters specified in the `paramList` argument and in the order specified.
- `list`: List of strings specifying the parameter names and sequence to trigger callbacks in the following format:

```
list(t_param1 t_param2 ...)
```

For example, to first run the `model` and then the `l` parameter, specify the following:

```
?paramCallbacks list("model" "l")
```

The default is `'paramList`.

Value Returned

l_returnDPL

A disembodied property list (DPL) used by the remaster infrastructure.

Examples

Defines a remaster function. The callback `MyRemasterFunc` function specifies:

1. The `formInitProc` of the new master is not called.
2. The `formDoneProc` of the new master is called.
3. The property `p1` is set to the value 1 and the property values `p2` and `p3` are set to the instance value before remastering.
4. The `?paramCallback` key argument is not specified and the default `'paramList` is used.

Virtuoso ADE SKILL Reference

CDF Functions

```
(defun MyRemasterFunc (instId newMasterCvId) cdfRemasterAction(?noFormInitCall t
?noFormDoneCall nil ?paramList list(list("p1" 1) "p2" "p3")))
```

Registers the remaster function.

```
cdfRegRemasterTrigger("MyRemasterFunc")
=>t
```

Performs the remastering on an instance.

```
dbRemasterAnyInst(inst new_cv)
=>t
```

Related Topics

[cdfRegRemasterTrigger](#)

[cdfUnregRemasterTrigger](#)

cdfUnregRemasterTrigger

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

Description

Unregisters a trigger previously registered with `cdfRegRemasterTrigger`. The callback infrastructure is only called when the `resetOnRemaster` environment variable is set to `t`.

It is possible to register a trigger before defining the function. No capturing of the function is performed and only the most up-to-date implementation is called. Only one trigger can be registered and any existing registration is replaced.

Arguments

None

Value Returned

<code>t</code>	The user-specified trigger was registered.
<code>nil</code>	The argument cannot be converted to a string.

Examples

Registers the remaster function.

```
cdfRegRemasterTrigger("MyRemasterFunc")  
=>t
```

Unregisters the remaster function.

```
cdfUnregRemasterTrigger("MyRemasterFunc")  
=>t
```

Related Topics

[cdfRegRemasterTrigger](#)

[cdfRemasterAction](#)