

Cadence User Interface SKILL Reference

**Product Version IC23.1
September 2023**

© 2023 Cadence Design Systems, Inc.

Printed in the United States of America.

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

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

<u>Introduction to Cadence User Interface</u>	1
<u>Bottom-Up Construction</u>	1
<u>Global Symbols</u>	1
<u>Destroy the User Interface</u>	2
<u>Cadence User Interface Callback Routines</u>	3
<u>Callbacks Supported by HI Forms and Fields</u>	5
<u>Function Object Callbacks</u>	7
<u>Function Object Callbacks with Additional Arguments</u>	8

2

<u>Command Interpreter Window</u>	13
<u>CIW Input Commands</u>	15
<u>Mouse and Keyboard Inputs</u>	15
<u>Input Prompt</u>	15
<u>Parse Command Line</u>	15
<u>Command Output Results</u>	15
<u>Error Recovery</u>	16
<u>Interrupts</u>	16
<u>Nesting Commands</u>	16
<u>General Environmental Functions</u>	17
<u>dplp</u>	18
<u>hiAddExtraRepeatCommand</u>	19
<u>hiAddNonRepeatPrefix</u>	20
<u>hiBoxCenter</u>	22
<u>hiDisplaySaveForRestoreDialog</u>	23
<u>hiEditfile</u>	24
<u>hiEnqueueCmd</u>	25
<u>hiFlush</u>	27
<u>hiFlushCIW</u>	28
<u>hiFlushInfo</u>	29

Cadence User Interface SKILL Reference

<u>hiFlushLogFile</u>	30
<u>hiFocusToCIW</u>	31
<u>hiGetAttention</u>	32
<u>hiGetDisplayName</u>	33
<u>hiGetMouseMoveSampleRate</u>	34
<u>hiGetMouseStopDetectTime</u>	35
<u>hiGetMultiClickTime</u>	36
<u>hiGetNonRepeatPrefixes</u>	37
<u>hiGetScreenSize</u>	38
<u>hiGetBannerPoint</u>	39
<u>hiGetBeepVolume</u>	40
<u>hiGetCommandPoint</u>	41
<u>hiGetCIWindow</u>	42
<u>hiGetExtraRepeatCommands</u>	43
<u>hiGetPoint</u>	44
<u>hiGetRepeatCommand</u>	45
<u>hiGraphicMode</u>	46
<u>hiInsertBlankCIWOutputPage</u>	47
<u>hiIsExtraRepeatCommand</u>	48
<u>hiQuit</u>	49
<u>hiRegTimer</u>	50
<u>hiRemoveExtraRepeatCommand</u>	51
<u>hiRemoveNonRepeatPrefix</u>	52
<u>hiRepeat</u>	53
<u>hiSaveForRestore</u>	54
<u>hiScaleBox</u>	55
<u>hiSetBeepVolume</u>	57
<u>hiSetMouseMoveSampleRate</u>	58
<u>hiSetMouseStopDetectTime</u>	59
<u>hiSetMultiClickTime</u>	60
<u>hiSetUserPreferences</u>	61
<u>hiSetWindowDefaultPrompt</u>	62
<u>hiSetWindowFocus</u>	63
<u>hiSynchronize</u>	64
<u>hiSync</u>	65
<u>Process Control Functions</u>	67

Cadence User Interface SKILL Reference

<u>hiCheckAbort</u>	68
<u>hiGetUserAbort</u>	69
<u>hiResetAbort</u>	70
<u>hiSetAbort</u>	71
<u>Log File Functions</u>	72
<u>hiEndLog</u>	73
<u>hiGetLogFileName</u>	74
<u>hiIsInReplay</u>	75
<u>hiLogDragEvents</u>	76
<u>hiPrintToLogFile</u>	77
<u>hiReplayFile</u>	79
<u>hiSetFilter</u>	81
<u>hiSetFilterOptions</u>	83
<u>hiStartLog</u>	85
<u>Resource Management Functions</u>	86
<u>hiGetBBoxResource</u>	87
<u>hiGetGeometryResource</u>	88
<u>hiGetStringResource</u>	90
<u>Text Attribute Functions</u>	91
<u>hiGetFont</u>	92
<u>hiGetFontInfo</u>	94
<u>hiGetXFontName</u>	96
<u>hiGetTextWidth</u>	97
<u>hiQueryFont</u>	99
<u>hiSetFont</u>	102
<u>hiTextWidth</u>	105
<u>Cdsenv Editor Functions</u>	107
<u>startCdsenvEditor</u>	108
<u>envCyclicStringToIndex</u>	109
<u>envCyclicIndexToString</u>	110
<u>envGetAvailableTools</u>	111
<u>envGetDefVal</u>	112
<u>envGetLoadedTools</u>	114
<u>envGetModifiedTools</u>	115
<u>envGetVal</u>	116
<u>envGetVarType</u>	117

Cadence User Interface SKILL Reference

<u>envIsToolModified</u>	118
<u>envIsVal</u>	119
<u>envLoadFile</u>	120
<u>envLoadVals</u>	121
<u>envRegLoadDumpTrigger</u>	123
<u>envRegSetTrigger</u>	125
<u>envSetToolCurrValToDefault</u>	126
<u>envSetToolDefaultToCurrVal</u>	127
<u>envSetVal</u>	128
<u>envSetVarCurrValToDefault</u>	129
<u>envSetVarDefaultToCurrVal</u>	130
<u>envStoreEnv</u>	131
<u>envUnregLoadDumpTrigger</u>	138
<u>envUnregSetTrigger</u>	140

3

<u>Menu Functions</u>	143
<u>hiAddMenuItem</u>	145
<u>hiCreate2DMenu</u>	147
<u>hiCreateHorizontalFixedMenu</u>	149
<u>hiCreateMenu</u>	152
<u>hiCreateMenuItem</u>	154
<u>hiCreatePulldownMenu</u>	158
<u>hiCreateSimpleMenu</u>	162
<u>hiCreateSeparatorMenuItem</u>	164
<u>hiCreateSliderMenuItem</u>	165
<u>hiCreateTypeinMenuItem</u>	167
<u>hiSetTypeinMenuItemCompleterList</u>	170
<u>hiCreateVerticalFixedMenu</u>	172
<u>hiDeleteMenu</u>	175
<u>hiDeleteMenuItem</u>	176
<u>hiDisableMenuItem</u>	177
<u>hiDisplayFixedMenu</u>	179
<u>hiDisplayMenu</u>	182
<u>hiDisplayWindowMenu</u>	184

Cadence User Interface SKILL Reference

<u>hiEnableMenuItem</u>	185
<u>hiFixedMenuDown</u>	187
<u>hiGetMenuItems</u>	188
<u>hiGetWindowMenu</u>	190
<u>hiInsertMenuItem</u>	191
<u>hils2DMenu</u>	193
<u>hilsIcon</u>	194
<u>hilsMenu</u>	195
<u>hilsMenuItemEnabled</u>	196
<u>hiSetMenuItemCallback</u>	198
<u>hiSetMenuItemStatusTip</u>	200
<u>hiSetMenuItemText</u>	202
<u>hiSetWindowMenu</u>	204

4

<u>Toolbar Functions</u>	207
<u>ddHiCreateCellToolbarComboBox</u>	209
<u>ddHiCreateLibraryToolbarComboBox</u>	212
<u>ddHiCreateViewToolbarComboBox</u>	215
<u>hiAddToolbarItem</u>	219
<u>hiAddToolbarItems</u>	221
<u>hiCreateAction</u>	223
<u>hiCreateToolbar</u>	227
<u>hiCreateToolbarComboBox</u>	229
<u>hiCreateToolbarTypein</u>	232
<u>hiCreateToolbarSeparator</u>	234
<u>hiDeleteToolbar</u>	235
<u>hiDeleteToolbarBreak</u>	237
<u>hiDeleteToolbarItem</u>	238
<u>hiDeleteToolbarItems</u>	239
<u>hiFocusToToolbarItem</u>	240
<u>hiGetSharedIcon</u>	242
<u>hiGetToolbarObjName</u>	243
<u>hiGetWindowToolbars</u>	244
<u>hiHasToolbarBreak</u>	245

Cadence User Interface SKILL Reference

<u>hiHideToolbar</u>	246
<u>hiInsertToolbar</u>	248
<u>hiInsertToolbarBreak</u>	250
<u>hiInsertToolbarItem</u>	251
<u>hiInsertToolbarItems</u>	253
<u>hiListFocusableToolbarItems</u>	255
<u>hiPlaceToolbar</u>	257
<u>hiPlaceToolbarBreak</u>	259
<u>hiReattachToolbar</u>	260
<u>hiShowToolbar</u>	262
<u>hiSetToolbarObjName</u>	264
<u>hiSetActionChecked</u>	266

5

<u>Dialog Boxes</u>	267
<u>Information Dialog Boxes</u>	267
<u>File Selection Dialog Boxes</u>	268
<u>Progress Dialog Boxes</u>	268
<u>General Dialog Box Functions</u>	269
<u>hiDisplayAppDBox</u>	270
<u>hiDisplayUserDBox</u>	278
<u>hiDBoxCancel</u>	283
<u>hiDBoxOK</u>	285
<u>hiSetDBoxDefaultLocation</u>	286
<u>hiGetDBoxDefaultLocation</u>	288
<u>File Selection Dialog Box Functions</u>	289
<u>hiDisplayFileDialog</u>	290
<u>hiFileDialogSelection</u>	297
<u>hiFileDialogSetSelection</u>	299
<u>hiFileDialogDone</u>	301
<u>hiGetExistingFile</u>	303
<u>hiGetExistingFiles</u>	307
<u>hiGetAnyFile</u>	311
<u>hiGetExistingDirectory</u>	315
<u>Progress Dialog Box Functions</u>	319

Cadence User Interface SKILL Reference

<u>hiAdvanceProgressBarOneStep</u>	320
<u>hiCancelProgressBox</u>	321
<u>hiDisplayProgressBox</u>	322
<u>hiGetProgress</u>	325
<u>hiGetProgressBarCurrentStep</u>	326
<u>hiGetProgressBarTotalSteps</u>	327
<u>hiGetProgressTotalSteps</u>	328
<u>hiHideProgressBar</u>	329
<u>hiIsProgressBoxCancelled</u>	330
<u>hiSetProgress</u>	331
<u>hiSetProgressText</u>	332
<u>hiSetProgressTotalSteps</u>	333
<u>hiSetProgressAndText</u>	334
<u>hiSetProgressButtonText</u>	335
<u>hiSetProgressBannerText</u>	336
<u>hiResetProgressBox</u>	337
<u>hiShowProgressBar</u>	338
<u>hiSetProgressBar</u>	340
<u>hiSetProgressLabel</u>	342
<u>Color Dialog Function</u>	343
<u>hiDisplayColorDialog</u>	344
<u>Old Dialog Box Functions</u>	346
<u>hiDisplayBlockingDBox</u>	347
<u>hiDisplayModalDBox</u>	348
<u>hiDisplayModelessDBox</u>	349
<u>hiDisplayNonBlockingDBox</u>	350

6

<u>List Box Functions</u>	351
<u>hiDisplayListBox</u>	352
<u>hiGetListBoxFieldFit</u>	354
<u>hiListBoxCancel</u>	356
<u>hiListBoxDone</u>	357
<u>hiShowListBox</u>	358

7

<u>Icon Functions</u>	365
<u>colorIndex</u>	366
<u>hiAddIconOverrides</u>	367
<u>hiCreateColorArray</u>	369
<u>hiGetIconOverrides</u>	370
<u>hiLoadIconFile</u>	372
<u>hiLoadImageFile</u>	375
<u>hiLoadIconData</u>	378
<u>hiMatchColor</u>	380
<u>hiMatchColorByName</u>	382
<u>hiRemoveIconOverrides</u>	384
<u>hiSaveIconData</u>	386
<u>hiStringToIcon</u>	388
<u>hiSetDisabledIcon</u>	390
<u>nameToColor</u>	391

8

<u>Form Functions</u>	393
<u>Form Layout</u>	395
<u>One Dimensional Form Layout</u>	395
<u>Two Dimensional Form Layout</u>	396
<u>hiLayout Form Functions</u>	403
<u>hiAddItemToLayout</u>	406
<u>hiAddField</u>	409
<u>hiAddFields</u>	411
<u>hiCreateFormLayout</u>	413
<u>hiCreateGridLayout</u>	417
<u>hiCreateHorizontalBoxLayout</u>	423
<u>hiCreateStackedLayout</u>	428
<u>hiCreateVerticalBoxLayout</u>	431
<u>hiCreateLayoutForm</u>	435
<u>hiDeleteField</u>	444
<u>hiDeleteFields</u>	446

Cadence User Interface SKILL Reference

<u>hiDeleteItemFromLayout</u>	448
<u>Traversing Fields</u>	449
<u>Creating Context Menus for Fields</u>	450
<u>Setting the hiContextMenu Property</u>	450
<u>Setting the hiShowContextMenuCallback Property</u>	450
<u>Creating Tool Tips for Fields</u>	452
<u>Form and Field Functions</u>	454
<u>ddHiCreateCellComboField</u>	456
<u>ddHiCreateLibraryComboField</u>	460
<u>ddHiCreateViewComboField</u>	464
<u>ddoAccepts</u>	469
<u>ddoPreferred</u>	470
<u>ddHiLinkFields</u>	472
<u>hiAddCyclicChoice</u>	475
<u>hiChangeFormCallback</u>	477
<u>hiChangeFormTitle</u>	478
<u>hiCollapseTreeItem</u>	479
<u>hiCreateAppForm</u>	480
<u>hiCreateBBoxField</u>	497
<u>hiCreateBooleanButton</u>	501
<u>hiCreateButton</u>	506
<u>hiCreateButtonBoxField</u>	509
<u>hiCreateComboField</u>	511
<u>hiCreateCyclicField</u>	517
<u>hiCreateFileSelectorField</u>	522
<u>hiCreateDisclosureButton</u>	529
<u>hiCreateDisclosureTriangle</u>	532
<u>hiCreateFloatField</u>	537
<u>hiCreateFrameField</u>	541
<u>hiCreateScrollRegion</u>	543
<u>hillsScrollRegion</u>	548
<u>hiCreateForm</u>	549
<u>hiCreateFormButton</u>	550
<u>hiCreateFormLabel</u>	552
<u>hiCreateHypertextField</u>	554
<u>hiCreateIntField</u>	560

Cadence User Interface SKILL Reference

<u>hiCreateLabel</u>	564
<u>hiCreateLayerCyclicField</u>	567
<u>hiCreateListField</u>	569
<u>hiCreateListBoxField</u>	573
<u>hiCreateMLTextField</u>	583
<u>hiCreateOptionsForm</u>	588
<u>hiCreateOutputStringField</u>	591
<u>hiCreatePointField</u>	593
<u>hiCreatePointListField</u>	597
<u>hiCreateRadioField</u>	601
<u>hiCreateReportField</u>	604
<u>hiCreateScaleField</u>	618
<u>hiCreateSeparatorField</u>	621
<u>hiCreateSimpleHypertextField</u>	623
<u>hiCreateSpinBox</u>	625
<u>hiCreateFloatSpinBox</u>	629
<u>hiCreateStringField</u>	633
<u>hiCreateTabField</u>	640
<u>hiCreateTree</u>	653
<u>hiCreateTreeltem</u>	655
<u>hiCreateTreeTable</u>	658
<u>hiCreateToggleField</u>	674
<u>hiDeleteForm</u>	678
<u>hiDisplayForm</u>	679
<u>hiEditPropList</u>	682
<u>hiEscapeStringChars</u>	685
<u>hiEscapeHTMLTags</u>	686
<u>hiExpandTreeltem</u>	687
<u>hiFormApply</u>	688
<u>hiFormCancel</u>	689
<u>hiFormClose</u>	690
<u>hiFormDefaults</u>	691
<u>hiFormDone</u>	692
<u>hiFormFinish</u>	694
<u>hiFormList</u>	696
<u>hiFormUnmap</u>	697

Cadence User Interface SKILL Reference

<u>hiGetButtonAreaHeight</u>	699
<u>hiGetChoiceStrings</u>	700
<u>hiGetCurrentField</u>	702
<u>hiGetCurrentForm</u>	703
<u>hiGetCyclicValueString</u>	704
<u>hiGetFieldAttachments</u>	706
<u>hiGetFieldInfo</u>	708
<u>hiGetFieldMinSize</u>	710
<u>hiGetRadioValueString</u>	712
<u>hiLayoutUTp</u>	714
<u>hiGetFieldOverlaps</u>	715
<u>hiGetFieldScope</u>	716
<u>hiGetFormLocation</u>	717
<u>hiGetFormSize</u>	718
<u>hiGetInsertionPosition</u>	719
<u>hiGetItemOfLayout</u>	720
<u>hiGetLayerCyclicValue</u>	721
<u>hiGetLayoutChildLayouts</u>	722
<u>hiGetLayoutFields</u>	723
<u>hiGetLayoutItems</u>	725
<u>hiGetListBoxValue</u>	726
<u>hiGetLayoutFrame</u>	727
<u>hiGetLayoutItemIndex</u>	729
<u>hiGetLayoutType</u>	730
<u>hiGetStackedLayoutCurrentIndex</u>	732
<u>hiSetFieldMinSize</u>	734
<u>hiSetKeyboardFocusField</u>	736
<u>hiSetLayoutFrameTitle</u>	738
<u>hiSetLayoutMargins</u>	740
<u>hiGetGridLayoutItemLocation</u>	742
<u>hiGetKeyboardFocusField</u>	744
<u>hiGetLayoutMargins</u>	745
<u>hiGetNumVisibleItems</u>	747
<u>hiGetScrollBarInfo</u>	748
<u>hiGetTextFieldFit</u>	750
<u>hiGetTopListItem</u>	751

Cadence User Interface SKILL Reference

<u>hiGetTree</u>	752
<u>hiGetTreeItems</u>	753
<u>hiGetTreeItemDescription</u>	754
<u>hiGetTreeItemIcons</u>	755
<u>hiGetTreeItemParent</u>	756
<u>hiGetTreeParent</u>	757
<u>hiHighlightField</u>	758
<u>hiHypertextReadin</u>	760
<u>hiHypertextReplace</u>	761
<u>hiHypertextScroll</u>	762
<u>hiIgnoreProp</u>	764
<u>hiInFormApply</u>	765
<u>hiInsertItemToLayout</u>	766
<u>hiInstantiateForm</u>	769
<u>hilsForm</u>	770
<u>hilsFormDisplayed</u>	771
<u>hilsInstantiated</u>	773
<u>hilsInFieldCancel</u>	775
<u>hiItemInsertTree</u>	776
<u>hiLayerMatchCyclicStr</u>	777
<u>hiLaunchBrowser</u>	778
<u>hiLayerStringToLPP</u>	780
<u>hiMoveField</u>	781
<u>hiMoveToFormField</u>	782
<u>hiMoveInsBarToEnd</u>	784
<u>hiOffsetField</u>	785
<u>hiOffsetFields</u>	787
<u>hiResizeField</u>	789
<u>hiReattachField</u>	791
<u>hiReportSelectItem</u>	793
<u>hiReportSelectItems</u>	795
<u>hiReportSelectAllItems</u>	797
<u>hiReportDeselectItem</u>	799
<u>hiReportDeselectItems</u>	801
<u>hiReportDeselectAllItems</u>	803
<u>hiReportGetSelectedItems</u>	805

Cadence User Interface SKILL Reference

<u>hiSetButtonIcon</u>	807
<u>hiSetButtonLabel</u>	808
<u>hiSetCallbackStatus</u>	809
<u>hiSetCurrentField</u>	811
<u>hiSetCurrentForm</u>	813
<u>hiSetFieldEditable</u>	815
<u>hiSetFieldEnabled</u>	816
<u>hiSetFormButtonEnabled</u>	817
<u>hiSetFormBlock</u>	818
<u>hiSetFormHighlights</u>	820
<u>hiSetFormMinMaxSize</u>	821
<u>hiSetFormName</u>	822
<u>hiSetFormPosition</u>	823
<u>hiSetFormSize</u>	824
<u>hiSetFormToDefaults</u>	825
<u>hiSetInsertionPosition</u>	826
<u>hiSetLayerCyclicValue</u>	827
<u>hiSetStackedLayoutCurrentIndex</u>	828
<u>hiSetScrollBarValue</u>	829
<u>hiSetTopListItem</u>	831
<u>hiSetTreeItemDescription</u>	832
<u>hiSetTreeItemIcons</u>	833
<u>hiSetListItemVisible</u>	834
<u>hiSetListItemCenter</u>	835
<u>hiShowFieldBorders</u>	836
<u>hiStoreFormLocation</u>	837
<u>hiSwapTreeItemSelectionIcons</u>	838
<u>hiTreeAppendItem</u>	839
<u>hiTreeAppendItems</u>	840
<u>hiTreePrependItem</u>	842
<u>hiTreePrependItems</u>	843
<u>hiTreeRemoveItem</u>	844
<u>hiTreeRemoveItems</u>	846
<u>hiTreeRemoveAllItems</u>	848
<u>hiTreeTableDeselectItem</u>	849
<u>hiTreeTableDeselectAllItems</u>	851

Cadence User Interface SKILL Reference

<u>hiTreeTableDeselectItems</u>	852
<u>hiTreeTableGetExpandedItems</u>	854
<u>hiTreeTableGetExpandedItemCount</u>	855
<u>hiTreeTableGetItems</u>	856
<u>hiTreeTableGetItemCount</u>	857
<u>hiTreeTableGetSelectedItems</u>	858
<u>hiTreeTableGetSelectedItemCount</u>	859
<u>hiTreeTableSelectItem</u>	860
<u>hiTreeTableSelectAllItems</u>	862
<u>hiTreeTableSelectItems</u>	864
<u>hiItemRemoveTree</u>	866
<u>hiUpdateFormBlock</u>	867

9

<u>Window Functions</u>	869
<u>dwindow</u>	871
<u>dwindowp</u>	872
<u>getCurrentWindow</u>	873
<u>getMaxScreenCoords</u>	874
<u>hiAddFixedMenu</u>	875
<u>hiCloseWindow</u>	877
<u>hiCreateDockWindow</u>	879
<u>hiCreateWindow</u>	884
<u>hiDeiconifyWindow</u>	889
<u>hiDisplayWindow</u>	890
<u>hiDockWindow</u>	892
<u>hiFocusToCursor</u>	895
<u>hiGetAbsWindowScreenBBox</u>	896
<u>hiGetActiveTabWindow</u>	900
<u>hiGetAppType</u>	901
<u>hiGetCurrentEnterFunPoints</u>	902
<u>hiGetCurrentWindow</u>	904
<u>hiGetCursor</u>	905
<u>hiGetCursorByName</u>	906
<u>hiGetDbuPoint</u>	908

Cadence User Interface SKILL Reference

<u>hiGetDrawThruDelta</u>	909
<u>hiGetEFDrawThruDelta</u>	911
<u>hiGetEFunCursor</u>	912
<u>hiGetHelp</u>	913
<u>hiGetIconName</u>	914
<u>hiGetScreenPoint</u>	915
<u>hiGetSessionWindow</u>	916
<u>hiGetWindowState</u>	917
<u>hiGetWindowDisplayState</u>	919
<u>hiGetWindowIconifyState</u>	921
<u>hiGetWindowList</u>	923
<u>hiGetWindowName</u>	925
<u>hiOpenWindow</u>	926
<u>hiUndockWindow</u>	932
<u>hiIsWindowSpecifier</u>	933
<u>hiSetDrawThruDelta</u>	934
<u>hiSetSessionFocus</u>	936
<u>hiSetTabWindowTitle</u>	937
<u>hiSetTabWindowIcon</u>	938
<u>hiSetTabWindowTip</u>	939
<u>hiSetWindowAtts</u>	940
<u>hiSetWinStyle</u>	942
<u>hiRegCloseProc</u>	943
<u>hiUnregCloseProc</u>	945
<u>hiSwitchWindowType</u>	946
<u>hiIconifyWindow</u>	948
<u>hiLowerWindow</u>	949
<u>hiRaiseWindow</u>	950
<u>hiRaiseTab</u>	952
<u>hiLowerTab</u>	953
<u>hiPickWindow</u>	955
<u>hiMapWindow</u>	956
<u>hiUnmapWindow</u>	958
<u>hiMoveWindow</u>	960
<u>hiResizeWindow</u>	962
<u>hiSetWindowName</u>	964

Cadence User Interface SKILL Reference

<u>hiSetIconName</u>	965
<u>hiSetWindowIcon</u>	966
<u>hiGetWidgetType</u>	967
<u>hiIsWidgetType</u>	968
<u>hiSetCurrentWindow</u>	969
<u>hiRegCurWindowTrigger</u>	971
<u>hiUnregCurWindowTrigger</u>	972
<u>hiGetMaxScreenCoords</u>	973
<u>hiGetWMOffsets</u>	975
<u>hiRemoveFixedMenu</u>	977
<u>hiMoveFixedMenu</u>	979
<u>hiGetWindowFixedMenu</u>	981
<u>hiSetCursor</u>	982
<u>hiGetCursorName</u>	990
<u>hiSetShadowMode</u>	992
<u>hiRemovePromptLine</u>	993
<u>hiSetEFunCursor</u>	994
<u>hiSetEFDrawThruDelta</u>	995
<u>setCurrentWindow</u>	997
<u>window</u>	998
<u>swindow</u>	1000
<u>swindowp</u>	1001
<u>windowp</u>	1002
<u>wtypep</u>	1003
<u>Viewing Functions</u>	1004
<u>hiZoomIn</u>	1005
<u>hiZoomOut</u>	1007
<u>hiZoomRelativeScale</u>	1009
<u>hiZoomAbsoluteScale</u>	1011
<u>hiZoomWindowAtPoint</u>	1013
<u>hiZoomWindowAtMouse</u>	1015
<u>hiZoomInAtMouse</u>	1017
<u>hiZoomOutAtMouse</u>	1019
<u>hiZoomExpandAtPoint</u>	1020
<u>hiZoomExpandAtMouse</u>	1021
<u>hiGetViewBBox</u>	1022

Cadence User Interface SKILL Reference

<u>hiPan</u>	1023
<u>hiVectorPan</u>	1025
<u>hiDeltaPan</u>	1027
<u>hiAbsolutePan</u>	1029
<u>hiDynamicPanAbsolute</u>	1031
<u>hiDynamicPanRelative</u>	1033
<u>hiDynamicPanInteractive</u>	1035
<u>hiDynamicPanGrabbing</u>	1037
<u>hiRedraw</u>	1039
<u>hiRegZoomPanProc</u>	1040
<u>hiUnregZoomPanProc</u>	1042
<u>hiSaveView</u>	1043
<u>hiListView</u>	1044
<u>hiRestoreView</u>	1045
<u>hiPrevWinView</u>	1046
<u>hiNextWinView</u>	1047
<u>hiGetUndoLimit</u>	1048
<u>hiSetUndoLimit</u>	1049
<u>hiUndo</u>	1050
<u>hiRedo</u>	1051
<u>hiEnableBlink</u>	1052
<u>hilsBlinkEnabled</u>	1053
<u>World View Functions</u>	1054
<u>hiCreateWorldView</u>	1055
<u>hiWorldViewFit</u>	1056
<u>hiWorldViewRedraw</u>	1057
<u>hiWorldViewZoomIn</u>	1058
<u>hiWorldViewZoomOut</u>	1059
<u>Thumbnail View Functions</u>	1060
<u>hiGenerateThumbnails</u>	1061
<u>hiGetDefaultThumbnail</u>	1065
<u>hiGetDefaultThumbnailResolution</u>	1066
<u>hiGetDefaultWidgetSpacing</u>	1067
<u>hiGetThumbnailFilename</u>	1069
<u>Workspace Functions</u>	1071
<u>hiGetWinConfig</u>	1073

Cadence User Interface SKILL Reference

<u>hiSaveWinConfig</u>	1075
<u>hiReadWinConfig</u>	1077
<u>hiApplyWinConfig</u>	1078
<u>hiWinConfigModified</u>	1080
<u>hiGetWinConfigInfo</u>	1082
<u>hiSetWinObjName</u>	1084
<u>hiGetWinObjName</u>	1086
<u>Screen Capture Functions</u>	1088
<u>hiWindowSaveImage</u>	1089
<u>hiExportImage</u>	1093
<u>hiExportImageDialog</u>	1101
<u>Splash Screen Functions</u>	1103
<u>hiAbout</u>	1104
<u>hiSetSplashBackground</u>	1105
<u>hiSetSplashDefaultBackground</u>	1106
<u>hiSetSplashIcon</u>	1107
<u>hiSetSplashFamily</u>	1108
<u>hiSetSplashProduct</u>	1109
<u>hiSetSplashLicense</u>	1110
<u>Magnifier Functions</u>	1111
<u>hiGetMagnifierSize</u>	1112
<u>hiIsMagnifierAnchored</u>	1113
<u>hiIsMagnifierOpened</u>	1114
<u>hiResizeMagnifier</u>	1115
<u>hiScaleMagnifier</u>	1116
<u>hiToggleAnchorMagnifier</u>	1117
<u>hiToggleMagnifier</u>	1118
<u>hiToggleMagnifierResolution</u>	1119
<u>hiUpdateMagOptions</u>	1120
<u>hiUseMagnifierResolution</u>	1121
<u>hiUsingMagnifierResolution</u>	1123
<u>hiZoomInMagnifier</u>	1124
<u>hiZoomOutMagnifier</u>	1125

10

<u>Window Banner Functions</u>	1127
<u>hiChangeBannerLabel</u>	1128
<u>hiDeleteBannerLabel</u>	1130
<u>hiDeleteBannerMenu</u>	1131
<u>hiDeleteBannerMenus</u>	1132
<u>hiGetBannerLabels</u>	1133
<u>hiGetBannerMenus</u>	1134
<u>hiGetNumMenus</u>	1135
<u>hiInsertBannerMenu</u>	1136
<u>hiIsMenuSlotFilled</u>	1138
<u>hiReplaceAllBannerMenus</u>	1139

11

<u>Bindkey Functions</u>	1141
<u>hiBindKeyModifiers</u>	1143
<u>hiSetBindKey</u>	1146
<u>hiSetBindKeys</u>	1151
<u>hiRegSetBindKeyProc</u>	1154
<u>hiGetBindKey</u>	1156
<u>hiGetBindKeys</u>	1158
<u>hiShowBindKeys</u>	1162
<u>hiShowBindKeysByAppType</u>	1164
<u>hiShowBindKeysByWindow</u>	1166
<u>hiRegisterBindKeyPrefix</u>	1167
<u>hiInheritBindKey</u>	1168
<u>hiGetBindKeyInheritRoot</u>	1169
<u>hiGetBindKeyInheritAlias</u>	1170
<u>hiGetBindKeyPrefixList</u>	1171
<u>hiConfigureBindKeys</u>	1172
<u>hiEnableAsstBindKeyPrefix</u>	1174
<u>hiDisableAsstBindKeyPrefix</u>	1175
<u>hiGetDisabledAsstBindKeyPrefixes</u>	1176
<u>hiPrintBindKeyUsage</u>	1177

12

User Entry Functions	1179
<u>Enterfunction Prompts</u>	1181
<u>Terminating an Enterfunction</u>	1181
<u>Callback Procedures</u>	1183
<u>Callback Procedure addPointProc</u>	1183
<u>Callback Procedure delPointProc</u>	1183
<u>Callback Procedure formProc</u>	1184
<u>Callback Procedure doneProc</u>	1184
<u>Enterfunction Flags</u>	1185
<u>Enterfunction Flag dontDraw</u>	1185
<u>Enterfunction Flag alwaysMap</u>	1185
<u>Enterfunction Flag acceptString</u>	1185
<u>Enterfunction Flag acceptNumber</u>	1185
<u>Enterfunction Flag noInfix</u>	1186
<u>Nesting Enterfunctions</u>	1187
<u>addPoint</u>	1189
<u>applyEnterFun</u>	1191
<u>cancelEnterFun</u>	1193
<u>changeEnterFun</u>	1195
<u>changeNextEnterFun</u>	1199
<u>clearAllEnterFunctions</u>	1200
<u>deletePoint</u>	1201
<u>drawEnterFun</u>	1203
<u>enterArc</u>	1204
<u>enterBox</u>	1208
<u>enterCircle</u>	1212
<u>enterDonut</u>	1216
<u>enterEllipse</u>	1220
<u>enterFunNestLevel</u>	1224
<u>enterLine</u>	1225
<u>enterNumber</u>	1229
<u>enterPath</u>	1231
<u>enterPoint</u>	1236
<u>enterPoints</u>	1239

Cadence User Interface SKILL Reference

<u>enterPolygon</u>	1243
<u>enterScreenBox</u>	1247
<u>enterSegment</u>	1249
<u>enterMultiRep</u>	1253
<u>enterString</u>	1258
<u>finishEnterFun</u>	1260
<u>hiGetCurrentCmd</u>	1262
<u>hiInEnterFun</u>	1263
<u>hiMarkNestable</u>	1264
<u>hiMarkNonNestable</u>	1265
<u>hiToggleEnterForm</u>	1266
<u>hiUpdate</u>	1267
<u>preXY</u>	1268
<u>setEFunFlightLineTarget</u>	1269
<u>undrawEnterFun</u>	1270

13

Encapsulation Window 1271

<u>hiAppendInputCmd</u>	1273
<u>hiEncap</u>	1275
<u>hiFocusToEncap</u>	1277
<u>hiGetEncapSkillCmd</u>	1278
<u>hiSetEncapHistory</u>	1279
<u>hiSetEncapPrompt</u>	1281
<u>hiSetEncapSkillCmd</u>	1282

14

Functions of Viewfile Window 1285

<u>Viewfile Management</u>	1287
<u>HTML Viewfile Window</u>	1287
<u>Text Device Mode</u>	1287
<u>Hypertext Viewfile Window</u>	1288
<u>hiClearClipboard</u>	1290
<u>hiClearCurrentTextSelection</u>	1292
<u>hiDisableTailViewfile</u>	1293

Cadence User Interface SKILL Reference

<u>hiEnableTailViewfile</u>	1294
<u>hiGenTextIndex</u>	1295
<u>hiViewfile</u>	1296
<u>hiViewTextFile</u>	1297
<u>hiSetViewfile</u>	1299
<u>hiSaveViewfile</u>	1300
<u>hiSaveAsViewfile</u>	1301
<u>hiStartGenTextIndex</u>	1302
<u>hiGetTextSelection</u>	1303
<u>hiGetTextSelByLoc</u>	1304
<u>hiSetTextSelection</u>	1305
<u>hiSelectTextByLoc</u>	1306
<u>hiGetCurrentTextSelection</u>	1307
<u>hiGetCurrentTextSelectionLoc</u>	1308
<u>hiSetCurrentTextSelectionLoc</u>	1309
<u>hiSetTextSelectAll</u>	1310
<u>hiUnselectText</u>	1311
<u>hiUnselectTextByLoc</u>	1312
<u>hiUnselectTextClass</u>	1313
<u>hiUnselectTextAll</u>	1314
<u>hiGoToLine</u>	1315
<u>hiGoToIndex</u>	1317
<u>hiScrollWindowLeft</u>	1319
<u>hiScrollWindowRight</u>	1320
<u>hiScrollWindowUp</u>	1321
<u>hiScrollWindowDown</u>	1322
<u>hiScrollWindowTop</u>	1323
<u>hiScrollWindowBottom</u>	1324
<u>hiScrollWindowToCurrentIndex</u>	1325
<u>hiScrollWindowToIndex</u>	1326
<u>hiScrollWindowToName</u>	1327
<u>hiGetCurrentIndex</u>	1329
<u>hiSetCurrentIndex</u>	1330
<u>hiGetClipboard</u>	1331
<u>hiSetClipboard</u>	1333
<u>hiGetTextClass</u>	1335

Cadence User Interface SKILL Reference

<u>hiSetTextClass</u>	1336
<u>view</u>	1337
<u>Text Customization Functions</u>	1339
<u>hiSetTextHighlightColor</u>	1340
<u>hiTextDisplayString</u>	1342
<u>hiRefreshTextWindow</u>	1344
<u>hiUpdateTextSelectionColors</u>	1345
<u>Word Delimiters</u>	1346
<u>hiGetTextWordDelimiter</u>	1347
<u>hiAddTextWordDelimiter</u>	1348
<u>hiRemoveTextWordDelimiter</u>	1349
<u>hiReplaceTextWordDelimiter</u>	1350
<u>Miscellaneous Viewfile Functions</u>	1351
<u>hiGetTextCharAtLoc</u>	1352
<u>hiGetTextSourceLength</u>	1353
<u>hiGetTextLineColumn</u>	1354
<u>hiGetTextIndexLoc</u>	1355
<u>hiGetTextDispLoc</u>	1356

15

<u>Online Help Mechanism</u>	1357
<u>Form and Dialog Box Help</u>	1360
<u>Window Help</u>	1361
<u>Menu Help</u>	1362
<u>Properties Required for Task Assistant</u>	1363
<u>Specify Your Own Callback for Help</u>	1365
<u>Setting the hiHelpAction Property on a Form or Dialog Box</u>	1365
<u>Writing a Callback Function for a Form or Dialog Box</u>	1365
<u>Specifying Your Own Callback for Windows</u>	1366
<u>Online Help Functions</u>	1368
<u>hiHelp</u>	1369
<u>hiLookupHRFtag</u>	1372
<u>hiModalDBoxHelp</u>	1374
<u>hiModelessDBoxHelp</u>	1375
<u>hiObShowTag</u>	1376

<u>Debugging Help</u>	1377

16

<u>QtReplay Functions</u>	1379

<u>nrDialogAccept</u>	1380
<u>nrGetIndexesFromSelection</u>	1381
<u>nrGetSelectedIndexes</u>	1383
<u>nrListColumnMove</u>	1385
<u>nrListColumnSort</u>	1386
<u>nrListItemAction</u>	1387
<u>nrListItemExpand</u>	1388
<u>nrListItemRename</u>	1390
<u>nrResizeWindow</u>	1391
<u>nrSetCurrentWindow</u>	1392
<u>nrSetFileFilter</u>	1393
<u>nrSetWindowVisibility</u>	1394
<u>nrUpdateMenu</u>	1395

17

<u>Performance Diagnostic Functions</u>	1397

<u>perfDiagInstall</u>	1398
<u>perfRefreshControl</u>	1400
<u>perfStartCallStackProc</u>	1401
<u>perfStopCallStackProc</u>	1402

18

<u>Display Lists</u>	1403

<u>Add Objects to the Display List</u>	1405

<u>dlAddArc</u>	1407
<u>dlAddBox</u>	1408
<u>dlAddCircle</u>	1409
<u>dlAddDonut</u>	1410
<u>dlAddEventObject</u>	1411
<u>dlAddPath</u>	1413

Cadence User Interface SKILL Reference

<u>dlAddPoint</u>	1414
<u>dlAddPolygon</u>	1415
<u>dlAddRasterText</u>	1416
<u>dlAddSegment</u>	1418
<u>dlAddSkillObject</u>	1419
<u>dlAddStrokeText</u>	1420
<u>dlMakeDisplayList</u>	1422
<u>dlSetClearOnDraw</u>	1424
<u>Display List Draw Functions</u>	1425
<u>dlDrawArc</u>	1427
<u>dlDrawBox</u>	1428
<u>dlDrawCircle</u>	1429
<u>dlDrawDonut</u>	1430
<u>dlDrawPath</u>	1431
<u>dlDrawPoint</u>	1432
<u>dlDrawPolygon</u>	1433
<u>dlDrawRasterText</u>	1434
<u>dlDrawSegment</u>	1436
<u>dlDrawStrokeText</u>	1437
<u>Display List Pen Functions</u>	1438
<u>dlMakePenTable</u>	1440
<u>dlMakeStipple</u>	1441
<u>dlSetCurrentPen</u>	1442
<u>dlSetPenColor</u>	1443
<u>dlSetPenFillStyle</u>	1444
<u>dlSetPenFilled</u>	1445
<u>dlSetPenStipple</u>	1446
<u>dlSetPenTable</u>	1447
<u>View Display List Functions</u>	1448
<u>dlAttachDlistToWidget</u>	1449
<u>dlAttachDlistToWindow</u>	1450
<u>dlClearDisplayList</u>	1451
<u>dlCloseWidget</u>	1452
<u>dlConfigureButton</u>	1453
<u>dlDetachDlistFromWidget</u>	1454
<u>dlDetachDlistFromWindow</u>	1455

Cadence User Interface SKILL Reference

<u>dlDisplay</u>	1456
<u>dlDlistTolcon</u>	1457
<u>dlEnableItem</u>	1458
<u>dlGetRasterTextWidth</u>	1459
<u>dlFitDlistOnDraw</u>	1461
<u>dlMakeDlistButton</u>	1462
<u>dlMakeWidget</u>	1463
<u>dlMapWidget</u>	1464
<u>dlMoveButton</u>	1465
<u>dlQueryRasterFont</u>	1466
<u>dlResizeButton</u>	1468
<u>dlSaveDlist</u>	1469
<u>dlSetDlistPosition</u>	1470
<u>dlSetDlistScale</u>	1471
<u>dlSetWidgetName</u>	1472
<u>dlUnMapWidget</u>	1473

19

<u>Graph Browser</u>	1475
<u>Terms Used in Cadence User Interface</u>	1477
<u>Properties of the dagArc</u>	1479
<u>Properties of the dagClass</u>	1480
<u>Properties of the dagNode</u>	1482
<u>Properties of the dagTool</u>	1485
<u>Actions and Action Lists</u>	1489
<u>dag Functions</u>	1492
<u>dagAddActionToObject</u>	1493
<u>dagCreateClass</u>	1494
<u>dagCreateNode</u>	1495
<u>dagDeleteActionFromObject</u>	1496
<u>dagDestroyNode</u>	1497
<u>dagDisplayTool</u>	1498
<u>dagGetCurrentObject</u>	1499
<u>dagGetCurrentTool</u>	1501
<u>dagLinkParentToChild</u>	1502

Cadence User Interface SKILL Reference

<u>dagNumToTool</u>	1503
<u>dagOpenTool</u>	1504
<u>dagPopTool</u>	1506
<u>dagPushTool</u>	1507
<u>dagRefreshObject</u>	1508
<u>dagSetActionStatus</u>	1509
<u>dagSetCurrentTool</u>	1510
<u>dagSetExpandedActionStatus</u>	1511
<u>dagSetFont</u>	1512
<u>dagUnlinkParentFromChild</u>	1513

20

<u>Interprocess Communication</u>	1517
<u>hiSleep</u>	1517

21

<u>Stroke Functions</u>	1519
<u>hiGetStrokeBBox</u>	1520
<u>hiGetStrokeFirstPt</u>	1521
<u>hiGetStrokeLastPt</u>	1522
<u>hiStroke</u>	1523

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

At system start-up, the look and feel of the system is determined by a configurable system start-up file. Once the system is running, changing the look and feel of the system, creating new objects, and object manipulation is possible through SKILL and C user interface access routines. The goal of these interface routines is to provide layered, controlled access to the user interface and its objects.

Access is available through high-level objects, composed of primitive user interface elements: menus, scroll bars, popup menus, labels, command buttons, or through primitive objects themselves. Several layers of start up configuration files are supplied. You can customize the functionality at each level.

Bottom-Up Construction

Larger, composite user-interface objects must follow bottom-up construction; that is, each sub-object that makes up a composite object must be created before the composite object is created. A menu is an example of a composite object. Each menu item must be created before the call to `hiCreateMenu`, which accepts menu item objects as arguments.

Global Symbols

Most of the user interface objects you create, such as forms, menus, list boxes, non-blocking dialog boxes, are used and accessed in more than one SKILL routine. For this reason it is necessary to use a unique global SKILL symbol as the SKILL handle in the creation routines. The user interface code depends on this.

For example, when a form is created and the SKILL handle passed to `hiCreateAppForm` or `hiCreateForm` is not global, any attempts to close this form (via the *OK* or *Cancel* button) fails. This is because the user interface code cannot access the form contents if the form symbol is not global.

When defining names for forms, fields, or other user interface objects, do not use any characters other than alphanumerics, underscore, and question mark. Do not begin the name with a numeric character or a question mark.

Destroy the User Interface

It is recommended to avoid destroying a user- interface object by resetting its SKILL handle to `nil`. This destroys the SKILL contents of the object, but leaves the memory associated with the object unreferenced, and impossible to free. For menus and forms, use the appropriate delete function, such as `hiDeleteMenu` or `hiDeleteForm`. List boxes and dialog boxes gets destroyed when they are removed from the screen.

Related Topics

[Cadence User Interface Callback Routines](#)

[Callbacks Supported by HI Forms and Fields](#)

[Function Object Callbacks](#)

[hiCreateMenu](#)

[hiCreateAppForm](#)

[hiCreateForm](#)

Cadence User Interface Callback Routines

User interface objects can have callback routines associated with them. A callback is a SKILL routine that is asynchronously called back if some triggering user action takes place. Examples of such actions would be values being changed on forms, menu item selection, or dialog box confirmation. Typically, callbacks analyze and process data that you provide. In the case of forms, for example, a callback can be specified for each form field or for the form itself.

SKILL callbacks can be strings, symbols, or function objects. In any case, the string, the function specified by its symbol, or the function object, is evaluated in SKILL at the time the callback is triggered.

If the callback is:

- a string, it must be a complete, valid SKILL expression because the string is passed directly to SKILL for evaluation.
- a symbol, it is assumed that the value of this symbol is a function, and a predetermined set of arguments is passed to that function.
- a function object, a predetermined set of arguments are passed to that function.

A function object callback offers the following advantages over the string and symbol types:

- String and symbol callbacks are required to be in the global namespace. However, you need not define a function object type callback in the global namespace and if you define it in SKILL++, it is evaluated in the same environment in which it is defined.
- When string and symbol callbacks are used, to access a field, you have to know where that field is used at the callback creation time. However, for a function object, you do not need to know the location of the field.

The functionality of callback routines follow the following guidelines:

- They cannot return data on a stack. Instead, callback expressions return a value:

```
someSymbol = hiSomeRoutine(y)
```

In this example, `someSymbol` is a global SKILL symbol to be accessed later.

- Make callbacks concise. In particular, avoid corrupting program data. As an example, the results can be unpredictable if a form callback modifies the form itself. Callbacks with destructive actions such as form modification, should be associated with actions that cannot be reversed, and cannot be done programmatically; that is, do not have a destructive action associated with a toggle button press (the toggle button has two states). This process can be simulated programmatically, doing so may unintentionally

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

execute the destructive callback multiple times. In this case, the callback should be associated with a form button, which is an irreversible, non-programmatic action.

- Avoid recursion in programming callbacks. This is a common mistake, and is easily demonstrated by a field callback which again sets the value of the field itself. Since callbacks are called when the value of the field changes, this would recursively continue calling the field callback, which sets the field value, which calls the field's callback, which sets the field's value, and so on.

If an error is detected in a callback routine, a dialog box should be brought up to flag the error, or an error logged in the CIW output panel. The field (type-in fields only) can be highlighted as well.

Related Topics

[Callbacks Supported by HI Forms and Fields](#)

[Function Object Callbacks](#)

Callbacks Supported by HI Forms and Fields

In HI forms, you can define callbacks for the *OK* and *Cancel* buttons, customized callbacks in `?buttonLayout`, and `mapCallback`. In HI fields, the most common callback is the common value change callback (`?callback`). However, there are 22 different types of HI fields and you can define callbacks to handle different events for these fields. For example, for the type-in fields you can define the `?modifyCallback` and `?focusInCallback`. For report and tree fields, you can define callbacks to handle pointer events.

The following table lists the various callbacks supported by HI forms and fields:

HI Form/Field Name	Callback Name
<u><code>hiCreateAppForm</code></u>	<code>?callback</code> <code>?mapCB</code>
<u><code>hiChangeFormCallback</code></u>	<code>?callback</code>
<u><code>hiCreateStringField</code></u> Supports the same callbacks as <code>hiCreateStringField</code>	<code>?callback</code> <code>?modifyCallback</code> <code>?focusInCallback</code>
<u><code>hiCreateIntField</code></u>	<code>?callback</code> <code>?focusInCallback</code>
<u><code>hiCreateFloatField</code></u>	<code>?callback</code> <code>?focusInCallback</code>
<u><code>hiCreateListField</code></u>	<code>?callback</code> <code>?focusInCallback</code>
<u><code>hiCreatePointField</code></u>	<code>?callback</code> <code>?focusInCallback</code>
<u><code>hiCreateBBoxField</code></u>	<code>?callback</code> <code>?focusInCallback</code>
<u><code>hiCreateComboField</code></u>	<code>?callback</code> <code>?modifyCallback</code> <code>?focusInCallback</code>
<u><code>hiCreateMLTextField</code></u>	<code>?callback</code> <code>?focusInCallback</code> <code>?changeCB</code>
<u><code>hiCreateToggleField</code></u>	<code>?callback</code>
<u><code>hiCreateRadioField</code></u>	<code>?callback</code>
<u><code>hiCreateCyclicField</code></u>	<code>?callback</code>
<u><code>hiCreateButtonBoxField</code></u>	<code>?callback</code>

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

HI Form/Field Name	Callback Name
<u>hiCreateSpinBox</u>	?callback
<u>hiCreateFloatSpinBox</u>	?callback
<u>hiCreateScaleField</u>	?callback
<u>hiCreateHypertextField</u>	?callback
<u>hiCreateButton</u>	?callback
hiCreateFormButton is an alias to hiCreateButton and therefore the arguments remains the same for both the functions	
<u>hiCreateBooleanButton</u>	?callback
<u>hiCreateTabField</u>	?callback
<u>hiCreateListBoxField</u>	?callback ?changeCB ?doubleClickCB
<u>hiCreateReportField</u>	?callback ?doubleClickCB ?sortCallback
<u>hiCreateTreeTable</u>	?callback ?expandCallback ?collapseCallback ?hiDoubleClickCallback ?sortCallback

Related Topics

[Cadence User Interface Callback Routines](#)

[Function Object Callbacks](#)

Function Object Callbacks

The syntax of a function object callback is the following:

```
<callback> (o_field g_scope @rest_args)
```

where,

- *o_field* is the field whose callback is triggered. It can be used to access the properties of the field (*o_field*->*value*).
- *g_scope* is the location where the field is instantiated. It can be a HI form, a scroll-region field, or a page of a tab field. It can be used to access fields that are placed in the same scroll-region, tab page, or form where the field (*o_field*) for which the callback is triggered placed. To access the fields in the same form but outside *g_scope*, a function is provided. The function name is `hiGetFieldScope`.

The syntax for `hiGetFieldScope` is (`hiGetFieldScope g_field`).

When the given *g_field* is a HI field, which can be a scroll-region, tab or tab-page field, the function returns the scope containing the field. The return value can be a scroll-region, a tab page, a tab field, form, or nil.

- *@rest_args* is a variable argument list that is used for those fields whose callback expects one or more additional arguments.

Examples

In this example, the function `make_radians_degrees_fields()` creates two type-in fields, `Radians` and `Degrees` with a local callback `converter`. The callback is used to set the value of the `Radians` field based on the value of the `Degrees` field and vice versa. Also, in this example, the callback is not defined in the global namespace and that there is no need to know where the two type-in fields are to be placed.

```
(defun converter (field scope)
  (let ((scale 180.0/3.14159265))
    (caseq field->hiFieldSym
      (radians scope->degrees->value = (round field->value * scale))
      (degrees scope->radians->value = field->value / scale)
      (t (assert "this shouldn't happen")))))

(defun make_radians_degrees_fields ()
  (list
    (hiCreateFloatField ?name 'radians ?callback 'converter ?prompt "Radians"
      ?value 0.0)
```

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

```
(hiCreateIntField ?name 'degrees ?callback 'converter ?prompt "Degrees" ?value 0))
```

```
;; create the form
```

```
Form2D = hiCreateAppForm(  
    ?name          'Form2D  
    ?formTitle     "R2D&D2R Converter"  
    ?fields (make_radians_degrees_fields)  
)
```

```
hiDisplayForm(Form2D)
```

In this example, the function `list_png_files()` searches for all the `.png` files in the given directory, lists them in a cyclic field, and prints the filename when an icon is selected from the cyclic field.

```
(defun list_png_files ( dir )  
  (let ((png_files (rexMatchList ".png$" (getDirFiles dir)))  
        (png_table (makeTable "icon to .png file name" nil)))  
  
    ;; establish mapping  
    (mapcar (lambda (x) png_table[ (hiLoadIconFile x 24 24) ] = x) png_files)  
  
    (hiDisplayForm  
      (hiCreateAppForm  
        ?name 'png_list_form ?formTitle "PNG List" ?buttonLayout 'Close  
        ?fields (list (hiCreateCyclicField  
          ?name 'icons ?prompt "Icons" ?choices png_table->  
          ?callback (lambda (field scope item)  
            (printf "selected item is %L\n" png_table[ item]))))))))  
  
    (list_png_files (prependInstallPath "../share/cdssetup/icons/24x24"))
```

Function Object Callbacks with Additional Arguments

Function object callbacks in Cadence User Interface generally accept two standard arguments `o_field`, which represents the field whose callback is triggered and `g_scope`, which represents the location where the field is instantiated. Other than these two standard

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

arguments the callbacks of function object type for certain HI fields require additional arguments. The following table lists such callbacks:

Field Names	Callback Name	Callback Syntax
<u>hiCreateToggleField</u>	?callback	<pre>< toggle field callback > (o_field g_scope s_item g_value)</pre> where, <i>s_item</i> is the name of the item specified as a symbol. <i>g_value</i> specifies the value of the toggle item whose value was changed. If the toggle field is instantiated at multiple places, <i>g_scope</i> passed to a toggle field callback can be different from the one in which the toggle field is placed.
<u>hiCreateButtonBoxField</u>	?callback	<pre>< button box field callback > (o_field g_scope t_button_text)</pre> where, <i>t_button_text</i> specifies the text string of the button that was pressed.
<u>hiCreateBooleanButton</u>	?callback	<pre>< boolean button field callback > (o_field g_scope g_value)</pre> where, <i>g_value</i> specifies the value of the boolean button.
<u>hiCreateRadioField</u>	?callback	<pre>< radio field callback > (o_field g_scope t_selection)</pre> where, <i>t_selection</i> specifies the string representing the radio item that was selected.

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

Field Names	Callback Name	Callback Syntax
<u>hiCreateCyclicField</u>	?callback	<pre>< cyclic field callback > (o_field g_scope g_item)</pre> where, <i>g_item</i> specifies the item selected, which can be either text or icon.
<u>hiCreateReportField</u>	?callback	<pre>< report callback > (o_field g_scope l_selected_items)</pre> where, <i>l_selected_items</i> specifies a list of integers representing the selected items of the report. The <i>g_scope</i> argument passed to a report field callback can be different from the one in which the report field is placed, if the report field is instantiated multiple times.
	?doubleClickCB	<pre>< double click callback > (o_field g_scope x_item_index)</pre> where, <i>x_item_index</i> specifies an integer representing the item that was double- clicked.
	?sortCallback	<pre>< sort callback > (o_field g_scope l_sort_property)</pre> where, <i>l_sort_property</i> is a list (columnTo- Sort sortDirection), where columnTo- Sort is an integer indicating the column to sort and sortDirection is t or nil, indicating descending or ascending order, respectively.

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

Field Names	Callback Name	Callback Syntax
<u>hiCreateTreeTable</u>	?callback	<code>< tree table callback > (</code> <code> <i>o_field</i> <i>g_scope</i> <i>l_items</i>)</code> where, <i>l_items</i> is the list of selected items (symbols).
	?sortCallback	<code>< sort callback > (<i>o_field</i></code> <code> <i>g_scope</i> <i>l_sort_property</i>)</code> where, <i>l_sort_property</i> is a list (columnTo-Sort sortDirection), where <i>columnTo-Sort</i> is an integer indicating the column to sort and <i>sortDirection</i> is t or nil, indicating descending or ascending order, respectively.
	?expandCallback	<code>< expand callback > (<i>o_field</i></code> <code> <i>g_scope</i> <i>s_item</i>)</code> where, <i>s_item</i> is the symbol of the item being expanded.
	?collapseCallback	<code>< collapse callback > (<i>o_field</i></code> <code> <i>g_scope</i> <i>s_item</i>)</code> where, <i>s_item</i> is the symbol of the item being collapsed
	?doubleClickCB	<code>< double click callback > (</code> <code> <i>o_field</i> <i>g_scope</i> <i>s_item</i>)</code> where, <i>s_item</i> is the symbol of the item being double-clicked

Cadence User Interface SKILL Reference

Introduction to Cadence User Interface

Field Names	Callback Name	Callback Syntax
<u>hiCreateStringField</u>	?modifyCallback	<code>< field modify callback > (</code> <code> <i>o_field g_scope</i></code> <code> <i>t_latestTextValue</i></code> <code> <i>g_sourceOfChange</i>)</code> where, <i>t_latestTextValue</i> is the latest text value of the field <i>g_sourceOfChange</i> is <i>t</i> or <i>nil</i> , where <i>t</i> indicates that the change was made through the program (by changing the value of the field) and <i>nil</i> indicates that the change was made by a user action.
<u>hiCreateComboField</u>		
<u>hiCreateHypertextField</u>	?callback	<code>< hypertext field callback > (</code> <code> <i>o_field g_scope</i></code> <code> <i>t_hyperlinkName</i></code> <code> <i>t_hyperlinkRef</i>)</code> where, <i>t_hyperlinkName</i> is the name of the hyperlink. <i>t_hyperlinkRef</i> is the URL associated with the link.
<u>hiCreateAppForm</u>	?callback	<code>< form callback > (<i>r_form</i>)</code> where, <i>r_form</i> is the form for which an <i>OK</i> , <i>Cancel</i> , or custom action is triggered
<u>hiChangeFormCallback</u>	?callback	<code>< form callback > (<i>r_form</i>)</code> where, <i>r_form</i> is the form for which an <i>OK</i> , <i>Cancel</i> , or custom action is triggered

Related Topics

Cadence User Interface Callback Routines

Callbacks Supported by HI Forms and Fields

Command Interpreter Window

The Command Interpreter Window (CIW) is the first window you see and interact with using Cadence software. It contains menus with commands that allow you to do any operation Cadence supports. It is composed of different areas with separate functions, which are

- Window manager (optional) border
- Output history area
- Prompt line
- Input line
- Mouse bindings line

You can control the size and location of the CIW. It can stay on your screen throughout your work session or be temporarily hidden by other windows. The window manager border contains the location of the log file that is kept for each session. The banner contains the pulldown menus.

Each command line is copied into the output area from the input line as it is accepted for execution. Any error messages or output resulting from a command may appear at the bottom of the output area, depending on the settings of the log filter flags. This area is treated as a read-only disk file. The contents cannot be changed by direct keyboard entry.

Related Topics

[CIW Input Commands](#)

[General Environmental Functions](#)

[Process Control Functions](#)

[Log File Functions](#)

[Resource Management Functions](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

Text Attribute Functions

CIW Input Commands

The input line in the CIW holds the text of the current prompt until it is ready to be input. The command line text accumulates in the command input area, and a single event, usually typing a carriage return, triggers command execution.

Mouse and Keyboard Inputs

The CIW is activated by your keystrokes or mouse actions, just as the other windows are. All function keys, control keys, and mouse buttons are user configurable in any window. All alphanumeric or symbol character keystrokes have a literal meaning. It is a good idea to avoid binding any alphanumeric or symbol character keys to a command in the CIW.

The interpretation of keyboard and mouse actions depends on the application type of the window. All keys including function and control keys and mouse buttons are user-friendly in any window. Pressing a key submits a user-defined SKILL command string to the CIW for evaluation.

Input Prompt

When system initialization is complete, SKILL sends a prompt message to the prompt line and waits for an input event to occur. You can choose a different prompting style through the SKILL function `setPrompts`. If you change the prompt styles, the SKILL function is evaluated for each prompt. SKILL remains inactive until the command line message is received.

Parse Command Line

When a complete command line is received, it is parsed by SKILL. Arguments for SKILL functions are type checked against a template. All reasonable checking must be done at this level in the interests of consistency and economy of coding by the individual functions. All functions are executed according to the precedence of the SKILL command language.

Command Output Results

When a command line is received, SKILL sends the command language message to the output subwindow and to the transcript log. The result of the command is also written to the transcript log and possibly to the command output subwindow, depending on the setting of the log filter flags.

Cadence User Interface SKILL Reference

Command Interpreter Window

When a command runs successfully, the result is printed to the transcript log and possibly to the output subwindow.

Error Recovery

If an error is detected while evaluating a SKILL command, the standard error handler formats and generates a message before returning to the top level. This behavior can be modified by the use of `errSet` and `errSetString` within your SKILL program.

With the SKILL Development product, the standard error handler prints the contents of the SKILL stack and leaves you in a nested command interpreter to allow debugging. This includes detailed examination of the stack, and printing and modification of local variables. This nested interpreter runs using the same CIW windows as the top-level command interpreter. It is possible to switch between this and the non-development error handler.

If either of the standard error handlers are inappropriate, you can define your own with the SKILL Development product.

Interrupts

Interrupts are treated as errors. Execution resumes after an interrupt if you use the default error handler. The exception to this rule is `Control+C`, which always terminates the execution of the current SKILL command.

Nesting Commands

You can type a command in the CIW that prompts for more data, type another command that supports nesting, and then complete your first command. With command nesting you can gather additional data; the system supports multiple levels of command nesting. The default is one level, with a user-definable maximum of three levels of nesting.

Related Topics

[General Environmental Functions](#)

General Environmental Functions

This topic provides a list of Cadence® SKILL functions associated with Cadence environment and the CIW.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [dplp](#)
- [hiAddNonRepeatPrefix](#)
- [hiBoxCenter](#)
- [hiDisplaySaveForRestoreDialog](#)
- [hiEditfile](#)
- [hiEnqueueCmd](#)
- [hiFlush](#)
- [hiFlushCIW](#)
- [hiFlushInfo](#)
- [hiFlushLogFile](#)
- [hiFocusToCIW](#)
- [hiGetAttention](#)
- [hiSetMultiClickTime](#)
- [hiSetUserPreferences](#)
- [hiSetWindowDefaultPrompt](#)
- [hiSetWindowFocus](#)
- [hiSync](#)

dplp

```
dplp(  
    g_arg  
)  
=> t / nil
```

Description

Checks whether the specified argument is a disembodied property list, a list of symbol or value pairs that is not attached to a specific object or symbol.

The argument passes the test if it is a list that has an odd number of members and the first list member is `nil`.

Arguments

<i>g_arg</i>	Data to be evaluated.

Values Returned

<code>t</code>	Meets the test for a disembodied property list.
<code>nil</code>	Is not a disembodied property list.

Related Topics

[General Environmental Functions](#)

hiAddExtraRepeatCommand

```
hiAddExtraRepeatCommand(  
    t_command  
)  
=> t / nil
```

Description

Adds the specified SKILL command to the extra repeat command list so that the command becomes repeatable next time the `hiRepeat` function is called. Additionally, you can add the specified commands to the extra repeat command list that are not repeated automatically after their first run.

You can check the log for the correct name of the command you want to add to the extra repeat command list. The log file is `CDS.log` file by default or is the log file that is specified using the `-log` command-line option.

Arguments

<i>t_command</i>	Name of a SKILL command.

Value Returned

<i>t</i>	The specified command is added to the extra repeat command list.
<i>nil</i>	The specified command was not found or was not added to the extra repeat command list.

Examples

Adds the DRE opening command to the extra repeat command list and makes the command repeatable.

```
hiAddExtraRepeatCommand("dreInvokeDre(\"Virtuoso\")")
```

Related Topics

[hiRepeat](#)

[General Environmental Functions](#)

hiAddNonRepeatPrefix

```
hiAddNonRepeatPrefix(  
    t_prefix  
)  
=> t / nil
```

Description

Adds a string to the list of repeat filter strings. All strings is compared on every command run.

Any command that matches the prefix, for the number of characters of the prefix, does not get repeated by the `hiRepeat` command.

Arguments

<i>t_prefix</i>	The string prefix to be added to the list of repeat filter strings.

Values Returned

<i>t</i>	String prefix successfully added to the list of repeat filter strings.
<i>nil</i>	The <i>t_prefix</i> value is either empty or in an invalid format. A warning message is displayed in the CIW.

Examples

This section explains the use model of `hiAddNonRepeatPrefix()`.

- If a prefix is added that is more generic than a prefix that already exists in the list, the new prefix is added and it supersedes the more specific filter.

For example:

```
hiAddNonRepeatPrefix( "println" )  
=> t  
hiAddNonRepeatPrefix( "print" )  
=> t
```

As `print` filters more than `println`, that is, it is more generic, only `print` is used for filtering unless it is removed using `hiRemoveNonRepeatPrefix()`. However, `println` still exist in a superseded state, removing “`print`” would cause `println` to be used as a filter.

Cadence User Interface SKILL Reference

Command Interpreter Window

- The command returns `nil`, and does not add the prefix to the list of repeat filter strings if another prefix is added with this function, as it would cause the same item to be filtered again.

For example:

```
hiAddNonRepeatPrefix( "print" )
=> t
hiAddNonRepeatPrefix( "println" )
=> nil
```

The list contains `print`, but not `println` after the above commands are run.

- The prefix `print` would filter any command that begins with `print`, such as:

```
println("Hello, world")
```

or

```
printf ( "x = %d\n" x )
```

but would not filter commands that simply contain the word `print`, such as:

```
printf ( MyStr "name = %s" MyName )
```

Related Topics

[hiRepeat](#)

[hiGetNonRepeatPrefixes](#)

[hiRemoveNonRepeatPrefix](#)

[General Environmental Functions](#)

hiBoxCenter

```
hiBoxCenter(  
    l_boundingBox  
)  
=> l_boxCenter
```

Description

Returns the center point of a bounding box.

Arguments

<i>l_boundingBox</i>	Specifies a valid bounding box, which is represented by a list of two points separated by blank space. Each point is a list of two integers: the first point is the lower-left coordinate of the box and the second point is the upper-right coordinate.

Values Returned

<i>l_boxCenter</i>	List of two integers representing the center point of the bounding box.

Related Topics

[hiScaleBox](#)

[General Environmental Functions](#)

hiDisplaySaveForRestoreDialog

```
hiDisplaySaveForRestoreDialog(  
    )  
=> nil
```

Description

Displays the Save Session Form, which lets you specify the file in which to save your session information. This function is the programmatic equivalent of selecting *Options – Save Session* in the CIW.

Arguments

None

Value Returned

`nil` Returns `nil` immediately after displaying the form.

Related Topics

[hiSaveForRestore](#)

[Save Session Form](#)

[General Environmental Functions](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

hiEditfile

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

Description

Brings up the Specify a File to Edit form in which you can specify the file you want to edit and then calls the `edit` function on the file.

Arguments

None

Values Returned

<code>t</code>	Closes the form after clicking <i>OK</i> on the form.
<code>nil</code>	Closes the form after clicking <i>Cancel</i> on the form.

Related Topics

[General Environmental Functions](#)

hiEnqueueCmd

```
hiEnqueueCmd(  
    t_command  
    [ g_logCommand ]  
)  
=> t
```

Description

Adds *t_command*, which is to be run after returning from the current function and re-entering the event loop, to the command queue.

If the command is required to be replayed, the *g_logCommand* argument should be specified as a non-nil value. However, in most cases, the command is enqueued by another command that was logged, so it should rarely be necessary to use the *g_logCommand* argument.

Arguments

<i>t_command</i>	Specifies the command to be executed.
<i>g_logCommand</i>	Causes the command to be logged for replay.

Values Returned

<i>t</i>	Always returns <i>t</i> .

Examples

Runs the `normalEnqueueCmd` command in CIW:

```
procedure (normalEnqueueCmd()  
    printf("Before Enqueuing Cmd\n")  
    hiEnqueueCmd("printf(\"Enqueued Cmd\n\")")  
    printf("After Enqueuing Cmd\n")  
)
```

Output when the `normalEnqueueCmd` command is run in CIW. In this case, the enqueued command is run after `normalEnqueueCmd` returns.

Cadence User Interface SKILL Reference

Command Interpreter Window

```
Before Enqueuing Cmd
After Enqueuing Cmd
t
Enqueued Cmd
```

Runs the `enqueueCmdwithDBox` command in CIW:

```
procedure(enqueueCmdwithDBox()
    printf("Before Enqueuing Cmd\n")
    hiEnqueueCmd("printf(\"Enqueued Cmd\n\")")
    hiDisplayAppDBox(?name 'Testing)
    printf("After Enqueuing Cmd\n")
)
```

Output when the `normalEnqueueCmd` command is run in CIW. In this case, the enqueued command is run immediately after the dialog box is raised because that is where the event loop is re-entered.

```
enqueueCmdwithDBox()
    Before Enqueuing Cmd
    # Displaying modal dbox "Testing", title "Testing"
    Enqueued Cmd
```

Related Topics

General Environmental Functions

Cadence User Interface SKILL Reference

Command Interpreter Window

hiFlush

```
hiFlush( )  
=> t
```

Description

Synchronizes event queues and processes all exposure events. This ensures that newly exposed or updated windows and forms are drawn. This function is used primarily for quicktests.

Arguments

None

Value Returned

t The queue is flushed.

Related Topics

[hiSynchronize](#)

[General Environmental Functions](#)

hiFlushCIW

```
hiFlushCIW( )  
=> t
```

Description

Flushes any buffered output to the CIW (Command Interpreter Window). The function also processes the input and timer events. To avoid any additional event processing use the `hiFlushInfo` function.

Arguments

None

Value Returned

`t` Returns `t` when the queue is flushed.

Related Topics

[hiFlushInfo](#)

[hiSynchronize](#)

[General Environmental Functions](#)

hiFlushInfo

```
hiFlushInfo(  
    )  
=> t
```

Description

Processes exposure events to allow the output of the previous program to be displayed in the CIW during SKILL code processing.

Arguments

None

Value Returned

t	The output of the previous program is visible in the CIW when a SKILL code is run.

Examples

The following example ensures that any output from the `printf` statement is immediately visible in the CIW.

```
(defun CalculateDifficultProblem ( )  
  (let (IsItFinished (WhichStep 0))  
    )  
    while( IsItFinished == nil  
      ++WhichStep  
      printf("Processing step %d\n" WhichStep)  
      hiFlushInfo()  
      IsItFinished = ProcessStep(WhichStep)  
    )  
  )  
)
```

Related Topics

[General Environmental Functions](#)

hiFlushLogFile

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

Description

Flushes the contents of the main file and any open secondary log files.

Arguments

None

Value Returned

t	Contents of the log files are flushed.
nil	The log file is either opened during Virtuoso startup or closed during exit.

Related Topics

[hiSynchronize](#)

[General Environmental Functions](#)

hiFocusToCIW

```
hiFocusToCIW(  
    [ w_windowId ]  
)  
=> t / nil
```

Description

Temporarily refocuses the input window to the CIW until a return is entered.

The input is then refocused back to the window specified by *w_windowId*.

Arguments

<i>w_windowId</i>	ID of the window where you want the function to act. If <i>w_windowId</i> is not supplied, the function acts on the current window.

Values Returned

<i>t</i>	The input window is refocused.
<i>nil</i>	The input window is not refocused. An error message is issued.

Related Topics

[General Environmental Functions](#)

hiGetAttention

```
hiGetAttention(  
    [ x_loudness ]  
)  
=> t / nil
```

Description

Rings the bell in the keyboard or terminal.

Arguments

<i>x_loudness</i>	Overrides the loudness of the beep for that iteration. <i>x_loudness</i> ranges from -100 (silence) to 100 (full volume). The default is zero. Setting <i>x_loudness</i> to zero causes the beep to be at the system default loudness. The absolute value of a negative number is the percentage from the system default loudness toward silence, and a positive number is the percentage from the system default loudness toward full volume. The amount of loudness control is system dependent. For example, Sun systems have the same volume for values -99 through 100, but HP systems have a wide range of volume.

Values Returned

t	There was no error.
nil	An error message is issued.

Related Topics

[General Environmental Functions](#)

hiGetDisplayName

```
hiGetDisplayName( )  
=> t_name / nil
```

Description

Returns the current display name.

In non-graphical mode, it returns the virtual display name created by Xvfb.

Arguments

None

Value Returned

<i>t_name</i>	Current display name.

Related Topics

[General Environmental Functions](#)

[hiGetScreenSize](#)

hiGetMouseMoveSampleRate

```
hiGetMouseMoveSampleRate ( )  
    => x_rate
```

Description

Gets the current mouse sample rate.

Arguments

None

Value Returned

x_rate Current mouse sample rate.

Related Topics

[General Environmental Functions](#)

[hiGetMouseStopDetectTime](#)

[hiSetMouseMoveSampleRate](#)

[hiSetMouseStopDetectTime](#)

hiGetMouseStopDetectTime

```
hiGetMouseStopDetectTime ( )  
    => x_time
```

Description

Checks if the mouse is working using the current setting (in milliseconds).

Arguments

None

Value Returned

x_time The time when the mouse stopped working.

Related Topics

[General Environmental Functions](#)

[hiGetMouseMoveSampleRate](#)

[hiSetMouseMoveSampleRate](#)

[hiSetMouseStopDetectTime](#)

hiGetMultiClickTime

```
hiGetMultiClickTime( )  
=> x_milliseconds
```

Description

Returns the number of milliseconds that must elapse before a mouse click is recognized as a single click rather than the second click of a double click.

Arguments

None

Value Returned

<i>x_milliseconds</i>	Number of milliseconds that must elapse before a mouse click as recognized as a click.

Related Topics

General Environmental Functions

hiSetMultiClickTime

hiGetNonRepeatPrefixes

```
hiGetNonRepeatPrefixes (
    [ g_includeInternalPrefixes ]
)
=> l_prefixes
```

Description

Returns list of `hiRepeat` filter strings.

Arguments

g_includeInternalPrefixes

Returns both active and superseded prefixes that have been added with the `hiAddNonRepeatPrefix` command, as well as the list of internal prefix filter strings.

Value Returned

l_prefixes

List of prefix filter strings requested.

`nil`

Returns active list of prefixes added with `hiAddNonRepeatPrefix`, if no argument is passed or argument is `nil`.

Related Topics

[General Environmental Functions](#)

[hiRepeat](#)

[hiAddNonRepeatPrefix](#)

[hiRemoveNonRepeatPrefix](#)

hiGetScreenSize

```
hiGetScreenSize(  
    )  
=> l_sizes ( l_widthHeightPixels l_widthHeightMillimeters )
```

Description

Returns the size of the current display screen as a list of two lists, the first being the width and height in pixels, the second being the width and height in millimeters.

Arguments

None

Value Returned

<i>l_sizes</i>	The size of the current display screen, expressed as a list of two lists.
<i>l_widthHeightPixels</i>	Width and height in pixels
<i>l_widthHeightMillimeters</i>	Width and height in millimeters

Examples

Returns the actual physical size of the display.

```
hiGetScreenSize()  
((14490 900)  
(381 237)  
)
```

Related Topics

[General Environmental Functions](#)

[hiGetDisplayName](#)

hiGetBannerPoint

```
hiGetBannerPoint(  
    [ w_windowID ]  
)  
=> l_point / nil
```

Description

Retrieves the X:Y location value stored in the banner of the specified window. If the window ID is not specified, Virtuoso returns the location of the current window.

Arguments

<i>w_windowID</i>	Database ID of the window for which the X:Y location value needs to be retrieved.

Value Returned

<i>l_point</i>	The X:Y location value stored in the banner of the specified window. The list returned is undefined if the specified window has a banner but the X:Y values are not set by the graphics application that is displaying the data in the window.
<i>nil</i>	There is no current window or the specified window does not have a banner. A warning message is displayed stating the reason.

Related Topics

[General Environmental Functions](#)

hiGetBeepVolume

```
hiGetBeepVolume ( )  
    => x_loudness
```

Description

Retrieves the loudness of the beep.

Arguments

None

Value Returned

x_loudness

An integer indicating the loudness of the beep. It ranges from -100 (silence) to 100 (full volume).

The default is zero, which means the beep is at the system default loudness. The absolute value of a negative number is the percentage from the system default loudness toward silence, and a positive number is the percentage from the system default loudness toward full volume. The amount of loudness control is system dependent. For example, Sun systems have the same volume for values -99 through 100, but other systems may have a wide range of volume.

Related Topics

[General Environmental Functions](#)

hiGetCommandPoint

```
hiGetCommandPoint(  
    [ w_windowId ]  
    [ g_dontAdjust ]  
)  
=> l_pointList / nil
```

Description

Gets the location where the cursor was the last time a mouse button or keyboard key was pressed, but only until the cursor leaves that window.

Arguments

<i>w_windowId</i>	Window where you want the function to act. If <i>w_windowId</i> is not specified, the function defaults to the current window.
<i>g_dontAdjust</i>	If argument is non-nil, the point returned is not adjusted by the application adjust proc. This means that the point returned is the point where the mouse was clicked not the nearest point as determined by the application.

Value Returned

<i>l_pointList</i>	The x:y position in user units.
nil	The <i>w_windowId</i> is invalid or the cursor is not in the specified window.

Related Topics

[General Environmental Functions](#)

hiGetCIWindow

```
hiGetCIWindow( )  
=> w_windowId
```

Description

Retrieves the window identity of the CIW.

Arguments

None

Value Returned

<i>w_windowId</i>	The window identity of the Command Interpreter Window.

Examples

```
hiGetCIWindow( )  
=> window:1
```

Related Topics

General Environmental Functions

hiGetExtraRepeatCommands

```
hiGetExtraRepeatCommands(  
    )  
=> l_commands / nil
```

Description

Retrieves all commands added by the `hiAddExtraRepeatCommand` SKILL function in the extra repeat command list.

Arguments

None

Value Returned

<i>l_commands</i>	A list of SKILL commands.
<i>nil</i>	No commands were added to the extra repeat command list.

Examples

Previously added commands are retrieved from the extra repeat command list.

```
hiAddExtraRepeatCommand("dreInvokeDre(\"Virtuoso\")")
```

```
hiAddExtraRepratCommand("pteOpenForm(\"Options\")")
```

```
hiGetExtraRepeatCommands()  
=> ("dreInvokeDre(\"Virtuoso\")" "pteOpenForm(\"Options\")")
```

Related Topics

General Environmental Functions

hilsExtraRepeatCommand

hiGetPoint

```
hiGetPoint(  
    w_windowId  
)  
=> pointList / nil
```

Description

Retrieves the *xy* position for the current location of the cursor within the boundaries of the window. It returns the *xy* position even if the cursor is in another window or form that is covering the specified window, as long as the cursor is within the boundaries of the specified window.

Arguments

<i>w_windowId</i>	Window where you want the function to run.

Values Returned

<i>pointList</i>	The <i>pointList</i> <i>xy</i> position in user units.
<i>nil</i>	The <i>w_windowId</i> is invalid or the cursor is outside the boundaries of the specified window.

Related Topics

[General Environmental Functions](#)

hiGetRepeatCommand

```
hiGetRepeatCommand( )  
=> t_SkillCommand / nil
```

Description

Returns the command that would be run by `hiRepeat`, which the command that was last run in the CIW.

During replay, this function always returns `nil`.

Arguments

None

Value Returned

<i>t_SkillCommand</i>	The command that was last run in the CIW.
<code>nil</code>	No command is run by <code>hiRepeat</code> .

Related Topics

[hiRepeat](#)

[General Environmental Functions](#)

hiGraphicMode

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

Description

Checks if the Virtuoso environment is currently running in graphical mode. Returns `nil` if it is running in `-nograph` mode.

Arguments

None

Value Returned

<code>t</code>	Virtuoso is in graphical mode.
<code>nil</code>	Virtuoso is in non-graphical mode.

Related Topics

[General Environmental Functions](#)

hiInsertBlankCIWOutputPage

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

Description

Inserts a blank page in the CIW output area.

Arguments

None

Values Returned

t	A blank page was inserted into the CIW output area.
nil	An error occurred.

Examples

```
hiInsertBlankCIWOutputPage()  
=> t
```

Related Topics

[General Environmental Functions](#)

hiIsExtraRepeatCommand

```
hiIsExtraRepeatCommand(  
    t_command  
)  
=> t / nil
```

Description

Checks whether the specified SKILL command is in the extra repeat command list.

Arguments

<i>t_command</i>	Name of SKILL command.

Values Returned

<i>t</i>	The specified command is in the extra repeat command list.
<i>nil</i>	The specified command is not in the extra repeat command list.

Examples

Checks if the Design Resource Editor open command is in the extra repeat command list.

```
hiIsExtraRepeatCommand("dreInvokeDre(\"Virtuoso\")")
```

Related Topics

[General Environmental Functions](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

hiQuit

`hiQuit ( )`

Description

Quits the Virtuoso session. It displays a dialog box to confirm if you wish to quit and then calls `exit` if you click *Yes*.

When typing in the CIW, it is quicker to use `exit`. When assigning bindkeys, it is safer to use `hiQuit`, because you cannot immediately quit by accidentally pressing the bindkey.

Arguments

None

Value Returned

`nil` `exit` failed.

This function does not return a value if `exit` succeeds.

Related Topics

[General Environmental Functions](#)

hiRegTimer

```
hiRegTimer(  
    t_callbackString  
    x_tenthsOfSeconds  
)  
=> t / nil
```

Description

Registers a SKILL function string that is initialized after the specified time.

Cadence recommends that you use `hiEnqueueCmd` to register a command that is called later without strict timing requirement.

Arguments

<i>t_callbackString</i>	SKILL function name (string) that is to be executed after the specified time (<i>x_tenthsOfSeconds</i>).
<i>x_tenthsOfSeconds</i>	Time, in tenths of seconds, after which the callback is run.

Values Returned

t	The function was successful.
nil	There was an error.

Related Topics

General Environmental Functions

hiEnqueueCmd

hiRemoveExtraRepeatCommand

```
hiRemoveExtraRepeatCommand(  
    t_command  
)  
=> t / nil
```

Description

Removes the specified SKILL command from the extra repeat command list. The commands in this list are added using `hiAddExtraRepeatCommand`. When the command is removed it becomes unrepeatable.

Arguments

<i>t_command</i>	Name of SKILL command.

Values Returned

<i>t</i>	The specified command is removed from the extra repeat command list.
<i>nil</i>	The specified command was not removed from the extra repeat command list.

Examples

Removes the DRE opening command from the extra repeat list.

```
hiRemoveExtraRepeatCommand("dreInvokeDre(\"Virtuoso\")")
```

Related Topics

[General Environmental Functions](#)

[hilsExtraRepeatCommand](#)

hiRemoveNonRepeatPrefix

```
hiRemoveNonRepeatPrefix(  
    t_prefix  
)  
=> t / nil
```

Description

Removes a repeat filter prefix that has been previously added using `hiAddNonRepeatPrefix`.

This command does not remove functions from the internal repeat filter list.

Arguments

<i>t_prefix</i>	The string prefix to be removed from the list of repeat filter strings.

Values Returned

t	Prefix was found in the list and has been removed.
nil	Prefix was not found in the list or <i>t_prefix</i> was an empty string.

Related Topics

[General Environmental Functions](#)

[hiAddNonRepeatPrefix](#)

[hiGetNonRepeatPrefixes](#)

hiRepeat

```
hiRepeat( )  
=> t
```

Description

Re-runs the last command run in the CIW.

Arguments

None

Value Returned

t Always returns t.

Related Topics

[General Environmental Functions](#)

[hilsExtraRepeatCommand](#)

hiSaveForRestore

```
hiSaveForRestore(  
    [ t_fileName ]  
)  
=> t / t_failedFilePath
```

Description

Saves your session information in the file you specify. This function is the programmatic equivalent of selecting the *Options – Save Sessions* command in the CIW and saving a file in the Save Sessions form that appears.

Arguments

<i>t_fileName</i>	The name of the file in which you want to save your session; the name can include a path. If you do not specify a path, the SKILL path is used to determine the directory in which the file is saved. If you do not specify this argument or if you specify an empty string (""), the session is saved in a file named <code>cdsSession.save</code> in the directory determined by the SKILL path.

Values Returned

<i>t</i>	The session information was saved in the file you specified.
<i>t_failedFilePath</i>	The full path to the file if the file could not be opened.

Related Topics

[General Environmental Functions](#)

[hiDisplaySaveForRestoreDialog](#)

hiScaleBox

```
hiScaleBox(  
    l_boundingBox  
    n_scale  
)  
=> l_scaledBBox
```

Description

Returns a bounding box that has the same center point as the bounding box passed, but whose sides are scaled by $1/n\_scale$.

For example, if you pass 2 as the scale, the sides are half as long. If you pass 0.5 as the scale, the sides are twice as long.

You enter the lower left and upper right coordinates of the bounding box. The function calculates the center point, scales the sides, and returns the new lower left and upper right coordinates.

Arguments

<i>l_boundingBox</i>	List of the lower left and upper right coordinates of the initial bounding box.
<i>n_scale</i>	Scaling factor.

Values Returned

<i>l_scaledBBox</i>	Returns the bounding box with the same center point as the bounding box passed.

Examples

This example starts with a bounding box that is 100 units high by 200 units wide and gets two bounding boxes with the same center point, one whose sides are 50 units by 100 units and the other whose sides are 200 units by 400 units.

```
let ( ( ( myBBox list( 0:0 200:100 ) ) smallerBBox biggerBBox )  
    smallerBBox = hiScaleBox(myBBox 2 )  
    ; result is ( ( 50 25 ) ( 150 75 ) ), which is 1/2 the dimensions  
    biggerBBox = hiScaleBox(myBBox .5 )
```

Cadence User Interface SKILL Reference

Command Interpreter Window

```
    ; result is ( ( -100.0 -50.0 ) ( 300.0 150.0 ) ), which is  
    ; twice the dimensions  
)
```

Here, the return values are floating-point numbers, even though the original bounding box consist integers.

Related Topics

General Environmental Functions

hiAddNonRepeatPrefix

hiSetBeepVolume

```
hiSetBeepVolume (  
    [ x_loudness ]  
)  
=> t
```

Description

Sets the loudness of the beep when a default in the system is detected.

Argument

x_loudness

An integer representing the loudness of the beep.
x_loudness ranges from -100 (silence) to 100 (full volume).

The default is zero. Setting *x_loudness* to zero causes the beep to be at the system default loudness. The absolute value of a negative number is the percentage from the system default loudness toward silence, and a positive number is the percentage from the system default loudness toward full volume. The amount of loudness control is system dependent. For example, Sun systems have the same volume for values -99 through 100, but other systems may have a wide range of volume.

If you set *x_loudness* to an integer less than -100, it is set to -100. If you set it to an integer more than 100, it is set to 100.

Related Topics

[General Environmental Functions](#)

hiSetMouseMoveSampleRate

```
hiSetMouseMoveSampleRate(  
    x_rate  
)  
=> t / nil
```

Description

Sets the number of mouse motion events to be processed.

Argument

<i>x_rate</i>	Sets % of the level of mouse motion events that should be processed. Valid values: 0–100.

Value Returned

<i>t</i>	Mouse move sample rate successfully set.
<i>nil</i>	Action failed.

Related Topics

General Environmental Functions

mouseMoveSampleRate

hiGetMouseStopDetectTime

hiGetMouseMoveSampleRate

hiSetMouseStopDetectTime

hiSetMouseStopDetectTime

```
hiSetMouseStopDetectTime(  
    x_time  
)  
=> x_time / nil
```

Description

Sets the stop detection time to detect if a mouse has stopped moving.

Argument

<i>x_time</i>	The time in milliseconds used to detect if the mouse has stopped moving. That is, if no motion events in <i>x_time</i> are received then the mouse can be classed as having stopped moving.

Value Returned

<i>x_time</i>	Mouse stop detect time was set as specified.
nil	Action failed.

Related Topics

[General Environmental Functions](#)

[mouseStopDetectTime](#)

[hiGetMouseStopDetectTime](#)

[hiSetMouseMoveSampleRate](#)

[hiGetMouseStopDetectTime](#)

hiSetMultiClickTime

```
hiSetMultiClickTime(  
    x_milliseconds  
)  
=> t
```

Description

Sets the number of milliseconds that must elapse before a mouse click is recognized as a separate single click rather than the second click of a double click.

The valid range is 50–1000 units (1/20th of a second to one second). Values below 50 are set to 50 and a warning is output. Values above 1000 are set to 1000 and a warning is output.

Argument

<i>x_milliseconds</i>	Number of milliseconds that should elapse before another mouse click is recognized as a separate single click. Valid values: 50–1000.

Value Returned

<i>t</i>	Returns <i>t</i> after the time is set.

Related Topics

[hiGetMultiClickTime](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

hiSetUserPreferences

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

Description

Displays the User Preferences form with the latest values.

Before displaying the form, `hiSetUserPreferences` checks for changes to the form values that might have been made through SKILL or environment variables and synchronizes the form with those values.

Arguments

None

Values Returned

<code>t</code>	Values are set.
<code>nil</code>	The operation failed.

Related Topics

[General Environmental Functions](#)

hiSetWindowDefaultPrompt

```
hiSetWindowDefaultPrompt(  
    w_windowId  
    t_prompt  
)  
=> t_prevPrompt / nil
```

Description

Sets the prompt for the window associated with the given window ID instead of the default SKILL prompt. The function returns either the value of the previously set prompt or `nil`. If you specify an empty string for `t_prompt`, the function resets the prompt to the default value, which is the SKILL prompt.

Arguments

<code>w_windowId</code>	A window ID you specify.
<code>t_prompt</code>	String specifying the prompt for the window associated with the given window ID.

Values Returned

<code>t_prevPrompt</code>	Returns the value of the previously set prompt.
<code>nil</code>	Returns <code>nil</code> if the previously set value was a SKILL prompt.

Related Topics

[General Environmental Functions](#)

hiSetWindowFocus

```
hiSetWindowFocus (
    w_windowId
    [ g_setCurrent ]
)
=> t / nil
```

Description

Sets the focus to the window associated with the given window ID.

Arguments

<i>w_windowId</i>	Specifies the ID of the window you want to set focus to.
<i>g_setCurrent</i>	A non-nil value specifies the window with the given ID is set as the current window, if it is not a session or dockable window.

Values Returned

<i>t</i>	Focus is set to the window with the given ID.
<i>nil</i>	Focus is not set.

Related Topics

General Environmental Functions

hiSynchronize

```
hiSynchronize(  
    g_ignored  
)  
=> t
```

Description

Flushes the event queue by calling `QApplication::flush()`.

Argument

<i>g_ignored</i>	Ignores the argument.

Value Returned

t	Always returns t.

Examples

```
hiSynchronize( t )
```

Related Topics

General Environmental Functions

hiFlush

hiSync

```
hiSync(  
    s_deferredDelete  
)  
=> t / nil
```

Description

For programs with several GUI-related creation and deletion operations running in a loop, Qt objects deferred to be deleted later may accumulate and affect performance. This function solves this issue by forcing Qt to immediately delete these objects.

Argument

<i>s_deferredDelete</i>	Delete objects deferred for deletion. Valid value: <code>deferredDelete</code>

Values Returned

<code>t</code>	The operation was successful.
<code>nil</code>	The operation failed.

Examples

In the following example, the `hiSync` SKILL function periodically deletes the objects created in the `while` loop.

```
i = 1  
while(i <= 500  
  
    schWin = deOpen('(nil libName "TEST" cellName "TEST" viewName "schematic")  
nil "edit")  
    schDB = schWin->cellView  
    hiSetCurrentWindow(schWin)  
    instDB = dbFindAnyInstByName(schDB "NM0")  
    geSelectFig(instDB)  
    schHiObjectProperty()  
    hiDisplayForm(schObjPropForm)  
    schHiCheckAndSave()
```

Cadence User Interface SKILL Reference

Command Interpreter Window

```
schObjPropForm->m->value="5"  
schObjPropForm->fingers->value="50"  
schObjPropForm->switchSD->value= t  
  
hiFormDone (schObjPropForm)  
schHiCheckAndSave ()  
hiCloseWindow (schWin)  
  
if (mod(i 5)==0 hiSync('deferredDelete))  
    i++  
);while
```

Related Topics

General Environmental Functions

Process Control Functions

This topic provides a list of Cadence® SKILL functions associated with global abort flag. This flag indicates whether the user has pressed `Ctrl+C` to stop a Cadence software process, but it can also be set pragmatically.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiCheckAbort](#)
- [hiGetUserAbort](#)
- [hiResetAbort](#)
- [hiSetAbort](#)

hiCheckAbort

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

Description

Returns the value of the global abort flag.

This flag can be set to `t` by calling `hiSetAbort` and to `nil` by calling `hiResetAbort`. The flag normally is set to `t` when the SKILL interrupt flag is set to `t`, typically when a user presses `Ctrl+C` in a Virtuoso window. The global abort flag, once set, remains set to `t` unless specifically reset using the SKILL routine `hiResetAbort` or when cleared by Cadence applications.

Arguments

None

Values Returned

<code>t</code>	Value returned of the global abort flag.
<code>nil</code>	No value returned and calls <code>hiResetAbort</code> .

Related Topics

[Process Control Functions](#)

[hiGetUserAbort](#)

[hiSetAbort](#)

[hiResetAbort](#)

hiGetUserAbort

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

Description

Aborts a procedure when `Ctrl+C` is pressed.

This function also updates the tumbling dice cursor. In procedures that are compute-intensive, you should call `hiGetUserAbort` at least once a second to update the cursor and inform the user that processing has not stopped.

Normally, if the function returns `t`, you should abort processing and return control to the user.

Arguments

None

Values Returned

<code>t</code>	The user has pressed <code>Ctrl+C</code> .
<code>nil</code>	The user has not pressed <code>Ctrl+C</code> .

Related Topics

Process Control Functions

hiCheckAbort

hiSetAbort

hiResetAbort

hiResetAbort

```
hiResetAbort ( )  
=> t
```

Description

Sets the global abort flag to `nil`. This flag can be set to `t` by calling `hiSetAbort`.

Arguments

None

Value Returned

`t` The flag is set to `nil`.

Related Topics

[Process Control Functions](#)

[hiCheckAbort](#)

[hiGetUserAbort](#)

[hiSetAbort](#)

hiSetAbort

```
hiSetAbort ( )  
=> t
```

Description

Sets the global abort flag to `t`. This flag can be set to `nil` by calling `hiResetAbort`.

Arguments

None

Value Returned

`t` The global abort flag is set to `t`.

Related Topics

[Process Control Functions](#)

[hiCheckAbort](#)

[hiGetUserAbort](#)

[hiResetAbort](#)

Log File Functions

This topic provides a list of Cadence® SKILL functions associated with the main session log file, `CDS.log`, and ancillary user-created log files.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiEndLog](#)
- [hiGetLogFileName](#)
- [hiIsInReplay](#)
- [hiLogDragEvents](#)
- [hiPrintToLogFile](#)
- [hiReplayFile](#)
- [hiSetFilter](#)
- [hiSetFilterOptions](#)
- [hiStartLog](#)

hiEndLog

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

Description

Stops writing the transaction log to the file specified by `hiStartLog`.

Use `hiStartLog` and `hiEndLog` to create short ancillary log files for specific operations. All CIW transactions continue to be logged in the main log file `CDS.log`.

Arguments

None

Value Returned

<code>t</code>	The transaction log is no longer being written to the file.
<code>nil</code>	Writing to the specified file was not stopped. An error message is also issued.

Related Topics

[hiStartLog](#)

[Log File Functions](#)

hiGetLogFileName

```
hiGetLogFileName ( )  
    => t_logPath
```

Description

Returns the rooted file name of the Virtuoso session log file.

Arguments

None

Value Returned

<i>t_logPath</i>	Rooted file name of the log file.

Related Topics

[Log File Functions](#)

[hiReplayFile](#)

[hiIsInReplay](#)

[hiSetFilter](#)

[hiSetFilterOptions](#)

hiIsInReplay

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

Description

Indicates whether your Virtuoso session is in replay mode, that is, if commands are being executed from a replay file, which is typically a log file. Your session is in replay mode if you are running a Virtuoso workbench with the `-replay logfile` or `-test` option or if you are replaying a file with the `hiReplayFile` SKILL function.

The `hiIsInReplay` function returns `t` while commands are being executed from the replay file. When the end of the file is reached and normal, interactive use of the CIW is possible, the function returns `nil`.

Arguments

None

Value Returned

<code>t</code>	Your Virtuoso session is in replay mode.
<code>nil</code>	Your Virtuoso session is not in replay mode.

Related Topics

[Log File Functions](#)

[hiReplayFile](#)

hiLogDragEvents

```
hiLogDragEvents(  
    g_logDragEvents  
    g_logAllDragEvents  
)  
=> t
```

Description

Starts logging drag events (mouse movements in a graphics window while an enter function is active) in the log file so that they can be replayed later.

After you call this function with the *g_logDragEvents* argument set to *t*, all drag events are logged. When you replay the log file, the drag callbacks are replayed.

Stroke points are not logged.

Arguments

<i>g_logDragEvents</i>	Turns on the logging of drag events. When set to <i>nil</i> , it turns off the logging of drag events.
<i>g_logAllDragEvents</i>	Logs all drag events, including those drag events that resolve to the same point. When set to <i>nil</i> , it does not log all drag events.

Value Returned

<i>t</i>	The function was successful.

Related Topics

[Log File Functions](#)

hiPrintToLogFile

```
hiPrintToLogFile(  
    t_string  
    [ g_comment ]  
)  
=> t
```

Description

Prints the specified string to the `CDS.log` file and to any secondary log files that might have been opened with `hiStartLog`, without displaying it in the CIW output area.

Arguments

<i>t_string</i>	The string to be printed to the log file, it is preceded with the prefix <code>\o</code> in the log file.
<i>g_comment</i>	When the value of this argument is not <code>nil</code> , a comment line with the prefix <code>\#</code> is added to the log file. It is not displayed in the program output line.

Value Returned

<i>t</i>	Prints the specified string to the <code>CDS.log</code> file.

Examples

```
hiPrintToLogFile("This is an output line.")  
t
```

The following is printed to the log file:

```
This is an output line.
```

```
hiPrintToLogFile("Can also print multiple output lines\nwith one statement.")  
t
```

The following is printed to the log file:

```
Can also print multiple output lines  
with one statement.
```

Cadence User Interface SKILL Reference

Command Interpreter Window

```
hiPrintToLogFile("This is a comment line." t)
t
```

The following is printed to the log file:

```
# This is a comment line.
```

```
hiPrintToLogFile("Can also print multiple comment lines\nwith one statement." t)
t
```

The following is printed to the log file:

```
# Can also print multiple comment lines
# with one statement.
```

Related Topics

Log File Functions

hiStartLog

hiEndLog

hiGetLogFileName

hiLogDragEvents

hiSetFilter

hiSetFilterOptions

hiReplayFile

```
hiReplayFile(  
    s_fileName  
)  
=> t / nil
```

Description

Replays a log file. Use this function only for testing or debugging. In general, use the functions `load` or `loading` instead of `hiReplayFile`. `hiReplayFile` is intended to be run only from a Command Interpreter Window.

Only the input lines (prefixed by `\i`), the accelerated input lines (prefixed by `\a`), and lines that have no backslash prefix are filtered from the replay file and sent to SKILL.

If `hiReplayFile` is used in a SKILL file, all the commands in the SKILL file, including the ones that follow `hiReplayFile`, runs before the commands in the replay file. If the SKILL file is embedded in another SKILL file, all the commands in that file run before the commands in the replay file.

Because `hiReplayFile` replays a log file, to read a SKILL file (and execute functions in it), use the functions `load` or `loadi` instead.

`hiReplayFile` does not start from a replay file (a file being replayed by the `-replay` command line argument or by using `hiReplayFile`).

Arguments

<i>s_fileName</i>	The file to be read.
	Do not specify a file that has the <code>hiReplayFile</code> function in it.

Values Returned

<i>t</i>	The file is read.
<i>nil</i>	The file is not read. An error message is displayed.

Related Topics

[Log File Functions](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

hiIsInReplay

hiGetLogFileName

hiSetFilter

```
hiSetFilter(  
    [ t_filter ]  
)  
=> t / nil
```

Description

Sets the output message filter for the CIW with the option specified by *t_filter*. The output area of the CIW is refreshed immediately. If no option is specified, the Set Log File Display Filter form appears allowing you to specify the display options you want.

Program output, typed output, and error and warning messages are displayed by default.

Arguments

<i>t_filter</i>	Filter setting for the CIW, for example, <code>hiSetFilter "verbose"</code>. The following options are available: default: Only program output, user typed output, and error and warning messages are displayed. error: Only error and warning messages are displayed. verbose: Only menu command input, program output, menu command output, and error and warning messages are displayed. all: Menu command input, prompt output, program output, menu command output, and error and warning messages (everything except user typed output) are displayed.

Cadence User Interface SKILL Reference

Command Interpreter Window

Values Returned

t	The output message filter is set.
nil	The output message filter is not set. An error message is also displayed.

Related Topics

[Log File Functions](#)

hiSetFilterOptions

```
hiSetFilterOptions(  
    g_accelInput  
    g_prompt  
    g_output  
    g_returnAccel  
    g_typedReturn  
    g_errorOutput  
    g_warnOutput  
)  
=> t / nil
```

Description

Sets the output message filter for the CIW with the specified options. This function corresponds to the Set Log File Display Filter form (displayed by choosing *Options – Log Filter* in the CIW). Standard message output, return values from typed-in commands, errors and warning messages are displayed by default.

Arguments

<code>g_accelInput</code>	Displays accelerated input (menu and bindkey commands, form and field value settings and callbacks, etc.). Log file prefix: \a The default is <code>nil</code> .
<code>g_prompt</code>	Displays prompt output. Log file prefix: \p The default is <code>nil</code> .
<code>g_output</code>	Displays standard message output. Log file prefix: \o The default is <code>t</code> .
<code>g_returnAccel</code>	Displays return values from accelerated input. Log file prefix: \r The default is <code>nil</code> .
<code>g_typedReturn</code>	Displays return values from typed-in commands. Log file prefix: \t The default is <code>t</code> .
<code>g_errorOutput</code>	Displays error message output. Log file prefix: \e The default is <code>t</code> .

Cadence User Interface SKILL Reference

Command Interpreter Window

g_warnOutput Displays warning message output. Log file prefix: \w
The default is t.

The defaults listed are the values in effect at start-up.

Values Returned

t The output message filter is set.
nil The output message filter is not set. An error message is also displayed.

Related Topics

[Log File Functions](#)

hiStartLog

```
hiStartLog(  
    t_filename  
)  
=> t / nil
```

Description

Writes a transaction log to a file.

Use `hiStartLog` and `hiEndLog` to create short ancillary log files for specific operations. All CIW transactions continue to be logged in the main log file `CDS.log`.

Argument

<i>t_filename</i>	Name of file to which the transaction log is to be written.

Values Returned

<i>t</i>	The transaction log is written to the specified file.
<i>nil</i>	The transaction log is not written to the specified file. An error message is also displayed.

Related Topics

[Log File Functions](#)

[hiEndLog](#)

Resource Management Functions

This topic provides a list of Cadence® SKILL functions associated with return resource values from various databases and resource files.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiGetBBoxResource](#)
- [hiGetGeometryResource](#)
- [hiGetStringResource](#)

hiGetBBoxResource

```
hiGetBBoxResource(  
    t_resourceName  
    [ t_resourceClass ]  
)  
=> l_bBoxSpec
```

Description

Returns the bounding box specification defined in the `.Xdefaults` file for the specified resource.

Arguments

<i>t_resourceName</i>	Name of the resource to look up in the resource database.
<i>t_resourceClass</i>	Resource class to further limit the number of potential matches in the resource database. (See X Window System documentation for a further discussion on X resource classes. For general use, a <i>resourceName</i> is usually sufficient.)

Values Returned

<i>l_bboxSpec</i>	A <i>// ur</i> bounding box (list of two points) specification. The proper specification of a geometry resource consists of <code><width>x<height>{+-}xOffset{+-}yOffset</code>, where <i>x y</i> is relative to the upper left corner of the screen. A conversion is performed to convert the geometry resource to the <i>// ur</i> notation. Negative values are not returned. If a value resulting from this conversion is negative, 0 is substituted for that value.

Related Topics

[Resource Management Functions](#)

[hiGetGeometryResource](#)

hiGetGeometryResource

```
hiGetGeometryResource(  
    t_resourceName  
    [ t_resourceClass ]  
)  
=> l_resource
```

Description

Gets geometry resources from the X resource database.

You must preface your resource with *Opus*. The syntax for describing a standard geometry string is documented in the *Xlib Programming Manual*, and is of the form:

```
<width>x<height>{+-}<xoffset>{+-}<yoffset>
```

where items enclosed in <> are integers and items enclosed in {} are a set from which one item is allowed.

The syntax accepts only positive "+" values. Also, the x and y values are relative to the upper left corner of the screen.

Arguments

<i>t_resourceName</i>	Name of the resource to look up in the resource database.
<i>t_resourceClass</i>	Resource class to further limit the number of potential matches in the resource database. (See X documentation for a further discussion on X resource classes. For general use, a <i>resourceName</i> is usually sufficient.)

Values Returned

<i>l_resource</i>	The list(<i>x y width height</i>) value of the specified resource. Any component of the geometry specification not specified is returned as a <code>nil</code> list value. If a negative x, y, width, or height is specified, 0 is returned for that component.
<code>nil</code>	The specified resource does not exist.

Cadence User Interface SKILL Reference

Command Interpreter Window

Examples

The following examples are for a `.Xdefaults` file specifying the following resources:

```
Opus.geo1: 600x400+100+400
Opus.geo2: +1+2
Opus.geo3: 600+99
Opus.geo4: 400+1-20
Opus.geo5: 400x600+2
hiGetGeometryResource("geo1")
=> list( 100 400 600 400 )
hiGetGeometryResource("geo2")
=> list( 1 2 nil nil )
hiGetGeometryResource("geo3")
=> list( 99 nil 600 nil )
hiGetGeometryResource("geo4")
=> list( 1 0 400 nil )
hiGetGeometryResource("geo5")
=> list( 2 nil 400 600 )
hiGetGeometryResource("notThere")
=> nil
```

Related Topics

Resource Management Functions

hiGetBBoxResource

hiGetStringResource

```
hiGetStringResource(  
    t_resourceName  
    [ t_resourceClass ]  
)  
=> g_resource / nil
```

Description

Gets resources from the resource database of the X Window System.

Arguments

<i>t_resourceName</i>	Name of the resource to look up in the resource database.
<i>t_resourceClass</i>	Resource class to further limit the number of potential matches in the resource database. (See X documentation for a further discussion on X resource classes. For general use, a <i>resourceName</i> is usually sufficient.)

Values Returned

<i>g_resource</i>	The value of the specified resource.
<i>nil</i>	The resource does not exist.

Examples

The following examples are for an `.Xdefaults` file specifying the following resources:

```
Opus.someString: hello  
Opus.someInt: 1  
hiGetStringResource("someString")  
=> "hello"  
hiGetStringResource("someInt")  
=> "1"  
hiGetStringResource("notThere")  
=> nil
```

Related Topics

Resource Management Functions

Text Attribute Functions

This topic provides a list of Cadence® SKILL functions associated with learning and setting font size and other text attributes.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiGetFont](#)
- [hiGetFontInfo](#)
- [hiGetXFontName](#)
- [hiGetXFontName](#)
- [hiGetTextWidth](#)
- [hiQueryFont](#)
- [hiSetFont](#)
- [hiTextWidth](#)

hiGetFont

```
hiGetFont(  
    S_fontType  
)  
=> t_fontType / l_fontList / nil
```

Description

Returns the font type (if valid), or a list of valid font types if *t_fontType* is *all*. This function is being provided for compatibility with previous releases and should no longer be used.

Argument

<i>S_fontType</i>	One of the following as a string or symbol: 'text', "text", 'label', "label", 'ciw', "ciw", or 'all or "all". 'all or "all" returns a list of valid font types.

Values Returned

<i>t_fontType</i>	The font type (string) if valid.
<i>l_fontList</i>	The list of valid font types (string) if the argument is "all" or 'all.
nil	The font type is not one of "text", "label", "ciw", or "all". A warning message is displayed.

Examples

```
hiGetFont('text)  
=> "text"  
hiGetFont("label")  
=> "label"  
hiGetFont('notafont)  
=> nil  
hiGetFont("all")  
=> ("label" "text" "ciw")
```

Cadence User Interface SKILL Reference

Command Interpreter Window

Related Topics

Text Attribute Functions

hiSetFont

hiGetFontInfo

```
hiGetFontInfo(  
    S_fontType  
)  
=> l_fontInfoDpl / l_fontInfoDpls / nil
```

Description

Returns the Disembodied Property List (DPL), which contains the name, size, weight, and italic style along with the description of the font.

Argument

S_fontType

Specifies one of the following strings:

- "text" or 'text: Font used in CIW and encapsulation output windows, viewfiles, and the input area of text fields.
- "label" or 'label: Font used in banner labels, menus, and forms.
- "ciw" or 'ciw: Font used solely by CIW, with a default value of `text`. This is a special-case value that indicates that CIW is using the current setting for the `text` font.
If `ciw` is set to any value other than `text`, then that value is the font used by CIW for its input and output areas.

If the `text` font is changed, the `ciw` font gets changed (if it is still set to `text`). Additionally, if at any time during a session, if the `ciw` font is specifically changed to anything other than `text`, then changing the `text` font does not modify the `ciw` font from that point forward.

The purpose of the `ciw` font is to allow you to independently change CIW fonts without impacting forms that are using the `text` format.

- "all" or 'all: Font used for `ciw`, `label`, and `text`.

Cadence User Interface SKILL Reference

Command Interpreter Window

Values Returned

l_fontInfoDpl / l_fontInfoDpls

The Disembodied Property List (DPL) that contains the name, size, bold, and italic along with the description of the font for any of the values except "all" or 'all. If the value is specified as "all" or 'all, a DPL is returned with the properties of ciw, text, and label.

nil The font type is invalid.

Examples

Returns the DPL containing the name, size, bold, and italic style of the font type, ciw.

```
hiGetFontInfo( "ciw" )
=> ( nil name "text" size 11
    bold 50 italic nil
  )
```

Returns the DPLs containing the name, size, bold, and italic style of the font types, text, ciw, and label, respectively.

```
hiGetFontInfo( "all" )
=> ( nil text (nil name "Roboto Mono" size 11
    bold 50 italic nil
  )
    ciw ( nil name "Roboto Mono" size 12
    bold 50 italic nil
  )
    label ( nil name "Open Sans" size 11
    bold 50 italic nil
  )
  )
```

Related Topics

[Text Attribute Functions](#)

hiGetXFontName

```
hiGetXFontName (
    S_fontType
)
=> t_XLFD_font_name / nil
```

Description

Returns the default font name for the specified font type that was used prior to TrueType fonts.

Argument

<i>S_fontType</i>	One of the following as a string or symbol: text, label, ciw, error, or help.

Values Returned

<i>t_XLFD_font_name</i>	The pre-TrueType font name.
<i>nil</i>	The font type is not valid. A warning message is displayed.

Examples

```
hiGetXFont( "ciw" )
=> "-*-courier-medium-r-*-12-"
```

Related Topics

Text Attribute Functions

hiGetTextWidth

```
hiGetTextWidth(  
    S_fontType  
    t_text  
)  
=> x_width / -1
```

Description

Returns the width, in pixels, of a string of characters that has a specified font.

This value is useful when creating two-dimensional forms.

Arguments

<i>S_fontType</i>	Type of the font you want to use. Valid values: "text", 'text', "label", or 'label'. text: The font used in CIW, Encapsulation output windows, viewfiles, and the input area of text fields. label: The font used in banner labels, menus, and forms.
<i>t_text</i>	A character string to return the width.

Values Returned

<i>x_width</i>	The width of the string, in pixels.
-1	<i>t_fontType</i> is not a valid font.

Unlike the `hiGetTextWidth` function, if an invalid font is specified in the `hiTextWidth` function, then the return value is `nil`.

Examples

```
hiGetTextWidth( 'text "This is a string." )  
=> 119  
hiGetTextWidth( "label" "This is a string." )  
=> 84  
hiGetTextWidth( "notafont" "This is a string." )  
=> *WARNING* hiiGetTextWidth: Invalid font type "notafont", must be "text" or  
"label".
```

Cadence User Interface SKILL Reference

Command Interpreter Window

-1

Related Topics

Text Attribute Functions

hiTextWidth

hiQueryFont

```
hiQueryFont(  
    S_fontType  
)  
=> l_fontAttributes
```

Description

Returns the values of the attributes of *S_fontType*.

Argument

S_fontType

Type of the font you want to use.

Valid values: "text", 'text', "label", 'label', "ciw", or 'ciw.

Cadence User Interface SKILL Reference

Command Interpreter Window

Values Returned

l_fontAttributes The font attributes in a disembodied property list. The attributes returned are:

fontName: String representing the font name.

When *fontName* is passed as an argument, one of the special-case strings is the Qt font name being used, and the metrics is Qt metrics, which can be different from the Xfont metrics.

fontAscent: Integer representing the number of pixels above the font baseline.

fontDescent: Integer representing the number of pixels below the font baseline.

fontHeight: Integer representing the maximum height of the specified font (*fontAscent* + *fontDescent*).

fontWidth: Integer representing width of the specified font.

minfontWidth: Integer representing minimum width of the specified font.

maxFontWidth: Integer representing maximum width of the specified font.

lineSpacing: Integer representing the distance between two consecutive text lines.

leading: Integer representing the space between the text lines.

For a fixed-width font, such as Courier, the three width values, *fontWidth*, *minfontWidth*, and *maxfontWidth*, remains the same. For a variable-width font, such as Helvetica, *fontWidth* is the average width, *minFontWidth* is the width of the narrowest character (typically the lowercase “i”), and *maxFontWidth* is the width of the widest character.

Examples

```
hiQueryFont( "text" )
```

```
=> ( ( nil fontname "Roboto Mono" fontAscent 12 fontDescent 2 fontHeight 14  
fontWidth 7 minfontWidth 7 maxFontWidth 13 lineSpacing 15 leading 0 )
```

Cadence User Interface SKILL Reference

Command Interpreter Window

```
hiQueryFont( "label" )  
=> ( nil fontname "Open Sans" fontAscent 12 fontDescent 3 fontHeight 15 fontWidth  
6 minfontWidth 3 maxFontWidth 13 lineSpacing 15 leading -1 )
```

Related Topics

[Text Attribute Functions](#)

hiSetFont

```
hiSetFont(  
    [ S_fontType  
      ?name t_fontName  
      ?size x_fontSize  
      ?bold g_useBold  
      ?italic g_useItalics  
      ?fontInfo l_fontInfoDpl ]  
    )  
=> t / nil
```

Description

Sets the required font attributes, such as font name, font size, or font weight for `label`, `text`, or `ciw`. If no argument is specified, then the Set Fonts window is displayed. In this window, you can set any of the valid fonts and click the *Apply* button to save the changes. If any of the key arguments is specified in this function, then it is mandatory to specify *S_fontType*.

Cadence User Interface SKILL Reference

Command Interpreter Window

Arguments

S_fontType

One of the following strings:

- "text" or 'text': Font used in CIW and encapsulation output windows, viewfiles, and the input area of text fields.
- "label" or 'label': Font used in banner labels, menus, and forms.
- "ciw" or 'ciw': Font used solely by CIW, with a default value of `text`. This is a special-case value that indicates that CIW is the current setting for the `text` font.

If `ciw` is set to any value other than `text`, then that value is used as the font by CIW for its input and output areas.

If the `text` font is changed, the `ciw` font gets changed (if it is still set to `text`). Additionally, if at any time, during a session, if the `ciw` font is specifically changed to anything other than `text`, then changing the `text` font does not modify the `ciw` font from that point forward.

The purpose of the `ciw` font is to allow you to independently change CIW fonts without impacting forms that are using the `text` format.

?name *t_fontName*

The name of the font you want to use.

?size *x_fontSize*

The size of the font in pixels.

?bold *g_useBold*

Specify `nil` for the normal font or `t` for bold .

?italic *g_useItalics*

Specify `nil` for the normal font or `t` for italic.

?fontInfo *l_fontInfoDpl*

A disembodied property list (DPL) that contains the same properties as those returned by `hiGetFontInfo`.

Values Returned

`t`

Fonts are set.

Cadence User Interface SKILL Reference

Command Interpreter Window

`nil` Font types are not valid. An error message gets displayed.

Examples

Displays the Set Fonts window.

```
hiSetFont( )  
=> t
```

Changes the font size in CIW to 14 pixels.

```
hiSetFont( "ciw" ?size 14 )  
=> t
```

Related Topics

[Text Attribute Functions](#)

[hiGetFontInfo](#)

hiTextWidth

```
hiTextWidth(  
    S_fontType  
    t_text  
)  
=> x_width / nil
```

Description

Determines the width, in pixels, of a string of characters that has a specified font. Returns `nil` if the font type is not valid.

Arguments

<i>S_fontType</i>	Type of the font you want to use. Valid values: "text", 'text', "label", or 'label. text: The font used in CIW and Encapsulation output windows, viewfiles, and the input area of text fields. label: The font used in banner labels, menus, and forms.
<i>t_string</i>	Text string whose width you want to check.

Values Returned

<i>x_width</i>	The width, in pixels, of the string.
<i>nil</i>	<i>t_fontType</i> is not valid.

Examples

Returns the width, 119 of the text string.

```
hiTextWidth( 'text "This is a string." )  
=> 119
```

Returns the width, 84 of the text string.

```
hiTextWidth( "label" "This is a string." )  
=> 84
```

Returns `nil` as the font type is not valid.

Cadence User Interface SKILL Reference

Command Interpreter Window

```
hiTextWidth( "notafont" "This is a string." )  
=> *WARNING* hiiGetTextWidth: Invalid font type "notafont", must be "text" or  
"label".  
  
nil
```

Related Topics

[Text Attribute Functions](#)

Cdsenv Editor Functions

This topic provides a list of Cadence® SKILL functions associated with Cdsenv editor.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [startCdsenvEditor](#)
- [envCyclicStringToIndex](#)
- [envCyclicIndexToString](#)
- [envGetAvailableTools](#)
- [envGetDefVal](#)
- [envGetLoadedTools](#)
- [envGetModifiedTools](#)
- [envGetVal](#)
- [envGetVarType](#)
- [envIsToolModified](#)
- [envIsVal](#)
- [envLoadFile](#)
- [envLoadVals](#)
- [envRegLoadDumpTrigger](#)
- [envRegSetTrigger](#)
- [envSetToolCurrValToDefault](#)
- [envSetToolDefaultToCurrVal](#)
- [envSetVal](#)
- [envSetVarCurrValToDefault](#)
- [envSetVarDefaultToCurrVal](#)
- [envUnregSetTrigger](#)

startCdsenvEditor

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

Description

Starts Cdsenv Editor.

Arguments

None

Values Returned

t	Starts Cdsenv Editor.
nil	The operation failed.

Examples

```
startCdsenvEditor()
```

Related Topics

[Cdsenv Editor Functions](#)

[Cdsenv Editor Form](#)

envCyclicStringToIndex

```
envCyclicStringToIndex(  
    t_tool[.Partition]  
    t_varName  
    t_cyclicString  
)  
=> x_itemIndex
```

Description

Returns the index of the string for the given cyclic variable. This order is determined by the order of choices specified in the tool's registration file. The index is 0-based. -1 is returned for an invalid choice.

Arguments

t_tool[.Partition]

The name of the tool and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.

t_varName

The name of the variable

t_cyclicString

The string whose index in the list of choices is desired.

Values Returned

x_itemIndex

Returns 0-based index of this string in the choice list. If the specified string cannot be found among the list of choices, or other error conditions occur, -1 is returned.

Related Topics

[Cdsenv Editor Functions](#)

envCyclicIndexToString

```
envCyclicIndexToString(  
    t_tool[.Partition]  
    t_varName  
    x_itemIndex  
)  
=> t_stringChoice
```

Description

Returns the string choice associated with a particular index. This order is determined by the order of choices specified in the tool's registration file. The index is 0-based. -1 is returned for an invalid choice.

Arguments

<i>t_tool[.Partition]</i>]	The name of the tool (application) and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.
<i>t_varName</i>	The name of the variable
<i>x_itemIndex</i>	The index for which the string choice is desired. This is a 0-based index.

Values Returned

<i>t_stringChoice</i>	The string whose index is specified in the list of choices. If the specified index is not found among the list of choices, or other error conditions occur, -1 is returned.

Examples

```
envCyclicIndexToString( "ui" "showMouseBar" 1 )  
=> "Bottom"
```

Related Topics

[Cdsenv Editor Functions](#)

envGetAvailableTools

```
envGetAvailableTools( )  
=> l_availableTools
```

Description

Allows users to know which tools are available to be loaded.

The return value from `envGetAvailableTools` is different if the shell environment variable `CDS_LOAD_ENV` is set to `CSF`. If this has been set, it returns a list of tools that have been found using `CSF`.

So, for example, if `CDS_LOAD_ENV` is unset, or not set to `CSF`, then:

`envGetAvailableTools` may return `("ui")`, indicating that only the "ui" tool is available.

However, when `CDS_LOAD_ENV` is set to `CSF`, in addition to `("ui")` being returned, the tool checks all top-level directories in `CSF` (as specified by `setup.loc`) for files that end in `.cdsenv`.

The case remains the same if `csfLookupConfig` is being used and it has a setting of:
`INCLUDE .cdsenv`.

Arguments

None

Values Returned

`l_availableTools` The list of tools that are available to be loaded.

Related Topics

[Cdsenv Editor Functions](#)

[Specifying CSF Search for Additional Files](#)

[The setup.loc File Location](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

envGetDefVal

```
envGetDefVal(  
    t_tool[.Partition]  
    t_varName  
    [ s_varType ]  
)  
=> g_data
```

Description

Retrieves the registered default value for a variable. The tool is faulted in if it is not already faulted in, where faulting in is the process of reading the `.cdsenv` file and initializing the variables.

Arguments

<code>t_tool[.Partition]</code>]	The name of the tool (application) and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.
<code>t_varName</code>	The name of the variable.
<code>s_varType</code>	The type of the variable. Possible choices are 'int', 'float', 'string', 'boolean', 'cyclic' or 'toggle'.

Values Returned

<code>g_data</code>	The value of the variable. The following types are returned for the following data types:

Data Type	Returned Value Type
int	int
float	float
string	string
boolean	t nil
cyclic	string
toggle	list of t nil

Cadence User Interface SKILL Reference

Command Interpreter Window

Related Topics

[Cdsenv Editor Functions](#)

envGetLoadedTools

```
envGetLoadedTools( )  
=> l_loadedTools
```

Description

Allows users to know which tools have currently been loaded.

Arguments

None

Values Returned

<i>l_loadedTools</i>	The list of loaded tools.

Related Topics

[Cdsenv Editor Functions](#)

envGetModifiedTools

```
envGetModifiedTools( )  
=> l_modToolList / nil
```

Description

Returns a list of all tools with modified variables.

Arguments

None

Values Returned

<i>l_modToolList</i>	The list of tools that have modified variables.
<i>nil</i>	No tools found with modified variables.

Related Topics

[Cdsenv Editor Functions](#)

envGetVal

```
envGetVal(
    t_tool[.Partition]
    t_varName
    [ s_varType ]
)
=> g_data
```

Description

Retrieves the value of an environment variable. The tool is faulted in if it is not already faulted in, where faulting in is the process of reading the `.cdsenv` file and initializing the variables.

Arguments

<code>t_tool[.Partition]</code>]	The name of the tool and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.
<code>t_varName</code>	The name of the variable.
<code>s_varType</code>	The type of the variable. Possible choices are 'int', 'float', 'string', 'boolean', 'cyclic' or 'toggle'.

Values Returned

<code>g_data</code>	The value of the variable. The following types is returned for the following data types:

Data Type	Returned Value Type
int	int
float	float
string	string
boolean	t nil
cyclic	string
toggle	list of t nil

Related Topics

Cdsenv Editor Functions

envGetVarType

```
envGetVarType(  
    t_tool[.Partition]  
    t_varName  
)  
=> s_varType
```

Description

Returns the type of the variable specified. This query loads the tool, if it has not already been loaded.

Arguments

<i>t_tool[.Partition]</i>	The name of the tool (application) and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.
<i>t_varName</i>	The name of the variable.

Value Returned

<i>s_varType</i>	The type of the variable. Possible choices are 'int, 'float, 'string, 'boolean, 'cyclic, 'toggle or 'unknown.

Related Topics

[Cdsenv Editor Functions](#)

envIsToolModified

```
envIsToolModified(  
    t_toolName  
)  
=> t / nil
```

Description

Checks if any of the variables for a specified tool have been modified.

Argument

<i>t_toolName</i>	The name of the tool.

Values Returned

<i>t</i>	Some of the variables for the specified tool have been modified.
<i>nil</i>	None of the variables for the specified tool have been modified.

Related Topics

[Cdsenv Editor Functions](#)

envIsVal

```
envIsVal(  
    t_tool[.Partition]  
    t_varName  
    [ g_faultIn ]  
)  
=> t / nil
```

Description

Indicates whether a particular variable has been loaded and whether it exists for the given *tool[.partition]*.

Arguments

<i>t_tool[.Partition]</i>]	The name of the tool and the partition, if any. A partition is a way of grouping related sets of variables associated with a tool.
<i>t_varName</i>	The name of the variable.
<i>g_faultIn</i>	Determines whether to fault in the tool if the variable is not loaded. Faulting in is the process of reading the <i>.cdsenv</i> file and initializing the variables.

Values Returned

<i>t</i>	The variable is currently loaded. If not, the <i>faultIn</i> flag determines whether to fault in that tool and then the query for the existence of the specified variable takes place.
<i>nil</i>	The variable is not loaded.

Related Topics

[Cdsenv Editor Functions](#)

envLoadFile

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

Description

Specifies that a set of environment variables is to be loaded from a specified file. This function is equivalent to calling `envLoadVals( )` with the `?tool` argument set to `ALL`. All values loaded is marked as modified.

Specify a customized `.cdsenv` file for `t_fileName`. The `envLoadFile` function continues loading a `.cdsenv` file even if it encounters an error in the file.

Argument

<code>t_fileName</code>	The name of the file from which a set of environment variables is to be loaded.

Values Returned

<code>t</code>	The file is added to the list of files to search.
<code>nil</code>	The file could not be added to the list of files to search.

Related Topics

[envCyclicStringToIndex](#)

[Cdsenv Editor Functions](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

envLoadVals

```
envLoadVals(  
    ?envFile t_fileName  
    [ ?tool t_toolName ]  
)  
=> t / nil
```

Description

Specifies that a set of environment variables is to be loaded from a specified file.

This file name is added to the list of files to search, which is set to *your_install_dir/local/.cdsenv* and *~/ .cdsenv* upon startup. When a new tool is loaded, the list of files are searched for the specified tool's variables. The variables for each subsequent file on the list overwrite any previous file values.

Specify a customized *.cdsenv* file for *t_fileName*.

Arguments

<i>t_fileName</i>	The name of the file from which a set of environment variables is to be loaded.
<i>t_toolName</i>	The tool for which you want the environment variables to be loaded. The default is an empty string (""). If no tools are specified, all the variables associated with the tools that are already loaded gets loaded from the specified file. If "ALL" is specified as <i>toolName</i> , the variables from all tools listed in this file gets loaded (possibly for the first time). Only if "ALL" is not specified as the tool name, the file is added to the list of files to load from when a new tool is invoked, following the <i>.cdsenv</i> files in <i>your_install_dir/etc/tool</i> directory, <i>your_install_dir/local</i> directory, and the home directory.

Cadence User Interface SKILL Reference

Command Interpreter Window

Values Returned

t	The file is added to the list of files to search.
nil	The file could not be added to the list of files to search.

Related Topics

[envCyclicStringToIndex](#)

[envLoadFile](#)

[Cdsenv Editor Functions](#)

envRegLoadDumpTrigger

```
envRegLoadDumpTrigger(  
    ?tool t_toolName  
    ?loadFunc s_loadFunc  
    ?dumpFunc s_dumpFunc  
    ?preDumpFunc s_preDumpFunc  
)  
=> t / nil
```

Description

Registers the trigger function to be called when a tool's environment variables are loaded or saved to a file. Both the load and dump trigger functions are called after the load or dump of the environment variables, respectively. The `preDump` trigger is called just before the dump occurs. The dump trigger can be used to update the *defValues* of forms when a saving of defaults is performed.

Arguments

`?tool t_toolName` The name of the tool for which the triggers should be called.

To set the trigger for all tools at once, use `ALL` for the `t_toolName`.

When you specify the `ALL` option, the `?loadFunc`, `?dumpFunc`, and `?preDumpFunc` argument(s) must accept a single argument `t_toolName` that was triggered by load or dump. If the user-defined function is generic, and does not vary according to the specific `t_toolName` value, it is recommended to make the argument as optional. Therefore, the function could be used with either the `ALL` option or with the specified `t_toolName`.

Example 1:

```
procedure( myLoadFunc( @optional tool ) ... ) ; optional  
single argument, compatible with either "ALL" or a  
specific t_toolName specified.
```

Example 2:

```
procedure( myLoadFunc( tool ) ... ) ; mandatory single  
argument, compatible only when the "ALL" option is used.
```

`?loadFunc s_loadFunc`

The function to be called after the tool's environment variables are loaded.

Cadence User Interface SKILL Reference

Command Interpreter Window

?dumpFunc *s_dumpFunc*

The function to be called after the tool's environment variables are saved to a file.

?preDumpFunc *s_preDumpFunc*

The function to be called before any values are stored for the tool.

Values Returned

t The trigger functions were registered.

nil There was an error.

Related Topics

[Cdsenv Editor Functions](#)

envRegSetTrigger

```
envRegSetTrigger(  
    t_tool  
    s_triggerFunc  
)  
=> t / nil
```

Description

Registers a set trigger function for a tool. This trigger is called after an environment variable is set.

Arguments

<i>t_tool</i>	The name of the tool for which you want to register a set trigger function. If a set trigger function has already been set for the tool, this function returns <code>nil</code> .
<i>s_triggerFunc</i>	The trigger function to be called after an environment variable is set. The trigger function is passed the variable that was set and its old and new values. The trigger function has the following format: <pre>setTrigger([toolPartition] t_varName g_oldValue g_newValue)</pre> where <code>toolPartition</code> is the name of the partition that the variable being set belongs to, else <code>nil</code>.

Values Returned

<code>t</code>	The set trigger function was registered.
<code>nil</code>	There was an error.

Related Topics

[Cdsenv Editor Functions](#)

envSetToolCurrValToDefault

```
envSetToolCurrValToDefault(  
    t_tool[.Partition]  
)  
=> t / nil
```

Description

Sets the current value of all the environment variables to the default values.

Argument

`t_tool[ .Partition]`

The name of the tool and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.

Values Returned

<code>t</code>	The variables were reset.
<code>nil</code>	The variables could not be reset.

Related Topics

[Cdsenv Editor Functions](#)

envSetToolDefaultToCurrVal

```
envSetToolDefaultToCurrVal(  
    t_tool[.Partition]  
)  
=> t / nil
```

Description

Sets the default value of all the environment variables to the current value of the variables.

Arguments

t_tool[.Partition]

The name of the tool and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.

You can specify *t_tool[.partition]* in the following ways:

- *tool*: Resets all the variables at the top level of the tool and in all the partitions.
- *tool.*: Resets only the variables at the top level of the tool; does not reset the variables in the partitions.
- *tool.partition*: Resets only the variables in the specified partition.

Values Returned

<i>t</i>	The variables were reset.
<i>nil</i>	The variables could not be reset.

Related Topics

[Cdsenv Editor Functions](#)

envSetVal

```
envSetVal(  
    t_tool[.Partition]  
    t_varName  
    s_varType  
    g_newVal  
)  
=> t / nil
```

Description

Sets an environment variable value in the setup database. The tool is faulted in if it is not already faulted in, where faulting in is the process of reading the `.cdsenv` file and initializing the variables. If a set trigger function is registered for this variable (see the `envRegSetTrigger` function), it gets called after the value is set.

You cannot set an environment variable that has not been registered.

Arguments

<code>t_tool[.Partition]</code>	The name of the tool and the partition, if any. A partition is a way of grouping related sets of variables associated with a tool.
<code>t_varName</code>	The name of the environment variable.
<code>s_varType</code>	The type of the variable. Valid choices are 'int', 'float', 'string', 'boolean', 'cyclic' or 'toggle.
<code>g_newVal</code>	The new value for the environment variable.

Values Returned

<code>t</code>	The new value was set.
<code>nil</code>	There was an error and the new value could not be set.

Related Topics

[Cdsenv Editor Functions](#)

envSetVarCurrValToDefault

```
envSetVarCurrValToDefault(  
    t_tool[.Partition]  
    t_varName  
)  
=> t / nil
```

Description

Sets the current value of the environment variable you specify to the default value.

Arguments

t_tool[.Partition]

The name of the tool and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.

t_varName

The name of the environment variable.

Values Returned

t

The variables were reset.

nil

The variables could not be reset.

Related Topics

[Cdsenv Editor Functions](#)

envSetVarDefaultToCurrVal

```
envSetVarDefaultToCurrVal(  
    t_tool[.Partition]  
    t_varName  
)  
=> t / nil
```

Description

Sets the default value of the environment variable you specify to the current value of the variable.

Arguments

t_tool[.Partition]

The name of the tool and the partition (if any). A partition is a way of grouping related sets of variables associated with a tool.

t_varName

The name of the environment variable.

Values Returned

t The variables were reset.

nil There was an error.

Related Topics

[Cdsenv Editor Functions](#)

envStoreEnv

```
envStoreEnv(  
    [ ?envFile t_fileName ]  
    [ ?tool t_tool[.partition] ]  
    [ ?toolStatus t_toolStatus ]  
    [ ?varStatus s_varStatus ]  
    [ ?fileStatus t_fileStatus ]  
)  
=> t / nil
```

Description

Stores environment variables associated with tools to a specific file. You can choose to save variables for a specific tool, all loaded tools, or all possible tools. You can also specify whether you want to save all the variables for a tool, the variables that have been modified, or only the variables that have been modified and that are different from the default value.

This function is the programmatic equivalent of saving variables with the Save Defaults form.

Cadence User Interface SKILL Reference

Command Interpreter Window

Arguments

t_fileName Specifies the name of the file in which you want to store the environment variables. If not specified, the variables gets saved in `~/.cdsenv`.

t_tool[.partition]

The name of the tool and the partition, if any. Use this argument only if you have set *t_toolStatus* to "Loaded tools by name".

A partition is a way of grouping related sets of variables associated with a tool. You can specify the value of this argument in the following ways:

- *tool*: Saves all the variables at the top level of the tool and in all the partitions
- *tool.*: Saves only the variables at the top level of the tool; does not save the variables in the partitions
- *tool.partition*: Saves only the variables in the specified partition

If you specify more than one tool, separate the names with a space.

The default value of this argument is "ALL".

t_toolStatus

The tools you want to save. Specify one of the following strings:

- "All possible tools"
- "All loaded tools"
- "Loaded tools by name"

If you specify "Loaded tools by name," specify the names of the tools in the *t_tool[.partition]* argument.

If you specify "All possible tools," the value of the *s_varStatus* argument is ignored. All the variables are saved.

The default value is "All loaded tools".

Cadence User Interface SKILL Reference

Command Interpreter Window

s_varStatus

The variables you want to save. Specify one of the following symbols:

- `'modified`: Saves only those variables that have been modified; a variable is considered to be modified if it is changed in any way, even if its value is reset to the original value.
- `'modifiedFromDefault`: Saves only those variables that have been modified and that are different from the default; float variables gets saved if the difference between the current value of the variable and the default value is greater than the tolerance specified in the `.cdsenv` file.
- `'all`: Saves all the variables.

If you specified "All possible tools" as the value of the *t_toolStatus* argument, the *s_varStatus* argument is ignored. *s_varStatus* is assumed to be `'all`, regardless of the value you specify.

t_fileStatus

The file status. Specify one of the following strings:

- `Overwrite`: Overwrites the original file.
- `Merge values`: Merges values that have been modified in the current session with the original file.
- `Retain values`: Retains values from the original file.

The default is "Merge values".

The following table shows the values that are saved to an existing file based on the values of *s_varStatus* and *t_fileStatus*:

Cadence User Interface SKILL Reference

Command Interpreter Window

	Merge Values	Overwrite	Retain values
Modified	All values in the existing default settings file is retained unless the variable has been modified in the current session. If the variable has been modified, its modified value is saved. In addition, non-private values that are not included in the original file but that have been modified is added to the file.	Only non-private values that have been modified in the current session are saved. Any pre-existing contents of the file is removed.	All values in the existing default settings file are retained. Any additional values that are not included in the original file but that have been modified in the current session is added to the file.
Modified From Default	All values in the existing default settings file are retained unless the variable has been modified in the current session and is different from the default value. If the variable has been modified and its value is different from the default value, its new value gets saved. In addition, non-private values that are not included in the original file is added to the file if they have been modified and are different from the defaults.	Only non-private values that have been modified in the current session and that are different from the default values gets saved. Any pre-existing contents of the file are removed.	All values in the existing default settings file are retained. In addition, any non-private values that are not included in the original file is added to the file if they have been modified and are different from the defaults.

Cadence User Interface SKILL Reference

Command Interpreter Window

	Merge Values	Overwrite	Retain values
All	All values in the existing default settings file are retained unless the variable has been modified in the current session. If the variable has been modified, its modified value is saved. In addition, all non-private values for the tool are written to the file, whether or not they have been used in the current session.	All non-private values for the tool are saved whether or not they are used in the current session. Any pre-existing contents of the file are removed.	All values in the existing default settings file are retained. In addition, all non-private values for the tool gets written to the file, whether or not they have been used in the current session.

Examples

The following example shows different environment variables that need to be saved to the existing file (`myDefs`) based on the values of `t_toolStatus`, `t_tool[.partition]`, `s_varStatus`, and `t_fileStatus`. Assume that the list of all possible tools includes `testTool`, `anotherTool` and `tool3`.

Master variable registration file located in `etc/tools/testTool/.cdsenv`:

```
testTool    nestLimit    int    5    nil
testTool    showScrollBars boolean nil    nil
testTool    setWinTraversal boolean nil    nil
testTool    undoLevel    int    1    nil
```

Master variable registration file located in `etc/tools/anotherTool/.cdsenv`:

```
anotherTool var1    string "hi" nil
anotherTool var2    int    50    nil
anotherTool var3    cyclic "v2" nil ("v1" "v2" "v3")
```

Master variable registration file located in `etc/tools/tool3/.cdsenv`:

```
tool3.part1 check boolean nil    nil
tool3.partb level int    20    nil
tool3      auto    boolean t    nil
```

Existing file `myDefs`:

```
testTool    nestLimit    int 3
testTool    undoLevel    int 5
tool3      auto    boolean nil
tool3.partb level    int 9
anotherTool var3    cyclic "v3"
```

Cadence User Interface SKILL Reference

Command Interpreter Window

Environment variable settings in the current session:

testTool	nestLimit	int 2	(modified and different from default)
testTool	showScrollBars	boolean nil	
testTool	setWinTraversal	boolean t	(modified and different from default)
testTool	undoLevel	int 1	
tool3	auto	boolean t	(modified and same as default)

The following is a list of environment variables that need to be saved to file `myDefs` depending on the settings for `t_fileStatus`, `t_tool[.partition]`, `s_varStatus` and `t_toolStatus`.

For `?fileStatus Overwrite`, `?toolStatus Loaded` tools by name, `?tool testTool tool3`, and `?varStatus 'modified`:

testTool	nestLimit	int	2
testTool	setWinTraversal	boolean	t
tool3	auto	boolean	t

For `?fileStatus Merge` values, `?toolStatus Loaded` tools by name, `?tool testTool tool3`, and `?varStatus 'modified`:

testTool	nestLimit	int	2
testTool	setWinTraversal	boolean	t
testTool	undoLevel	int	5
tool3	auto	boolean	t
tool3.partb	level	int	9
anotherTool	var3	cyclic	"v3"

For `?fileStatus Retain` values, `?toolStatus Loaded` tools by name, `?tool testTool tool3`, and `?varStatus 'modified`:

testTool	nestLimit	int	3
testTool	setWinTraversal	boolean	t
testTool	undoLevel	int	5
tool3	auto	boolean	nil
tool3.partb	level	int	9
anotherTool	var3	cyclic	"v3"

For `?fileStatus Merge` values, `?toolStatus Loaded` tools by name, `?tool testTool tool3`, and `?varStatus 'modifiedFromDefault`:

testTool	nestLimit	int	2
testTool	setWinTraversal	boolean	t

For `?fileStatus Overwrite` and `?toolStatus All` possible tools:

testTool	nestLimit	int	2
testTool	showScrollBars	boolean	nil
testTool	setWinTraversal	boolean	t
testTool	undoLevel	int	1
anotherTool	var1	string	"hi"
anotherTool	var2	int	50
anotherTool	var3	cyclic	"v2"
tool3.part1	check	boolean	nil
tool3.partb	level	int	20
tool3	auto	boolean	t

Cadence User Interface SKILL Reference

Command Interpreter Window

For ?fileStatus Merge values and ?toolStatus All possible tools:

testTool	nestLimit	int	2
testTool	showScrollBars	boolean	nil
testTool	setWinTraversal	boolean	t
testTool	undoLevel	int	5
anotherTool	var1	string	"hi"
anotherTool	var2	int	50
anotherTool	var3	cyclic	"v2"
tool3.part1	check	boolean	nil
tool3.partb	level	int	9
tool3	auto	boolean	t

For ?fileStatus Retain values and ?toolStatus All possible tools:

testTool	nestLimit	int	3
testTool	showScrollBars	boolean	nil
testTool	setWinTraversal	boolean	t
testTool	undoLevel	int	5
anotherTool	var1	string	"hi"
anotherTool	var2	int	50
anotherTool	var3	cyclic	"v3"
tool3.part1	check	boolean	nil
tool3.partb	level	int	9
tool3	auto	boolean	nil

For ?fileStatus Overwrite, ?toolStatus Loaded tools by name, ?tool tool3, and ?varStatus 'all:

tool3	auto	boolean	t
tool3.part1	check	boolean	nil
tool3.partb	level	int	20

For ?fileStatus Overwrite, ?toolStatus Loaded tools by name, ?tool tool3., and ?varStatus 'all:

tool3	auto	boolean	t

For ?fileStatus Overwrite, ?toolStatus Loaded tools by name, ?tool tool3.part1, and ?varStatus 'all:

tool3.part1	check	boolean	nil

Related Topics

Cdsenv Editor Functions

envUnregLoadDumpTrigger

```
envUnregLoadDumpTrigger(  
    ?tool t_toolName  
    ?loadFunc s_loadFunc  
    ?dumpFunc s_dumpFunc  
    ?preDumpFunc s_preDumpFunc  
)  
=> t / nil
```

Description

Deregisters the specified trigger functions for the specified tool. The specified functions are deleted from the corresponding list of the trigger functions for the tool.

Arguments

<code>?tool <i>t_toolName</i></code>	The name of the tool from where the specified functions need to be deleted.
<code>?loadFunc <i>s_loadFunc</i></code>	The function to be deleted from the list of load trigger functions.
<code>?dumpFunc <i>s_dumpFunc</i></code>	The function to be deleted from the list of dump trigger functions.
<code>?preDumpFunc <i>s_preDumpFunc</i></code>	The function to be deleted from the list of <code>preDump</code> trigger functions.

Values Returned

<code>t</code>	The trigger functions were deregistered.
<code>nil</code>	The operation is not completed.

Examples

```
;; There are two different load trigger functions  
( defun loadTrigger_1 ( )    ( ... ))  
( defun loadTrigger_2 ( )    ( ... ))
```


Cadence User Interface SKILL Reference

Command Interpreter Window

```
;; they are registered to be called after the load of the tool's environment variables
```

```
envRegLoadDumpTrigger( ?tool "tool" ?loadFunc 'loadTrigger_1 )
```

```
envRegLoadDumpTrigger( ?tool "tool" ?loadFunc 'loadTrigger_2 )
```

```
envLoadFile( "someFileWhichContansToolEnvVars.cdsenv" )
```

```
;; => Here both load trigger functions are called
```

```
;; Now the loadTrigger_1 function is unregistered as a load trigger function
```

```
;; and it is deleted from the list of load trigger functions
```

```
envUnregLoadDumpTrigger( ?tool "tool" ?loadFunc 'loadTrigger_1 )
```

```
envLoadFile( "someFileWhichContansToolEnvVars.cdsenv" )
```

```
;; => Here only the loadTrigger_2 load trigger function is called
```

Related Topics

Cdsenv Editor Functions

envUnregSetTrigger

```
envRegSetTrigger(  
    t_tool  
    s_triggerFunc  
)  
=> t / nil
```

Description

Deregisters the specified set trigger function for the specified tool. The specified function is deleted from the list of the set trigger functions for the tool.

Arguments

<i>t_tool</i>	The name of the tool for which you want to unregister a set trigger function.
<i>s_triggerFunc</i>	The function to be deleted from the list of set trigger functions.

Values Returned

<i>t</i>	The set trigger function was deregistered.
<i>nil</i>	The operation is not completed.

Examples

```
;; There are two different set trigger functions  
( defun setTriggerVar1( toolName varName oldValue newValue ) ( ... ))  
( defun setTriggerVar2( toolName varName oldValue newValue ) ( ... ))  
  
;; they are registered to be called after the tool's environment variable is set  
envRegSetTrigger( "tool" 'setTriggerVar1 )  
envRegSetTrigger( "tool" 'setTriggerVar2 )  
  
envSetVal( "tool" "boolVar" 'boolean t )  
;; => Here both set trigger functions are called  
  
;; Now the setTriggerVar2 function is unregistered as a set trigger function  
;; and it is deleted from the list of set trigger functions  
envUnregSetTrigger( "tool" 'setTriggerVar2 )
```

Cadence User Interface SKILL Reference

Command Interpreter Window

```
envSetVal( "tool" "boolVar" 'boolean nil )  
;; => Here only the setTriggerVar1 function is called
```

Related Topics

[Cdsenv Editor Functions](#)

Cadence User Interface SKILL Reference

Command Interpreter Window

Menu Functions

This topic provides a list of Cadence® SKILL functions associated with menus.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiAddMenuItem](#)
- [hiCreate2DMenu](#)
- [hiCreateHorizontalFixedMenu](#)
- [hiCreateMenu](#)
- [hiCreateMenuItem](#)
- [hiCreatePulldownMenu](#)
- [hiCreateSimpleMenu](#)
- [hiCreateSeparatorMenuItem](#)
- [hiCreateSliderMenuItem](#)
- [hiCreateTypeinMenuItem](#)
- [hiSetTypeinMenuItemCompleterList](#)
- [hiCreateVerticalFixedMenu](#)
- [hiDeleteMenu](#)
- [hiDeleteMenuItem](#)
- [hiDisableMenuItem](#)
- [hiDisplayFixedMenu](#)
- [hiDisplayMenu](#)
- [hiDisplayWindowMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

- [hiEnableMenuItem](#)
- [hiFixedMenuDown](#)
- [hiGetMenuItems](#)
- [hiGetWindowMenu](#)
- [hiInsertMenuItem](#)
- [hiIs2DMenu](#)
- [hiSetMenuItemStatusTip](#)
- [hiSetMenuItemText](#)
- [hiSetWindowMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiAddMenuItem

```
hiAddMenuItem(  
    r_hiMenu  
    r_menuItem  
)  
=> t / nil
```

Description

Adds the menu item or action to the end of all instances of the specified *hiMenu*.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> or <code>hiCreatePulldownMenu</code> or <code>hiCreateSeparatorMenuItem</code> .
<i>r_menuItem</i>	Menu item description returned by <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> or action returned <code>hiCreateAction</code> .

Values Returned

t	The menu item was added.
nil	The menu item was not added. An error message is also returned.

Related Topics

[Menu Functions](#)

[hiCreateMenu](#)

[hiCreatePulldownMenu](#)

[hiCreateMenuItem](#)

[hiCreateSeparatorMenuItem](#)

[hiCreateSliderMenuItem](#)

[hiDeleteMenuItem](#)

Cadence User Interface SKILL Reference

Menu Functions

hiDisableMenuItem

hiCreate2DMenu

```
hiCreate2DMenu(  
    s_menuHandle  
    t_menuTitle  
    l_menuItems  
)  
=> r_hiMenu
```

Description

Creates a two-dimensional popup menu. Once this menu is created, you can display it with `hiDisplayMenu` or associate it with a window using `hiSetWindowMenu`.

A popup menu is a tear-off menu by default, that is, it can be torn off and displayed in its own window anywhere on the screen.

Popup menus can also be created with `hiCreateSimpleMenu` and `hiCreateMenu`.

Arguments

<i>s_menuHandle</i>	A unique global SKILL symbol used to reference this menu.
<i>t_menuTitle</i>	Name of the menu that is displayed in the menu bar. To set the access key for the menu, add an ampersand before the desired letter. For example, if the menu title is <code>Edit</code> and you want <code>E</code> to be the access key for the menu, you would specify this argument as <code>"&Edit"</code>.
<i>l_menuItems</i>	List of menu items, each item being a list containing the 2D representation of a form button or label field. Each item list contains three elements: ■ A button or label form field created by <code>hiCreateButton</code> or <code>hiCreateLabel</code> ■ The point where the upper-left corner of this field should be placed . ■ A list containing the width and height of the field.

Cadence User Interface SKILL Reference

Menu Functions

Value Returned

r_hiMenu The SKILL structure representing the menu.

Related Topics

[Menu Functions](#)

[hiCreateButton](#)

[hiCreateLabel](#)

[hiDisplayMenu](#)

[hiSetWindowMenu](#)

[hiCreateSimpleMenu](#)

[hiCreateMenu](#)

hiCreateHorizontalFixedMenu

```
hiCreateHorizontalFixedMenu(  
    s_menuHandle  
    l_menuItems  
    x_rows  
    x_cols  
    [ S_doneCallback ]  
    [ s_showTextWithIcon ]  
)  
=> r_hiMenu
```

Description

Creates a horizontal fixed menu that is a free-floating menu unattached to any application window. The fixed menu is created as a grid of *row* x *column* dimensions, and contains menu item objects created by `hiCreateMenuItem`. An additional *Done* item is added as the last item. Selecting *Done* removes the fixed menu from the screen.

Menu items you want to display in the fixed menu are inserted in row-major order.

The width and height of the horizontal fixed menu is determined by the number of rows and columns in the menu. Once the menu is created, you can display it with `hiDisplayFixedMenu`.

Horizontal fixed menus appear across the entire top or bottom of the root window (screen), rather than an application window. These fixed menus are shared among all Cadence application windows in the root window.

Arguments

<i>s_menuHandle</i>	A unique global SKILL symbol used to reference this menu.
<i>l_menuItems</i>	List of menu items created by <code>hiCreateMenuItem</code> .
<i>x_rows</i>	Number of rows in the fixed menu.
<i>x_cols</i>	Number of columns in the fixed menu.

Cadence User Interface SKILL Reference

Menu Functions

S_doneCallback Symbol or string specifying the SKILL function that is called when the menu is closed either by clicking the `Done` button or by calling `hiFixedMenuDown`. If you specify a symbol, the function is passed one argument (*r_hiMenu*); if you specify a string, the string is evaluated and the function is not passed any argument.

If you do not want to use this argument but want to use the next argument, specify `'none` as the value of this argument.

s_showTextWithIcon

Specifies whether both text and icons should be displayed for all menu items for which both `?itemText` and `?itemIcon` have been set. Legal values are `'under` (the text is displayed below the icon), `'beside` (the text is displayed beside the icon) and `'none` (if both `?itemText` and `?itemIcon` have been set for a menu item, only the icon is displayed). The default value is `'none`.

Value Returned

r_hiMenu The SKILL structure representing the menu. The *menuHandle* can be referenced by *menu->hiMenuSym*.

Examples

Creates a horizontal fixed menu with items `trMenuItemOne` and `trMenuItemOne`.

```
hiCreateHorizontalFixedMenu (
  'trExampleHorizontalFixedMenu
  list ( trMenuItemOne trMenuItemTwo )
  1
  2
)
```

If you need to create a series of menu items for the fixed menu, it is more efficient to create a symbol rather than creating each menu item individually.

```
Procedure ( trCreateMenuItem( theMenuSymbol )
  set(
    theMenuSymbol
    hiCreateMenuItem(
      ?name theMenuSymbol
      ?itemText get_pname( theMenuSymbol )
      ?callback sprintf ( nil "println ( '%L )" theMenuSymbol )
    )
  )
)
```

Cadence User Interface SKILL Reference

Menu Functions

```
) ; set
) ; procedure

trCreateMenuItem ( 'item1 )
trCreateMenuItem ( 'item2 )
trCreateMenuItem ( 'item3 )
trCreateMenuItem ( 'item4 )
trCreateMenuItem ( 'item5 )
trCreateMenuItem ( 'item6 )

hiCreateHorizontalFixedMenu(
    'trExampleHorizontalFixedMenu
    list( item1 item2 item3 item4 item5 item6 )
    1      ;;; number of rows
    6      ;;; number of columns
)

hiDisplayFixedMenu(
    trExampleHorizontalFixedMenu
    "top"
)

hiCreateVerticalFixedMenu(
    'trExampleVerticalFixedMenu
    list ( item1 item2 item3 item4 item5 item6 )
    6      ;;; number of rows
    1      ;;; number of columns
)

hiDisplayFixedMenu(
    trExampleVerticalFixedMenu
    "left"
)
```

Related Topics

Menu Functions

hiCreateMenuItem

hiCreateMenu

```
hiCreateMenu(  
    s_menuHandle  
    t_menuTitle  
    l_menuItems  
)  
=> r_hiMenu
```

Description

Returns the SKILL representation of a popup menu. Once this menu is created, you can display it with `hiDisplayMenu` or associate it with a window using `hiSetWindowMenu`.

A popup menu is a tear-off menu by default, that is, it can be torn off and displayed in its own window anywhere on the screen.

Popup menus can also be created with [hiCreate2DMenu](#) and [hiCreateSimpleMenu](#).

Arguments

<i>s_menuHandle</i>	A unique global SKILL symbol used to reference this menu.
<i>t_menuTitle</i>	Title that is displayed in the menu bar. To set the access key for the menu, add an ampersand before the desired letter. For example, if the menu title is <code>Edit</code> and you want <code>E</code> to be the access key for the menu, you would specify this argument as <code>"&Edit"</code> .
<i>l_menuItems</i>	List of menu items, each created by <code>hiCreateMenuItem</code> , <code>hiCreateSliderMenuItem</code> , hiCreateSeparatorMenuItem , or <code>hiCreateAction</code> .

Values Returned

<i>r_hiMenu</i>	The SKILL structure representing the menu. The <i>menuHandle</i> can be referenced by <code>menu->hiMenuSym</code> .

Related Topics

[Menu Functions](#)

Cadence User Interface SKILL Reference

Menu Functions

hiDisplayMenu

hiSetWindowMenu

hiCreateMenuItem

hiCreateSliderMenuItem

hiCreateSeparatorMenuItem

hiCreateMenuItem

```
hiCreateMenuItem(  
    ?name s_itemHandle  
    ?itemText t_menuItemText  
    [ ?itemIcon g_menuIcon ]  
    [ ?callback t_itemCallback ]  
    [ ?disable g_disabled ]  
    [ ?statusTip t_statusTip ]  
    [ ?font t_font ]  
)  
=> r_hiMenuItem
```

Description

Creates a menu item that can be used in any type of menu, except two-dimensional menus and simple menus.

The menu item should contain text *t_menuItemText* and optionally an icon *g_menuIcon* to be displayed adjacent to the text. Also, the menu item can have a SKILL callback procedure associated with its selection (*t_itemCallback*).

If an action was created with a callback of the same name as the callback for this function, and if an icon was specified for the action, then that icon is used for this menu item. Otherwise, the icon you specify is used. The menu item can also have an access key or shortcut. Pressing this key selects the menu item.

You can use the same menu item in multiple menus.

The menu item created with this function is a non-slider entry, that is, it cannot display a submenu. Use `hiCreateSliderMenuItem` to create slider items and use `hiCreateSeparatorMenuItem` to create separators.

Arguments

?name s_itemHandle

SKILL handle to the menu item.

?itemText t_menuItemText

Cadence User Interface SKILL Reference

Menu Functions

String representing text for the menu item.

To set an access key for the menu item, add an ampersand (&) before the desired letter. For example, if the menu item is `Replace` and you want `P` to be the access key, you would specify this argument as `"Re&place"`.

In pulldown and popup menus, an icon is displayed automatically with the text menu item if `t_menuItemText` begins with one of the following strings:

`"Load"`

`"Open"`

`"Close"`

`"Save As"`

`"Undo"`

`"Redo"`

`"Exit"`

`"Print"`

If you do not want to display the default icon in the menu, add the following variable to your `.cdsenv` file:

```
ui useAutoPixmaps boolean nil
```

`?itemIcon g_menuIcon`

List representing an icon menu item. The icon occupies the icon area on the left of the menu item, and the text (specified in `t_menuItemText`) is displayed alongside. Menu icons can only be 16x16 pixels in size. If the icon is larger than 16x16 pixels, it gets shrunk (which is why only 16x16 icons are recommended for the menu items).

If an action was created with a callback of the same name as the callback for this function, and if an icon was specified for the action, then that icon is used for this menu item. Otherwise, the icon you specify for this argument is used.

`?callback t_itemCallback`

String that represents a SKILL callback procedure to be executed when that menu item is selected.

`?disable g_disabled`

Cadence User Interface SKILL Reference

Menu Functions

Disables the menu item, grayed out and unselectable, when displayed.

`?statusTip t_statusTip`

The status text is displayed in the status bar at the bottom of the window when the mouse hovers over the menu item. If you do not specify this argument, the item text (`t_menuItemText`) is displayed.

To set the status tip after the menu is instantiated, you can use the [`hiSetMenuItemStatusTip`](#) function.

`?font t_font`

This argument is ignored.

Value Returned

`r_hiMenuItem`

The SKILL structure representing the menu item. The item handle and text menu item can be referenced by `menuItem->hiMenuItemSym` and `menuItem->hiItemText` respectively.

Examples

Creates two menu items as One and Two.

```
trMenuItemOne = hiCreateMenuItem(  
    ?name 'trMenuItemOne  
    ?itemText "One"  
    ?callback "println( \"One\" )" )  
)  
  
trMenuItemTwo = hiCreateMenuItem(  
    ?name 'trMenuItemTwo  
    ?itemText "Two"  
    ?callback "println ( \"Two\" )" )  
)
```

Adds a new entry, *Library Browser*, to the CIW. It looks for the *Library Manager* entry in all the menus of the CIW, and adds an entry for the Library Browser immediately after it.

```
let( ((ciwBannerMenus hiGetBannerMenus(window(1))) toolMenu  
    pItem menuItem found libBrowser)  
    while(ciwBannerMenus  
        pItem = eval(car(ciwBannerMenus))  
        ciwBannerMenus = cdr(ciwBannerMenus)
```

Cadence User Interface SKILL Reference

Menu Functions

```
menuList = pdItem ->_menuItemList
while(menuList
  menuItem = car(menuList)
  menuList = cdr(menuList)
  when(menuItem == 'LibMan
    found = t
    menuList = nil
    ciwBannerMenus = nil
  )
)
when( found
  libBrowser = hiCreateMenuItem(
    ?name      'LibBrowser
    ?itemText   "Library Browser..."
    ?callback   "dmbOpenLibDAGBrowser()"
  )
  hiInsertMenuItem(pdItem libBrowser 'LibMan)
)
```

Related Topics

Menu Functions

hiAddMenuItem

hiCreateSeparatorMenuItem

hiCreateSliderMenuItem

hiDeleteMenuItem

hiDisableMenuItem

hiEnableMenuItem

hiGetMenuItems

hiCreatePulldownMenu

```
hiCreatePulldownMenu(  
    s_menuHandle  
    g_menuTitle  
    l_menuItems  
)  
=> r_hiMenu
```

Description

Creates a pulldown menu that contains text, icon, or slider menu items. *s_menuHandle* is the SKILL symbol set to the created menu.

After you create a pulldown menu, you need to add it to the menu bar of a window using `hiInsertBannerMenu`, or insert it as a submenu using `hiCreateSliderMenuItem`. You can share pulldown menus by placing the same menu in different windows.

A pulldown menu is a tear-off menu by default, it can be detached from the menu bar or the main menu and displayed in its own window anywhere on the screen.

Arguments

<i>s_menuHandle</i>	A unique global SKILL symbol used to reference this menu.
<i>g_menuTitle</i>	Title that is displayed in the menu bar. This can be either a text string or an icon. If you specify a text string, to set the access key for the menu, add an ampersand before the desired letter in the menu title. For example, if the menu title is <code>Edit</code> and you want <code>E</code> to be the access key for the menu, you would specify this argument as <code>"&Edit"</code> .
<i>l_menuItems</i>	List of menu items created by <code>hiCreateMenuItem</code> , <code>hiCreateSliderMenuItem</code> , <code>hiCreateSeparatorMenuItem</code> , or <code>hiCreateAction</code> .

Cadence User Interface SKILL Reference

Menu Functions

Value Returned

r_hiMenu The SKILL structure representing the menu. The *menuHandle* can be referenced by `menu->hiMenuSym`.

Examples

Creates a pulldown menu named Navigation and inserts it in the leftmost position of the CIW menu banner. If you select the Navigation-Raise Current Window, the current window is raised to the top of the screen.

```
hiCreatePulldownMenu(
    'trPulldownMenu
    "Example Menu"
    list ( trMenuItemOne trMenuItemTwo )
)

;;; Creating a new window to be used as the current window
hiOpenWindow()

;;; creating the "Raise Current Window" menu item
trRaiseCurrentWindow = hiCreateMenuItem(
    ?name      'trRaiseCurrentWindow
    ?itemText   "Raise Current Window"
    ?callback   "hiRaiseWindow( hiGetCurrentWindow() )"
)

;;; creating the Navigation pulldown menu
hiCreatePulldownMenu(
    'trPulldownMenu      ;;; menu handle
    "Navigation"          ;;; menu title
    list( trRaiseCurrentWindow)
    ""                    ;;; empty help string
) ; hiCreatePulldownMenu

;;; inserting the pulldown menu in the CIW
hiInsertBannerMenu(
    window( 1 )
    trPulldownMenu
    0
)
```

To remove the Navigation pulldown menu from the CIW menu banner use the following command:

```
hiDeleteBannerMenu( window(1) 0 )
```

Creates a pulldown menu named Example Menu. The first menu item of this pulldown menu is a slider menu item that prints the letters A, B, or C when selected. The second menu item resizes the CIW.

```
;;; creating menu items for the slider menu
trA MenuItem = hiCreateMenuItem(
    ?name 'trA MenuItem
    ?itemText "A"
```

Cadence User Interface SKILL Reference

Menu Functions

```
?callback "println('A)" ;;; prints A in the CIW
)

trB_MenuItem = hiCreateMenuItem(
    ?name 'trB_MenuItem
    ?itemText "B"
    ?callback "println('B)" ;;; prints B in the CIW
)

trC_MenuItem = hiCreateMenuItem(
    ?name 'trC_MenuItem
    ?itemText "C"
    ?callback "println('C)" ;;; prints C in the CIW
)

;;; building the slider menu
hiCreatePulldownMenu(
    'trSubMenu
    ""
    list( ; the list of menu items in the slider menu
        trA_MenuItem
        trB_MenuItem
        trC_MenuItem
    )
)

;;; creating the first menu item for the pulldown menu
trSliderMenuItem = hiCreateSliderMenuItem(
    ?name 'trSliderMenuItem
    ?itemText "Print"
    ?subMenu trSubMenu
)

;;; creating the second menu item for the pulldown menu
trMenuItemResize = hiCreateMenuItem(
    ?name 'trMenuItemResize
    ?itemText "Resize"
    ?callback "hiResizeWindow( window(1) list(0:0 500:500))"
)

;;; creating the Example Menu pulldown menu
hiCreatePulldownMenu(
    'trPulldownMenu
    "Example Menu"
    list( trSliderMenuItem trMenuItemResize )
)

;;; inserting the pulldown menu in the CIW
hiInsertBannerMenu( window(1) trPulldownMenu 0 )
```

To test the sample menu, from the Example Menu pulldown menu, select *Print*. Selecting each letter in the slider menu displays it in the CIW. Selecting the *Resize* menu item from the pulldown menu resizes the CIW.

To delete the Example Menu:

```
hiDeleteBannerMenu( window(1) 0 )
```

Cadence User Interface SKILL Reference

Menu Functions

Related Topics

[Menu Functions](#)

[hiInsertBannerMenu](#)

[hiInsertBannerMenu](#)

[hiCreateSliderMenuItem](#)

hiCreateSimpleMenu

```
hiCreateSimpleMenu(  
    s_menuHandle  
    t_menuTitle  
    l_menuItems  
    l_menuItemCallbacks  
)  
=> r_hiMenu
```

Description

Returns the SKILL description of a simple menu, which is a single level menu containing only text menu items, none of which invoke a submenu. Once this menu is created, you can display it with `hiDisplayMenu` or associate it with a window using `hiSetWindowMenu`.

Arguments

<i>s_menuHandle</i>	A unique global SKILL symbol used to reference this menu.
<i>t_menuTitle</i>	Title that is displayed in the menu bar. To set the access key for the menu, add an ampersand before the desired letter. For example, if the menu title is <code>Edit</code> and you want <code>E</code> to be the access key for the menu, you would specify this argument as <code>"&Edit"</code> .
<i>l_menuItems</i>	List of strings that are the menu items for the simple menu. To set access keys for the menu items, add an ampersand before the desired letter in each menu item name. For example, if you want <code>P</code> to be the access key for the menu item <code>Replace</code> , you would specify the menu item as <code>"Re&place"</code> .
<i>l_menuItemCallbacks</i>	List of string specifications of SKILL callback procedures that are called when the menu item is selected. There should be one callback string listed for every menu item. If only one callback string is specified, that routine is called back regardless of which menu item is selected. In this case, the text of the selected menu item gets appended to the callback string.

Cadence User Interface SKILL Reference

Menu Functions

Values Returned

r_hiMenu The SKILL structure representing the menu. The *menuHandle* can be referenced by *menu->hiMenuSym*.

Examples

The following sample first creates a popup menu that lists the Virtuoso windows currently accessible on the desktop, and then sets the F6 function key to display this popup menu.

```
hiCreateSimpleMenu(
    'trExampleMenu
    "Example Menu"
    ' ( "One" "Two" )
    ' ( "println( \"One\" )" "println( \"Two\" )" ) )
=> array[7]:27015368

procedure( trDisplayWindowsPopUp()
    hiDisplayMenu(
        ;;; creating a dynamic popup menu
        hiCreateSimpleMenu(
            'trWindowsPopUp
            "Windows"
            foreach( mapcar wid hiGetWindowList()
                hiSetWindowName( wid )
            ) ; foreach
            foreach( mapcar wid hiSetWindowName()
                sprintf(
                    nil
                    "hiRaiseWindow( window( %d )"
                    wid->windowNum
                )
            ) ; foreach
        ) ; hiCreateSimpleMenu
    ) ; hiDisplayMenu
) ; procedure

;;; Setting the F6 function key to display the popup menu
hiSetBindKey( "Command Interpreter" "<Key>F6"
    "trDisplayWindowsPopUp()" )
```

Related Topics

Menu Functions

hiDisplayMenu

hiSetWindowMenu

hiCreateSeparatorMenuItem

```
hiCreateSeparatorMenuItem(  
    ?name s_itemHandle  
)  
=> r_hiSeparatorMenuItem
```

Description

Creates a menu separator—a horizontal line that separates menu items. This menu item is used in the *l_menuItems* argument of the `hiCreateMenu` and `hiCreatePulldownMenu` functions. You can use the separator menu item in more than one menu or in multiple locations in the same menu.

Argument

<i>s_itemHandle</i>	SKILL handle to the menu item.

Values Returned

<i>r_hiSeparatorMenuItem</i>	The SKILL structure representing the separator menu item.

Related Topics

[Menu Functions](#)

[hiDisplayMenu](#)

[hiCreate2DMenu](#)

[hiCreateSimpleMenu](#)

[hiCreateMenuItem](#)

[hiCreateSliderMenuItem](#)

hiCreateSliderMenuItem

```
hiCreateSliderMenuItem(  
    ?name s_itemHandle  
    ?itemText t_menuItemText  
    ?subMenu r_pulldownMenu  
    [ ?itemIcon l_menuIcon ]  
    [ ?disable g_disabled ]  
    [ ?statusTip t_statusTip ]  
    [ ?font t_font ]  
)  
=> r_hiSliderItem
```

Description

Creates a slider menu item that is used in functions that create a menu, all menus except simple or 2D menus. Slider menu items bring up submenus when selected. They can contain text (*t_menuItemText*) or icon (*l_menuIcon*) information.

A slider menu entry is a composite object. You must create all its subobjects before making your slider menu entries. Call `hiCreateMenuItem` or `hiCreateSliderMenuItem` to create your pulldown menu items and then `hiCreatePulldownMenu` to generate your slider menu entry.

You can use slider menu items within a popup or pulldown menu. You can also share them by placing the same menu item within different menus.

Arguments

?name s_itemHandle

SKILL handle to the menu item.

?itemText t_menuItemText

String representing a text menu item.

To set the access key for the menu item, add an ampersand before the desired letter. For example, if the menu item is `Replace` and you want `P` to be its access key, you would specify this argument as `"Re&place"`.

?subMenu r_pulldownMenu

Cadence User Interface SKILL Reference

Menu Functions

Unlike `hiCreateMenuItem`, a user-defined callback procedure is not associated with slider menu items but with the menu items in `r_pulldownMenu`. `r_pulldownMenu` is a menu created with `hiCreatePulldownMenu` and displays when this menu item is selected.

?itemIcon *l_menuIcon*

List representing an icon menu item.

?disable *g_disabled*

Disables the slider menu item when displayed.

?statusTip *t_statusTip*

The status text that is displayed in the status bar at the bottom of the window when the mouse hovers over the menu item. If you do not specify this argument, the menu item text (*t_menuItemText*) is displayed.

?font *t_font*

This argument is ignored.

Value Returned

r_hiSliderItem

The SKILL structure representing the menu item. The *itemHandle*, *menuItemText*, and *pulldownMenu* can be referenced by:

`menuItem->hiMenuItemSym`

`menuItem->hiItemText`

`menuItem->hiSubMenu`

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateSimpleMenu](#)

[hiCreateMenuItem](#)

hiCreateTypeinMenuItem

```
hiCreateTypeinMenuItem(  
    ?name s_name  
    [ ?itemIcon g_icon ]  
    [ ?itemText t_text ]  
    [ ?callback S_funcname ]  
    [ ?disable g_disable ]  
    [ ?statusTip t_statusTip ]  
    [ ?toolTip t_toolTip ]  
    [ ?completerType s_type ]  
    [ ?completerList l_list ]  
)  
=> r_menuItem / nil
```

Description

Creates a type-in menu item that can receive user inputs. When a user types text in the type-in menu item created using this function and presses `Enter`, the specified callback function *S_funcname* is run.

Arguments

<code>?name <i>s_name</i></code>	SKILL handle to the menu item.
<code>?itemIcon <i>g_icon</i></code>	The icon associated with the menu item.
<code>?itemText <i>t_text</i></code>	The placeholder text set to prompt user input in the type-in menu item. This text is displayed when the menu item is not in focus.
<code>?callback <i>S_funcname</i></code>	The callback function that is run when user accesses the menu item and presses <code>Enter</code> .
<code>?disable <i>g_disable</i></code>	Disables the menu item (grayed out and unselectable) when displayed.
<code>?statusTip <i>t_statusTip</i></code>	The status text is displayed in the status bar at the bottom of the window when the menu item is in focus. If you do not specify this argument, the menu item text <i>t_text</i> is displayed.
<code>?toolTip <i>t_toolTip</i></code>	

Cadence User Interface SKILL Reference

Menu Functions

Displays tool tip when the mouse hovers over the menu item. If you do not specify this argument, the menu item text *t_text* is displayed.

?completerType *s_type*

The setting to assist user input in the type-in menu item. This argument can have one of the following values:

- *inline*: The history of user inputs is displayed in line.
- *popup*: The history of user inputs appears in a popup list.
- *none*: The history of user inputs is not shown.

?completerList *l_list*

The list of strings that store the history of the previous user inputs. This list is used to assist user inputs in the type-in menu item. It is updated when the user types text in the type-in menu item and presses *Enter*. The latest user inputs are placed in the beginning of the history list and older inputs are moved below in the list.

It is possible to set or replace this list of strings with custom strings using *hiSetTypeinMenuItemCompleterList*.

All instances of the same type-in menu item share the same history list.

Value Returned

r_menuItem

The SKILL structure representing the type-in menu item.

nil

The menu item could not be created.

Example

The following code creates a type-in menu item with a pre-seeded completer list and a popup completer. This code can be specified in the list of menu items, for example, *hiCreatePulldownMenu*.

```
PopupTypein = hiCreateTypeinMenuItem( ?name 'PopupCompleter ?itemText "Has Popup"
                                       ?itemIcon "search.png" ?callback "println"
                                       ?completerType 'popup
                                       ?completerList '( "Apple" "Orange" "Grape"
"Grapefruit"
```

Cadence User Interface SKILL Reference

Menu Functions

```
"Guava" "Stawberry" "Watermelon" )  
?toolTip "This has a popup completer" )
```

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiSetTypeinMenuItemCompleterList](#)

[hiCreateMenuItem](#)

[hiAddMenuItem](#)

[hiCreateSeparatorMenuItem](#)

hiSetTypeinMenuItemCompleterList

```
hiSetTypeinMenuItemCompleterList(  
    r_menu  
    s_menuItem  
    l_completerList  
)  
=> t / nil
```

Description

Replaces the list of strings that store the history of user inputs for the specified type-in menu item. This list is used to assist user inputs in the type-in menu item.

Arguments

<i>r_menu</i>	The popup or pull-down menu containing the type-in menu item.
<i>s_menuItem</i>	The symbol name of the type-in menu item.
<i>l_completerList</i>	The list of strings that must replace the current history list of the type-in menu item used to assist users to input data.

Values Returned

<i>t</i>	The list of strings was replaced successfully.
<i>nil</i>	The list of strings could not be replaced.

Examples

Consider that the menu `MyMenu` contains the type-in menu item `MyTypein` and the `completerType` argument was not set to `none` when this menu item was created using `hiCreateTypeinMenuItem`. To replace the list of strings that assist users to input data in `MyTypein` with a custom list, use the following command:

```
hiSetTypeinMenuItemCompleterList( MyMenu 'MyTypein' ("One" "Two" "Three" "Four" ) )
```

Related Topics

Menu Functions

Cadence User Interface SKILL Reference

Menu Functions

hiCreateTypeinMenuItem

hiCreateMenuItem

hiAddMenuItem

hiCreateSeparatorMenuItem

hiDeleteMenuItem

hiEnableMenuItem

hiCreateVerticalFixedMenu

```
hiCreateVerticalFixedMenu(  
    s_menuHandle  
    l_menuItems  
    x_rows  
    x_cols  
    [ S_doneCallback ]  
    [ s_showTextWithIcon ]  
)  
=> r_hiMenu
```

Description

Creates a vertical fixed menu that can be displayed in the root window. Fixed menus can no longer be displayed in application windows; use the new toolbar functions to create toolbars for windows instead.

The fixed menu is created as a grid of *row* x *column* dimensions, and contains menu item objects created by `hiCreateMenuItem`. An additional *Done* item is added as the last menu item. Selecting *Done* removes the fixed menu from the screen.

The width and height of the vertical fixed menu is determined by the number of rows and columns in the menu. Menu items you want to display within the fixed menu are inserted in row-major order.

Vertical fixed menus appear along the side of the root window (the screen). These fixed menus are shared among all Cadence application windows contained within the root window. To display a fixed menu, use `hiDisplayFixedMenu`.

Fixed menus can no longer be displayed in windows. Use the new toolbar functions or toolbar files to create toolbars for windows instead. Existing fixed menus that are attached to windows placed in session windows are no longer displayed. Existing fixed menus attached to stand-alone windows are displayed but might have display problems. Therefore, you should replace all existing fixed menus attached to windows with toolbars.

Arguments

<i>s_menuHandle</i>	A unique global SKILL symbol used to reference this menu.
<i>l_menuItems</i>	List of menu items created by <code>hiCreateMenuItem</code> .
<i>x_rows</i>	Number of rows in the fixed menu.
<i>x_cols</i>	Number of columns in the fixed menu.

Cadence User Interface SKILL Reference

Menu Functions

S_doneCallback Symbol or string specifying the SKILL function that is called when the menu is closed either by clicking the `Done` button or by calling `hiFixedMenuDown`. If you specify a symbol, the function is passed one argument (*r_hiMenu*); if you specify a string, the string is evaluated and the function is not passed any argument.

If you do not want to use this argument but want to use the next argument, specify `'none` as the value of this argument.

s_showTextWithIcon

Specifies whether both text and icons should be displayed for all menu items for which both `?itemText` and `?itemIcon` have been set. Legal values are `'under` (the text is displayed below the icon), `'beside` (the text is displayed beside the icon) and `'none` (if both `?itemText` and `?itemIcon` have been set for a menu item, only the icon is displayed). The default value is `'none`.

Value Returned

r_hiMenu The SKILL structure representing the menu. The *menuHandle* can be referenced by *menu->hiMenuSym*.

Examples

Creates a vertical fixed menu, displays it, and then attaches it to a window. In this example, the `trCreateMenuItem` procedure allows an easy means of generating menu items with a similar callback.

```
;;; The following procedure returns a menu item
;;; based on theMenuSymbol.
procedure( trCreateMenuItem( theMenuSymbol )
  ;;; The set function assigns the menu item data structure
  ;;; to the theMenuSymbol's value.
  set(
    theMenuSymbol
    hiCreateMenuItem(
      ?name      theMenuSymbol
      ;;; get_pname returns the name of the symbol
      ;;; as a text string.
      ?itemText  get_pname( theMenuSymbol )
      ;;; All menu items will have a similar callback
      ;;; that prints the menu item's ?name variable.
      ?callback  sprintf( nil "println( '%L )" theMenuSymbol )
    )
  )
```

Cadence User Interface SKILL Reference

Menu Functions

```
    ) ; set
) ; procedure

;;; creating the menu items for the fixed menu
trCreateMenuItem( 'item1 )
trCreateMenuItem( 'item2 )
trCreateMenuItem( 'item3 )

;;; creating the vertical fixed menu
hiCreateVerticalFixedMenu(
    'trExampleVerticalFixedMenu
    list( item1 item2 item3 )
    3          ;;; number of rows
    1          ;;; number of columns
)
```

To display the sample fixed menu, use the following function:

```
hiDisplayFixedMenu(
    trExampleVerticalFixedMenu
    "left"          ;;; placement must be one of "top", "bottom",
                    ;;; "right", or "left"
)
```

After the fixed menu is displayed, press the menu items. Each menu item displays its name in the CIW.

You can add a vertical fixed menu to a window of type `graphics`, `browser`, `form`, `text`, or `encap`. The following example opens a text file named `fixedmenu.sample` and attaches the `trExampleVerticalFixedMenu` to this text window. You can open a different window and replace its `wid` with `window(3)` in the following example.

```
view("~/test/fixedmenu.sample")
;;; The file is displayed as window(3).
```

```
hiAddFixedMenu( ?window window(3) ?fixedMenu trExampleVerticalFixedMenu)
```

After the fixed menu is attached to the window, select the menu items. Each menu item displays its name in the CIW.

To remove `trExampleVerticalFixedMenu` from `window(3)`, use the following command:

```
hiRemoveFixedMenu( window(3) )
```

Related Topics

Menu Functions

hiDisplayFixedMenu

Cadence User Interface SKILL Reference

Menu Functions

hiDeleteMenu

```
hiDeleteMenu(  
    r_hiMenu  
)  
=> t / nil
```

Description

Destroys the menu given by *hiMenu*, which must be a valid data structure. Individual menu items contained in this menu are also destroyed.

Argument

r_hiMenu

The menu returned by any of the menu creation routines, except for `hiCreatePulldownMenu`.

To delete window fixed menus, first use `hiRemoveFixedMenu` to remove the fixed menu from the window, and then use `hiDeleteMenu` to destroy the fixed menu.

Values Returned

t

The menu is deleted.

nil

The menu is not deleted. The SKILL handle referencing this menu is reset to *nil*. An error message is also issued.

Related Topics

[Menu Functions](#)

[hiDeleteBannerMenu](#)

[hiRemoveFixedMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiDeleteMenuItem

```
hiDeleteMenuItem(  
    r_hiMenu  
    s_itemHandle  
)  
=> t / nil
```

Description

Deletes a menu item or action from all instances of the specified *r_hiMenu*.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> or <code>hiCreatePulldownMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> , <code>hiCreateSliderMenuItem</code> or <code>hiCreateSeparatorMenuItem</code> , or when the action was created with <code>hiCreateAction</code> .

Values Returned

t	The menu item is deleted.
nil	The menu item is not deleted. An error message is also issued.

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateMenu](#)

[hiCreateMenuItem](#)

[hiCreateSliderMenuItem](#)

[hiCreateSeparatorMenuItem](#)

Cadence User Interface SKILL Reference

Menu Functions

hiDisableMenuItem

```
hiDisableMenuItem(  
    r_hiMenu  
    s_itemHandle  
    [ w_windowId ]  
)  
=> t / nil
```

Description

Grays out a menu item on pull-down, pop-up, or fixed menus, making it unselectable.

Unless otherwise specified, all menu items are selectable when they are created.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> , <code>hiCreatePulldownMenu</code> , <code>hiCreateHorizontalFixedMenu</code> , or <code>hiCreateVerticalFixedMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> , or when the action was created with <code>hiCreateAction</code> .
<i>w_windowId</i>	Window argument created with <code>hiCreateWindow</code> or <code>hiOpenWindow</code> . If this argument is used, only the menu item in a menu on this window is disabled. This is useful for shared menus, when you want to disable a menu item in one menu only and leave the other occurrences of the same menu item enabled. This argument is also necessary when disabling a menu item on a window fixed menu.

Values Returned

<i>t</i>	The menu items are disabled.
<i>nil</i>	The menu items are not disabled. An error message is also issued.

Cadence User Interface SKILL Reference

Menu Functions

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateMenu](#)

[hiCreateMenuItem](#)

[hiCreateSliderMenuItem](#)

[hiCreateSeparatorMenuItem](#)

hiDisplayFixedMenu

```
hiDisplayFixedMenu(  
    r_hiMenu  
    g_menuLocation  
    [ x_width ]  
    [ x_height ]  
    [ g_staysOnTop ]  
)  
=> t / nil
```

Description

Displays the specified fixed menu in the default root window (screen). The menu appears on a side of the screen, to be shared among Cadence application windows contained within the screen. The *x_width* and *y_height* arguments are ignored and currently exist only for compatibility reasons. *x_width* and *y_height* of the menu are automatically determined by the size of the menu items and the number of columns and rows.

Arguments

<i>r_hiMenu</i>	The menu returned from <code>hiCreateHorizontalFixedMenu</code> or <code>hiCreateVerticalFixedMenu</code> or an Edge-styled fixed menu created with the <i>menutrans</i> program.

Cadence User Interface SKILL Reference

Menu Functions

<i>g_menuLocation</i>	This argument must be one of the following, which aligns the fixed menu along the edge of the screen: "top" "bottom" "left" "right" "topLeft" "topRight" "bottomLeft" "bottomRight" The "top" and "bottom" are allowed only for horizontal fixed menus and "left" and "right" for vertical fixed menus. The options "topLeft", "topRight", "bottomLeft" and "bottomRight" place the menus in the corners. For the corner options, either horizontal or vertical menus are allowed.
<i>x_width</i>	Specifies the width of the menu. However, even if specified this value is ignored and currently exists only for compatability reasons.
<i>x_height</i>	Specifies the height of the menu. However, even if specified this value is ignored and currently exists only for compatability reasons.
<i>g_staysOnTop</i>	When set to <code>t</code>, the vertical or horizontal fixed menus stay on top of other windows.

Values Returned

<code>t</code>	The menu is displayed.
<code>nil</code>	The menu is not displayed. An error message is also issued.

Examples

The following example displays `trExampleHorizontalFixedMenu`. Before using this example, see the `hiCreateHorizontalFixedMenu` example to find out how to create `trExampleHorizontalFixedMenu`.

Cadence User Interface SKILL Reference

Menu Functions

```
hiDisplayFixedMenu(  
    trExampleHorizontalFixedMenu  
    "top"  
)
```

Related Topics

[Menu Functions](#)

[hiCreateHorizontalFixedMenu](#)

hiDisplayMenu

```
hiDisplayMenu(  
    r_hiMenu  
    [ l_position ]  
)  
=> t / nil
```

Description

Displays a menu created by `hiCreateMenu`, `hiCreateSimpleMenu`, or `hiCreate2DMenu`. For non-2D menus, the menu is always displayed at the current cursor's position at the time `hiDisplayMenu` is called.

A call to `hiDisplayMenu()` for 2D menus immediately returns `t`. However, if you want that the call to `hiDisplayMenu()` should not return `t` until the menu is closed (to match the behavior of the previous release), set the menu property `dontBlock` to `nil` after creating the menu, as shown below:

```
MyMenu = hiCreate2DMenu( ... )  
MyMenu->dontBlock = nil
```

Arguments

<i>r_hiMenu</i>	The menu returned from a call to <code>hiCreateMenu</code> , <code>hiCreateSimpleMenu</code> or <code>hiCreate2DMenu</code> .
<i>l_position</i>	Optional x, y screen coordinates used for 2D menus. If specified, the 2D menus are displayed with its upper left corner at this position (assuming 0 : 0 to be the upper left corner of the screen). If not specified, the 2D menu is displayed at the location of the last mouse button click in the window or, if no previous button click has occurred in the window, the exact location is undefined.

Values Returned

<i>t</i>	The menu is displayed.
<i>nil</i>	The menu is not displayed. An error message is also issued.

Cadence User Interface SKILL Reference

Menu Functions

Examples

The following example uses the `hiSetBindKey` function to define a bindkey for the Schematics application. The bindkey displays the `trExampleMenu` pop-up menu. For an example of creating this pop-up menu, see `hiCreateSimpleMenu`.

```
hiSetBindKey( "Schematics"  
    "Shift Ctrl<Btn2Down> ( 2 )"   
    "hiDisplayMenu( trExampleMenu )" )
```

Related Topics

[Menu Functions](#)

[hiSetWindowMenu](#)

[hiCreate2DMenu](#)

[hiCreateSimpleMenu](#)

[hiSetBindKey](#)

[hiCreateMenuItem](#)

[hiCreateSliderMenuItem](#)

hiDisplayWindowMenu

```
hiDisplayWindowMenu(  
    w_windowId  
)  
=> t / nil
```

Description

Displays the popup menu associated with the given *w_windowId*, which must be a valid data structure returned by a call to `hiOpenWindow` or `hiCreateWindow`.

Argument

<i>w_windowId</i>	Window ID you specify. A menu can be associated with a window by calling <code>hiSetWindowMenu</code> . The menu is displayed at the current cursor's position at the time <code>hiDisplayWindowMenu</code> was called. The menu is brought down by a menu selection or by clicking outside the menu.

Values Returned

t	The menu is displayed.
nil	The menu is not displayed. An error message is also issued.

Related Topics

[Menu Functions](#)

[hiOpenWindow](#)

[hiDisplayWindowMenu](#)

[hiSetWindowMenu](#)

[hiCreate2DMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiEnableMenuItem

```
hiEnableMenuItem(  
    r_hiMenu  
    s_itemHandle  
    [ w_windowId ]  
)  
=> t / nil
```

Description

Makes unselectable menu items in pull-down, pop-up, or fixed menus selectable.

All menu items are selectable when created.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> , <code>hiCreatePulldownMenu</code> , <code>hiCreateHorizontalFixedMenu</code> , or <code>hiCreateVerticalFixedMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> , or when the action was created with <code>hiCreateAction</code> .
<i>w_windowId</i>	Window argument created with <code>hiCreateWindow</code> or <code>hiOpenWindow</code> . If this argument is used, only the menu item in a menu on this window is disabled. This is useful for shared menus, when you want to disable a menu item in one menu only and leave the other occurrences of the same menu item enabled. This argument is also necessary when disabling a menu item on a window fixed menu.

Values Returned

<i>t</i>	The menu items are disabled.
<i>nil</i>	The menu items are not disabled. An error message is also issued.

Cadence User Interface SKILL Reference

Menu Functions

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateMenu](#)

[hiCreateMenuItem](#)

[hiCreateSliderMenuItem](#)

[hiCreateSeparatorMenuItem](#)

Cadence User Interface SKILL Reference

Menu Functions

hiFixedMenuDown

```
hiFixedMenuDown(  
    r_hiMenu  
)  
=> t / nil
```

Description

Removes the specified fixed menu from the screen.

Argument

<i>r_hiMenu</i>	Specifies the menu to be removed

Values Returned

t	The menu is removed.
nil	The menu is not removed. An error message is also issued.

Related Topics

[Menu Functions](#)

hiGetMenuItems

```
hiGetMenuItems (  
    r_hiMenu  
)  
=> l_menuItems / nil
```

Description

Returns a list of the menu items in the specified menu.

Argument

<i>r_hiMenu</i>	The menu SKILL structure, returned by <code>hiCreateMenu</code> , <code>hiCreatePulldownMenu</code> , <code>hiCreateHorizontalFixedMenu</code> , or <code>hiCreateVerticalFixedMenu</code> .

Values Returned

<i>l_menuItems</i>	The list of menu items in the specified menu.
<i>nil</i>	There was an error, such as the following: <i>r_hiMenu</i> is not a menu structure or <i>r_hiMenu</i> is a 2D menu.

Examples

The following code gets the list of menu items from the first menu on the CIW:

```
hiGetMenuItems (eval (car (hiGetBannerMenus (hiGetCIWindow() ) ) ) )
```

Related Topics

[Menu Functions](#)

[hiCreateMenu](#)

[hiCreatePulldownMenu](#)

[hiCreateMenuItem](#)

[hiCreateSeparatorMenuItem](#)

Cadence User Interface SKILL Reference

Menu Functions

hiCreateSliderMenuitem

Cadence User Interface SKILL Reference

Menu Functions

hiGetWindowMenu

```
hiGetWindowMenu(  
    w_windowId  
)  
=> s_hiMenu / nil
```

Description

Returns the symbol of the menu associated with the specified window. This menu was associated with a window by using `hiSetWindowMenu`.

Argument

<code>w_windowId</code>	Window ID you specify.

Values Returned

<code>s_hiMenu</code>	The symbol of the menu associated with the window.
<code>nil</code>	There is no menu associated with the window.

Related Topics

[Menu Functions](#)

[hiSetWindowMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiInsertMenuItem

```
hiInsertMenuItem(  
    r_hiMenu  
    r_menuItem  
    g_position  
)  
=> t / nil
```

Description

Inserts a menu item or action into a specified position of all instances of *r_hiMenu*.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> or <code>hiCreatePulldownMenu</code> .
<i>r_menuItem</i>	Menu item description returned from with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> , or when the action was created with <code>hiCreateAction</code> .
<i>g_position</i>	Symbol or integer specifying the item to insert after. The only way to insert a menu item at the beginning of a menu is to specify position 0. An integer position that is larger than the number of menu items defaults to the end of the menu; a negative position defaults to 0. Any symbol specified must represent the menu item handle used when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> .

Values Returned

<i>t</i>	The menu item is inserted.
<i>nil</i>	The menu item is not inserted. An error message is also issued.

Related Topics

Menu Functions

hiCreatePulldownMenu

Cadence User Interface SKILL Reference

Menu Functions

hiCreateMenu

hiCreateMenuItem

hiCreateSliderMenuItem

hiCreateSeparatorMenuItem

Cadence User Interface SKILL Reference

Menu Functions

hIs2DMenu

```
hIs2DMenu(  
    g_menu  
)  
=> non-nil / nil
```

Description

Determines whether the specified argument is a valid 2D menu.

Argument

<i>g_menu</i>	Menu you specified.

Values Returned

<i>non-nil</i>	The specified menu is a valid 2D menu.
<i>nil</i>	The menu is not a 2D menu.

Related Topics

[Menu Functions](#)

Cadence User Interface SKILL Reference

Menu Functions

hiIsIcon

```
hiIsIcon(  
    g_icon  
)  
=> t / nil
```

Description

Determines whether the specified argument is a valid icon.

This function only checks whether the argument types are valid, not whether the icon itself is valid.

Argument

<i>g_icon</i>	The icon you want to check.

Values Returned

t	The specified icon is valid.
nil	The icon is not valid.

Related Topics

[Menu Functions](#)

Cadence User Interface SKILL Reference

Menu Functions

hiIsMenu

```
hiIsMenu(  
    g_menu  
)  
=> non-nil / nil
```

Description

Determines whether the specified argument is a valid menu.

Argument

<i>g_menu</i>	Specify the menu.

Values Returned

non-nil	The specified menu is a valid popup, pulldown, or fixed menu.
nil	The menu is not valid.

Related Topics

[Menu Functions](#)

hIsMenuItemEnabled

```
hIsMenuItemEnabled(  
    r_hiMenu  
    s_itemHandle  
    [ w_windowId ]  
)  
=> t / nil
```

Description

Determines whether a menu item is enabled.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> , <code>hiCreatePulldownMenu</code> , <code>hiCreateHorizontalFixedMenu</code> , or <code>hiCreateVerticalFixedMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> , or when the action was created with <code>hiCreateAction</code> .
<i>w_windowId</i>	Optional window argument. If specified, only the menu on this window is looked at. This is useful if the same menu has been placed on more than one menu. This argument is also necessary when disabling a menu item on a window fixed menu.

Values Returned

<i>t</i>	The specified menu item is enabled.
<i>nil</i>	The menu or menu item cannot be found, or the menu item is disabled.

Related Topics

Menu Functions

hiCreatePulldownMenu

Cadence User Interface SKILL Reference

Menu Functions

hiCreateMenu

hiCreateMenuItem

hiCreateSliderMenuItem

hiCreateSeparatorMenuItem

hiSetActionChecked

hiDisableMenuItem

hiEnableMenuItem

hiSetMenuItemCallback

```
hiSetMenuItemCallback(  
    r_hiMenu  
    s_itemHandle  
    t_callback  
)  
=> t / nil
```

Description

Changes the callback associated with all instances of a specified menu item.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> , <code>hiCreatePulldownMenu</code> , <code>hiCreateHorizontalFixedMenu</code> , or <code>hiCreateVerticalFixedMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> . <i>s_itemHandle</i> cannot be an action.
<i>t_callback</i>	Specifies the new SKILL procedure to run when this menu item is selected. Separate multiple procedures with a space.

Values Returned

t	The callback is changed.
nil	The callback is unchanged. An error message is also issued.

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiCreateMenuItem

hiCreateSliderMenuItem

hiCreateSeparatorMenuItem

hiDisableMenuItem

hiEnableMenuItem

hiSetMenuItemStatusTip

```
hiSetMenuItemStatusTip(  
    r_hiMenu  
    s_itemHandle  
    t_statusTip  
)  
=> t / nil
```

Description

Sets the status tip for the specified menu item. The status tip is displayed on the status bar at the bottom of the window when the mouse hovers over the menu item.

Arguments

<i>r_hiMenu</i>	Menu created with <code>hiCreateMenu</code> , <code>hiCreatePulldownMenu</code> , <code>hiCreateHorizontalFixedMenu</code> , or <code>hiCreateVerticalFixedMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> . <i>s_itemHandle</i> cannot be an action.
<i>t_statusTip</i>	Displays status tip when the mouse hovers over the menu item.

Values Returned

t	The status tip was set for the menu item.
nil	There was an error.

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiCreateMenuItem

hiCreateSliderMenuItem

hiCreateSeparatorMenuItem

hiDisableMenuItem

hiEnableMenuItem

Cadence User Interface SKILL Reference

Menu Functions

hiSetMenuItemText

```
hiSetMenuItemText(  
    r_hiMenu  
    s_itemHandle  
    t_itemText  
)  
=> t / nil
```

Description

Changes the text associated with all instances of a specified menu item.

Arguments

<i>r_hiMenu</i>	Created with <code>hiCreateMenu</code> or <code>hiCreatePulldownMenu</code> .
<i>s_itemHandle</i>	SKILL menu item handle specified when the menu item was created with <code>hiCreateMenuItem</code> or <code>hiCreateSliderMenuItem</code> . <i>s_itemHandle</i> cannot be an action.
<i>t_itemText</i>	Displays the new text specified in the menu item. To set the access key for the menu item, add an ampersand before the desired letter. For example, if the menu item is <code>Replace</code> and you want <code>P</code> to be the access key, you would specify this argument as <code>"Re&place"</code> .

Values Returned

<i>t</i>	The text is changed.
<i>nil</i>	The text is unchanged. An error message is also issued.

Related Topics

[Menu Functions](#)

[hiCreatePulldownMenu](#)

[hiCreateMenu](#)

Cadence User Interface SKILL Reference

Menu Functions

hiCreateMenuItem

hiCreateSliderMenuItem

hiCreateSeparatorMenuItem

hiDisableMenuItem

hiEnableMenuItem

Cadence User Interface SKILL Reference

Menu Functions

hiSetWindowMenu

```
hiSetWindowMenu(  
    g_menu  
    w_windowId  
)  
=> t / nil
```

Description

Associates the given menu with the specified window.

Arguments

g_menu

Menu must have been returned by a call to either `hiCreateSimpleMenu` or `hiCreateMenu`. If menu is `nil`, any previous menu gets disassociated with the window.

w_windowId

Window ID you specify. You should not associate a fixed menu with a window. Only one menu can be associated with a window at any given time. Multiple calls to this routine disassociate any previous menu that was set. After a menu has been associated with a window, the menu can be displayed with `hiDisplayWindowMenu`.

Values Returned

`t`

The menu is set.

`nil`

The menu is not set. An error message is also issued.

Related Topics

[Menu Functions](#)

[hiCreateMenu](#)

[hiCreateMenuItem](#)

[hiCreateSliderMenuItem](#)

[hiCreateSeparatorMenuItem](#)

Cadence User Interface SKILL Reference

Menu Functions

hiSetActionChecked

hiDisplayWindowMenu

Cadence User Interface SKILL Reference

Menu Functions

Toolbar Functions

This topic provides a list of Cadence® SKILL functions associated with toolbars.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [ddHiCreateCellToolbarComboBox](#)
- [ddHiCreateLibraryToolbarComboBox](#)
- [ddHiCreateViewToolbarComboBox](#)
- [hiCreateToolbar](#)
- [hiPlaceToolbar](#)
- [hiInsertToolbar](#)
- [hiShowToolbar](#)
- [hiSetToolbarObjName](#)
- [hiSetActionChecked](#)
- [hiCreateToolbarComboBox](#)
- [hiCreateToolbarTypein](#)
- [hiCreateToolbarSeparator](#)
- [hiDeleteToolbarBreak](#)
- [hiFocusToToolbarItem](#)
- [hiGetSharedIcon](#)
- [hiGetWindowToolbars](#)
- [hiHasToolbarBreak](#)
- [hiHideToolbar](#)

Cadence User Interface SKILL Reference

Toolbar Functions

- [hiInsertToolbar](#)
- [hiInsertToolbarBreak](#)
- [hiInsertToolbarItem](#)
- [hiInsertToolbarItems](#)
- [hiListFocusableToolbarItems](#)
- [hiPlaceToolbar](#)
- [hiPlaceToolbarBreak](#)
- [hiReattachToolbar](#)
- [hiShowToolbar](#)
- [hiSetToolbarObjName](#)
- [hiSetActionChecked](#)

ddHiCreateCellToolbarComboBox

```
ddHiCreateCellToolbarComboBox(  
    ?name s_name  
    [ ?prompt t_prompt ]  
    [ ?value t_currentValue ]  
    [ ?callback g_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?width x_width ]  
    [ ?statusTip t_statusTip ]  
    [ ?toolTip t_toolTip ]  
    [ ?validationType g_validationType ]  
    [ ?highlightType g_highlightType ]  
)  
=> o_comboBox
```

Description

Creates a customized Cell combo-toolbar item, which is an extension of a regular toolbar combo box item, for selecting cells.

Arguments

<code>?name <i>s_name</i></code>	A handle to this combo box item. To reference it, use <code>toolbarHandle-><name></code>
<code>?prompt <i>t_prompt</i></code>	A string specifying the prompt for the combo box. To reference it, use <code>toolbarHandle-><name>->prompt</code>
<code>?value <i>t_currentValue</i></code>	The initial and current values of this combo box. To reference it, use <code>toolbarHandle-><name>->value</code>
<code>?callback <i>g_callback</i></code>	

Cadence User Interface SKILL Reference

Toolbar Functions

A string, symbol, or function object type callback that is executed if the value of the combo box changes. If a new value is typed in, the callback is called when the combo box has “lost focus”. This happens for example, when you move the cursor outside the toolbar, or tab to another widget.

- If a new value is selected from the drop-down list, then the callback is called immediately, not when the combo box loses focus.
- If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that get executed when the value of the combo box changes.

`?editable g_editable`

Edits the combo box.

`?enabled g_enabled`

Specifies whether the combo box is enabled. The default value is `t`.

`?width x_width` A width of the combo box.

`?statusTip t_statusTip`

Displays the status text in the status bar at the bottom of the window when the pointer hovers over the combo box.

`?toolTip t_toolTip`

Displays the tool tip when the pointer hovers over the combo box.

`?validationType g_validationType`

Specifies the criteria that determines if the input is valid or not. Acceptable values (symbol) are:

- `allowAny`: No restriction on input type.
- `allowExisting` (default): Allow existing unfiltered data returned by (DDPI) generator.
- `allowNew`: Allow data other than the return value of (DDPI) generator.

`?highlightType g_highlightType`

Cadence User Interface SKILL Reference

Toolbar Functions

Specifies when and how a widget is to be highlighted.
Acceptable values (symbol) are:

- background
- highlight
- error
- highlightInvalidText (default)

The first three values have the same definitions as given for the *g_highlightType* argument of `hiHighlightField`.

Setting the property to `highlightInvalidText` turns on automatic highlighting, which highlights the widget when an invalid input is entered. The validity of the input is determined based on the setting of the `validationType` property.

Value Returned

o_comboBox The cell combo-toolbar item.

Examples

A Cell combo-toolbar item is created.

```
trCellItem = ddHiCreateCellToolbarComboBox(  
    ?name 'cellItem  
    ?prompt "Cell" )
```

Related Topics

[Toolbar Functions](#)

[hiHighlightField](#)

ddHiCreateLibraryToolBarComboBox

```
ddHiCreateLibraryToolBarComboBox(  
    ?name s_name  
    [ ?prompt t_prompt ]  
    [ ?value t_currentValue ]  
    [ ?callback g_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?width x_width ]  
    [ ?statusTip t_statusTip ]  
    [ ?toolTip t_toolTip ]  
    [ ?validationType g_validationType ]  
    [ ?highlightType g_highlightType ]  
)  
=> o_comboBox
```

Description

Creates a customized Library combo-toolbar item, which is an extension of a regular toolbar combo box item, for selecting libraries.

Arguments

<code>?name <i>s_name</i></code>	A handle to this combo box item. To reference it, use <code>toolbarHandle-><name></code>
<code>?prompt <i>t_prompt</i></code>	A string specifying the prompt for the combo box. To reference it, use <code>toolbarHandle-><name>->prompt</code>
<code>?value <i>t_currentValue</i></code>	The initial and current values of this combo box. To reference it, use <code>toolbarHandle-><name>->value</code>
<code>?callback <i>g_callback</i></code>	

Cadence User Interface SKILL Reference

Toolbar Functions

A string, symbol, or function object type callback that is executed if the value of the combo box changes. If a new value is typed in, the callback is called when the combo box has “lost focus”. This happens for example, when you move the pointer outside the toolbar, or tab to another widget.

If a new value is selected from the drop-down list, then the callback is called immediately, not when the combo box loses focus.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that get executed when the value of the combo box changes.

?editable *g_editable*

Edits the combo box.

?enabled *g_enabled*

Specifies whether the combo box is enabled. The default value is `t`.

?width *x_width*

The width of the combo box.

?statusTip *t_statusTip*

Displays the status text in the status bar at the bottom of the window when the pointer hovers over the combo box.

?toolTip *t_toolTip*

The tool tip that is displayed when the pointer hovers over the combo box.

?validationType *g_validationType*

Specifies the criteria that determines if the input is valid or not. Acceptable values (symbol) are:

`allowAny`: No restriction on input type.

`allowExisting` (default): Allow existing unfiltered data returned by (DDPI) generator.

`allowNew`: Allow data other than the return value of (DDPI) generator.

?highlightType *g_highlightType*

Cadence User Interface SKILL Reference

Toolbar Functions

Specifies when and how a widget is to be highlighted.
Acceptable values (symbol) are:

`background`

`highlight`

`error`

`highlightInvalidText` (default)

The first three values have the same definitions as given for the *g_highlightType* argument of `hiHighlightField`.

Setting the property to `highlightInvalidText` turns on automatic highlighting, which highlights the widget when an invalid input is entered. The validity of the input is determined based on the setting of the `validationType` property.

Value Returned

o_comboBox Returns the library combo-toolbar item.

Examples

A Library combo-toolbar item is created.

```
trLibraryItem = ddHiCreateLibraryToolbarComboBox(  
    ?name 'libItem  
    ?prompt "Library"  
)
```

Related Topics

[Toolbar Functions](#)

[hiHighlightField](#)

ddHiCreateViewToolBarComboBox

```
ddHiCreateViewToolBarComboBox(  
    ?name s_name  
    [ ?prompt t_prompt ]  
    [ ?value t_currentValue ]  
    [ ?callback g_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?width x_width ]  
    [ ?statusTip t_statusTip ]  
    [ ?toolTip t_toolTip ]  
    [ ?viewTypes l_viewTypes ]  
    [ ?validationType g_validationType ]  
    [ ?highlightType g_highlightType ]  
)  
=> o_comboBox
```

Description

Creates a customized View combo-toolbar item, which is an extension of a regular toolbar combo box item, for selecting views.

Arguments

<code>?name <i>s_name</i></code>	A handle to this combo box within a form. To reference it, use <code>toolbarHandle-><name></code>
<code>?prompt <i>t_prompt</i></code>	A string specifying the prompt for the combo box. To reference it, use <code>toolbarHandle-><name>->prompt</code>
<code>?value <i>t_currentValue</i></code>	The initial and current values of this combo box. To reference it, use <code>toolbarHandle-><name>->value</code>
<code>?callback <i>g_callback</i></code>	

Cadence User Interface SKILL Reference

Toolbar Functions

A string, symbol, or function object type callback that is executed if the value of the combo box changes. If a new value is typed in, the callback is called when the combo box has “lost focus”. This happens for example, when you move the pointer outside the toolbar, or tab to another widget.

If a new value is selected from the drop-down list, then the callback is called immediately, not when the combo box loses focus.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that get executed when the value of the combo box changes.

?editable *g_editable*

If set to `nil`, the combo box is read-only; setting *g_editable* to `t` makes the combo box editable.

?enabled *g_enabled*

Specifies whether the combo box is enabled. The default value is `t`.

?width *x_width*

The width of the combo box.

?statusTip *t_statusTip*

Displays the status text in the status bar at the bottom of the window when the pointer hovers over the combo box.

?toolTip *t_toolTip*

The tool tip that is displayed when the pointer hovers over the combo box.

?viewTypes *l_viewTypes*

A list of DFII view types (string). The list is used for view-type based filtering in the view combo-box widget. A view is made available for selection only if the view type associated with its master file is a member of the *l_viewTypes* list.

?validationType *g_validationType*

Cadence User Interface SKILL Reference

Toolbar Functions

Specifies the criteria that determines if the input is valid or not.
Acceptable values (symbol) are:

`allowAny`: No restriction on input type.

`allowExisting` (default): Allow existing unfiltered data returned by (DDPI) generator.

`allowNew`: Allow data other than the return value of (DDPI) generator.

`?highlightType` *g_highlightType*

Specifies when and how a widget is to be highlighted.
Acceptable values (symbol) are:

`background`

`highlight`

`error`

`highlightInvalidText` (default)

The first three values have the same definitions as given for the *g_highlightType* argument of `hiHighlightField`.

Setting the property to `highlightInvalidText` turns on automatic highlighting, which highlights the widget when an invalid input is entered. The validity of the input is determined based on the setting of the `validationType` property.

Value Returned

o_comboBox Returns the view combo-toolbar item.

Examples

A view combo-toolbar item is created.

```
trCellItem = ddHiCreateViewToolbarcomboBox(  
    ?name 'cellItem  
    ?prompt "Cell"  
    ?viewTypes ( list "schematicSymbol" "schematic" )  
)
```

Cadence User Interface SKILL Reference

Toolbar Functions

Related Topics

[Toolbar Functions](#)

[hiHighlightField](#)

[ddHiCreateViewComboField](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiAddToolbarItem

```
hiAddToolbarItem(  
    o_hiToolbar  
    o_hiToolbarItem  
)  
=> t / nil
```

Description

Adds one toolbar item to a toolbar. The item is added to the end of the toolbar.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>o_hiToolbarItem</i>	Handle to the toolbar item returned by <code>hiCreateAction</code> , <code>hiCreateToobarComboBox</code> , <code>hiCreateToolbarTypein</code> , or <code>hiCreateToolbarSeparator</code> .

Values Returned

<i>t</i>	The toolbar item was added to the toolbar.
<i>nil</i>	There was an error.

Examples

```
hiAddToolbarItem(toolbar typein)
```

Related Topics

[Toolbar Functions](#)

[hiAddToolbarItems](#)

[hiInsertToolbarItems](#)

[hiDeleteToolbarItem](#)

[hiDeleteToolbarItems](#)

[hiCreateToolbarTypein](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiCreateToolbarSeparator

hiAddToolbarItems

```
hiAddToolbarItems(  
    o_hiToolbar  
    l_toolbarItems  
)  
=> t / nil
```

Description

Adds a list of toolbar items to a toolbar. The items are added at the end of the toolbar.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>l_toolbarItems</i>	List of toolbar items, each of which is returned by <code>hiCreateAction</code> , <code>hiCreateToobarComboBox</code> , <code>hiCreateToolbarTypein</code> , or <code>hiCreateToolbarSeparator</code> .

Values Returned

<i>t</i>	The toolbar items were added to the toolbar.
<i>nil</i>	There was an error.

Examples

```
hiAddToolbarItems( toolbar list ( ComboItem typein ) )
```

Related Topics

[Toolbar Functions](#)

[hiInsertToolbarItems](#)

[hiDeleteToolbarItem](#)

[hiDeleteToolbarItems](#)

[hiCreateToolbarTypein](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiCreateToolbarSeparator

hiCreateAction

```
hiCreateAction(  
    ?name s_name  
    [ ?callback t_callback ]  
    [ ?icon g_icon ]  
    [ ?iconText t_iconText ]  
    [ ?toolTip t_tooltip ]  
    [ ?enabled g_enabled ]  
    [ ?checkable g_checkable ]  
    [ ?checked g_checked ]  
    [ ?defAction g_defaultAction ]  
    [ ?actionList l_actionList ]  
    [ ?statusTip t_statusTip ]  
    [ ?useNameForIconText g_useNameForIconText ]  
)  
=> o_action
```

Description

Creates an action for a toolbar or menu. An action is a toolbar or menu item that, when clicked, triggers a callback. Actions are typically used for common commands that are added to multiple places in the user interface, such as toolbars and menus. An action present in both a toolbar and a menu remains synchronized. For example, if it is selected from the toolbar, the corresponding menu item is also checked.

Actions can have sub-actions that are displayed in a drop-down list (in a toolbar) or a submenu (in a menu). Use the `?actionList` argument to create sub-actions.

Actions can be checkable. Checkable actions are toggle items; they can remain pressed (on a toolbar) or display a check box (on a menu) to visually indicate that they are currently selected. Use the `?checkable` argument to create these toggle actions.

For callbacks to work correctly, you must specify either the `?callback` or `?actionList` argument. If a callback is specified, it is triggered when the action is clicked. If a callback is not specified, then the `?defaultAction` specified is used if it is on the action list. If the default action is not on the action list, then the first item in the action list is used.

Arguments

<code>?name <i>s_name</i></code>	Symbol specifying a name for the action. After action for the toolbar item is created, you can change its visibility by setting the <code>invisible</code> property.

`?callback t_callback`

Cadence User Interface SKILL Reference

Toolbar Functions

The callback function is triggered when the action is clicked.

?icon *g_icon*

The icon is displayed for the action.

After the toolbar is created, you can change the icon with the `hiIcon` property. For example:

```
windowID->hiToolbars->toolbarName->actionName->hiIcon = "EditIcon.gif"
```

?iconText *t_iconText*

The text that is displayed for the action. *t_iconText* is displayed if *g_icon* is not specified or is `nil`.

After the toolbar is created, you can change the text with the `hiIconText` property. For example:

```
windowID->hiToolbars->toolbarName->actionName->hiIconText = "Edit Mode"
```

?toolTip *t_tooltip*

The tooltip for the action that is displayed when the pointer hovers over it.

After the toolbar is created, you can change the tooltip with the `hiToolTip` property. For example:

```
windowID->hiToolbars->toolbarName->actionName->hiToolTip = "Sets mode"
```

?enabled *g_enabled*

Enables the specified action. The default value is `t`.

After the toolbar is created, you can use the `enabled` property to enable or disable it. For example:

```
windowID->hiToolbars->toolbarName->actionName->enabled = t
```

?checkable *g_checkable*

Enables the action to be checkable, that is, if you want it to be a toggle item. A checkable item remains pressed (on a toolbar) or has a check-mark next to it (on a menu) when it is selected.

?checked *g_checked*

Checks the action initially.

?defAction *g_defaultAction*

Cadence User Interface SKILL Reference

Toolbar Functions

The default action for the action list. This argument is used if `?callback` is not specified.

After the toolbar is created, you can use the `defaultAction` property to set the default action. For example:

```
windowID->hiToolbars->toolbarName->actionName->defaultAction = t
```

`?actionList` *l_actionList*

List containing the sub-actions for an action. Each element in the list can be a non-checkable action created with `hiCreateAction` or a separator created with `hiCreateToolbarSeparator`. There is no restriction on uncheckable sub-actions.

This list of choices appears as a drop-down list on a toolbar or as a submenu of the action on a menu.

After the toolbar is created, you can use the `actionList` property to set the action list. For example:

```
windowID->hiToolbars->toolbarName->actionName->actionList = newActionlist
```

A sub-action is a property of its parent action. For example, to access a sub-action from a pull-down menu, you can have:

```
mypulldown->toplevel_action->sub_action->checked  
mypulldown->toplevel_action->sub_action->enabled
```

`?statusTip` *t_statusTip*

Displays the status text in the status bar at the bottom of the window when the mouse hovers over the action. If you do not specify this argument, the action's text (`t_iconText`) is displayed.

`?useNameForIconText` *g_useNameForIconText*

When an action is instantiated, the `?iconText` argument, if unspecified or is an empty string, is set to the print name of `?name` if and only if `g_useNameForIconText` is not nil. If this argument is nil, and the `?iconText` argument is not specified or is an empty string, the tool button shows an icon without text. The default value is `t`.

Cadence User Interface SKILL Reference

Toolbar Functions

Value Returned

o_action The handle to the action.

Examples

```
new=hiCreateAction( ?name 'new ?iconText "New" ?callback "newCBFunction" )
hiAddToolbarItem( toolbar new )
OR
hiAddMenuItem( menu new )
OR
hiCreatePulldownMenu( 'file "File" list ( new ) )
```

Related Topics

[Toolbar Functions](#)

[hiSetActionChecked](#)

[hiSetActionChecked](#)

[hiCreateToolbarTypein](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiCreateToolbar

```
hiCreateToolbar(  
    ?name s_name  
    [ ?title t_title ]  
    [ ?toolButtonStyle s_buttonStyle ]  
    [ ?toolTip t_toolTip ]  
    [ ?items l_toolbarItems ]  
    [ ?invisible g_invisible ]  
    [ ?contextMenuItems l_contextMenuItems ]  
)  
=> o_hiToolbar
```

Description

Creates a toolbar that you can add to windows and session windows. A toolbar is a container object that can contain combo boxes, type-in items, actions, and separators.

Arguments

<code>?name <i>s_name</i></code>	Symbol specifying a name for the toolbar. You can reference the toolbar with this symbol in the following way: <i>windowID->hiToolbars->name</i>
<code>?title <i>t_title</i></code>	Title of the toolbar that is displayed in the session window's popup menu. If you do not specify a title, <i>s_name</i> is used.
<code>?toolButtonStyle <i>s_buttonStyle</i></code>	One of the following symbols specifying the style of the toolbar buttons: 'textOnly, 'iconOnly, 'textBesideIcon, or 'textUnderIcon. Default value: 'iconOnly.
<code>?toolTip <i>t_toolTip</i></code>	Specifies the tooltip, which is displayed whenever the cursor hovers over the toolbar.
<code>?items <i>l_toolbarItems</i></code>	List of items to be displayed in the toolbar; return value of <code>hiCreateAction</code> , <code>hiCreateToolbarComboBox</code> , <code>hiCreateToolbarTypein</code> , or <code>hiCreateToolbarSeparator</code> . The items appear in the order in which they are listed.

Cadence User Interface SKILL Reference

Toolbar Functions

`?invisible` *g_invisible*

t or *nil*. Specify *t* if you want the toolbar to be invisible initially; specify *nil* if you want it to be displayed. The default value is *nil*.

You can modify the value of this argument after the toolbar is created with the *invisible* property. For example:

```
windowId->hiToolbars->toolbarName->invisible  
= nil
```

`?contextMenuItems` *l_contextMenuItems*

List of menu items to populate the context menu.

Values Returned

o_hiToolbar Handle to the toolbar.

Example

```
ComboItem=hiCreateToolbarComboBox(  
    ?name 'comboItem  
    ?value "DesignA"  
    ?items list ( "DesignA" "DesignB" "DesignC" )  
    ?editable t )  
  
toolbar=hiCreateToolbar( ?name 'toolbar ?title "My Designs" ?items list (ComboItem  
    ) )  
  
w=hiCreateWindow( 'default "graphics" "Test Window" )  
hiPlaceToolbar( w toolbar 'top )  
hiDisplayWindow(w)
```

Related Topics

[Toolbar Functions](#)

[hiPlaceToolbar](#)

[hiInsertToolbar](#)

[hiShowToolbar](#)

[hiSetToolbarObjName](#)

hiCreateToolbarComboBox

```
hiCreateToolbarComboBox(
    ?name s_name
    ?items l_items
    [ ?prompt t_prompt ]
    [ ?value t_value ]
    [ ?toolTip t_tooltip ]
    [ ?callback t_callback ]
    [ ?editable g_editable ]
    [ ?iconFile t_iconFile ]
    [ ?enabled g_enabled ]
    [ ?width x_width ]
    [ ?statusTip t_statusTip ]
    [ ?CBOnEnter g_cbOnEnter ]
    [ ?focusInCallback g_focusInCallback ]
)
=> o_comboBox
```

Description

Creates a drop-down list box item (popularly called a combo box) to be added to the toolbar created using `hiCreateToolbar`. The drop-down list box has a type-in field as well as a drop-down list. You can either type in a value or select an item from the list.

Arguments

<code>?name s_name</code>	Symbol specifying the name of the toolbar drop-down list box. After drop-down list box item is created, you can change its visibility by setting the <code>invisible</code> property. For example, to hide workspace drop-down list box in the workspaces toolbar: <pre>(deFindToolbar "Workspaces" swindow(1)) -> deWorkspacesCombo->invisible=t"</pre>
<code>?items l_items</code>	List of strings that represent the choices in the drop-down list.
<code>?prompt t_prompt</code>	String specifying the prompt (label) for the combination box.
<code>?value t_value</code>	The initial value of the combination box.
<code>?toolTip t_tooltip</code>	The tooltip for the combo box that appears when the pointer hovers over it.
<code>?callback t_callback</code>	

Cadence User Interface SKILL Reference

Toolbar Functions

Specifies the callback function, which gets triggered when the combination box value changes.

`?editable g_editable`

Edits the combination box. Specify `nil` if you want it to be read-only. The default value is `t`.

`?iconFile t_iconFile`

Specifies the name of the icon file.

`?enabled g_enabled`

Enables the combination box. Specify `nil` if you want it disabled, that is, grayed-out. The default value is `t`.

`?width x_width` Width of the combination box.

`?statusTip t_statusTip`

The status text that is displayed in the status bar at the bottom of the window when the mouse hovers over the combination box.

`?CBOnEnter g_CBOnEnter`

If specified as a non-`nil` value, the callback function is triggered when `Enter` or `Return` key is pressed in the drop-down list box whether the value entered changes or not. If specified as `nil`, the callback function is triggered when the value in the drop-down list box changes.

`?focusInCallback g_focusInCallback`

SKILL function to be executed whenever the field comes into focus, specifically when the type-in part of the field comes into focus. This happens either when you click in the type-in part of the field or when you dismiss the drop-down list, which shifts the focus to the type-in part of the field. The callback can be of string, symbol, or function object type.

Cadence User Interface SKILL Reference

Toolbar Functions

Values Returned

o_comboBox Handle to the toolbar combination box.

Examples

```
ComboItem = hiCreateToolbarComboBox(  
    ?name 'comboItem  
    ?value "DesignA"  
    ?items list ( "DesignA" "DesignB" "DesignC" )  
    ?editable t )  
  
toolbar = hiCreateToolbar( ?name 'toolbar ?title "My Designs" ?items list (  
    ComboItem ) )
```

Related Topics

[Toolbar Functions](#)

[hiCreateToolbarTypein](#)

[hiCreateToolbarSeparator](#)

[hiAddToolbarItems](#)

[hiInsertToolbarItems](#)

hiCreateToolbarTypein

```
hiCreateToolbarTypein(  
    ?name s_name  
    [ ?prompt t_prompt ]  
    [ ?value t_value ]  
    [ ?tooltip t_tooltip ]  
    [ ?callback t_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?width x_width ]  
    [ ?statusTip t_statusTip ]  
    [ ?CBOnEnter g_cbOnEnter ]  
)  
=> o_typein
```

Description

Creates a type-in field to be added to the toolbar created using `hiCreateToolbar`.

Arguments

<code>?name <i>s_name</i></code>	Symbol specifying the name of the type-in toolbar item. After type-in toolbar item is created, you can change its visibility by setting the <code>invisible</code> property.
<code>?prompt <i>t_prompt</i></code>	String specifying the prompt (label) for the type-in toolbar item.
<code>?value <i>t_value</i></code>	The initial value of the type-in toolbar item.
<code>?tooltip <i>t_tooltip</i></code>	The tooltip for the type-in item that appears when the pointer hovers over it.
<code>?callback <i>t_callback</i></code>	Specifies the callback function, which gets triggered when the type-in item value changes.
<code>?editable <i>g_editable</i></code>	<code>t</code> or <code>nil</code> . Specify <code>t</code> if you want the toolbar item to be editable; specify <code>nil</code> if you want it to be read-only. The default value is <code>t</code> .
<code>?enabled <i>g_enabled</i></code>	

Cadence User Interface SKILL Reference

Toolbar Functions

`t` or `nil`. Specify `t` if you want the toolbar item to be enabled; specify `nil` if you want it disabled, that is, grayed-out. The default value is `t`.

`?width` *x_width* Width of the toolbar item.

`?statusTip` *t_statusTip*

The status text that is displayed in the status bar at the bottom of the window when the mouse hovers over the toolbar item.

`?CBOnEnter` *g_CBOnEnter*

If specified as a non-`nil` value, the callback function is triggered when `Enter` or `Return` key is pressed in the drop-down list box whether the value entered changes or not. If specified as `nil`, the callback function is triggered when the value in the drop-down list box changes.

Values Returned

o_typein Handle to the type-in toolbar item.

Examples

```
hiCreateToolbarTypein(?name 'typein ?prompt "Account Number" ?value "123" )
```

Related Topics

[Toolbar Functions](#)

[hiCreateToolbarComboBox](#)

[hiCreateToolbarSeparator](#)

[hiInsertToolbarItems](#)

hiCreateToolbarSeparator

```
hiCreateToolbarSeparator(  
    ?name s_name  
)  
=> o_separator
```

Description

Creates a separator that is used to separate the items in a toolbar created using `hiCreateToolbar`.

Argument

<i>s_name</i>	Symbol specifying a name for the toolbar separator. After toolbar separator item is created, you can change its visibility by setting the <code>invisible</code> property.

Values Returned

<i>o_separator</i>	The handle to the toolbar separator.

Examples

```
hiCreateToolbarSeparator( ?name 'separator )
```

Related Topics

[Toolbar Functions](#)

[hiCreateToolbarComboBox](#)

[hiCreateToolbarTypein](#)

[hiInsertToolbarItems](#)

[hiAddToolbarItems](#)

hiDeleteToolbar

```
hiDeleteToolbar(  
    w_windowID  
    s_toolbar  
)  
=> t / nil
```

Description

Deletes a toolbar from a window.

Arguments

<i>w_windowID</i>	The ID of the window that contains the toolbar.
<i>s_toolbar</i>	The name of the toolbar, which was set with the ?name argument when the toolbar was created.

Values Returned

t	Deleted the toolbar.
nil	There was an error.

Example

Deletes 'Custom toolbar from window (1).

```
hiDeleteToolbar(window(1) 'Custom)
```

Related Topics

[Toolbar Functions](#)

[hiCreateToolbar](#)

[hiPlaceToolbar](#)

[hiInsertToolbar](#)

[hiShowToolbar](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiSetToolbarObjName

hiDeleteToolbarBreak

```
hiDeleteToolbarBreak(  
    w_windowID  
    s_toolbar  
)  
=> t / nil
```

Description

Deletes the toolbar break before the specified toolbar from the toolbar area of a window.

Arguments

<i>w_windowID</i>	The ID of the window that contains the toolbar.
<i>s_toolbar</i>	The name of the toolbar before which you want to delete the toolbar break

Values Returned

<i>t</i>	The toolbar break before the specified toolbar was deleted.
<i>nil</i>	The operation fails in case of an error.

Example

```
hiDeleteToolbarBreak( window(1) toolbar )
```

Related Topics

[Toolbar Functions](#)

hiDeleteToolbarItem

```
hiDeleteToolbarItem(  
    o_hiToolbar  
    s_toolbarItem  
)  
=> t / nil
```

Description

Deletes a toolbar item from a toolbar.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>s_toolbarItem</i>	The name of the toolbar item to be deleted from the toolbar.

Value Returned

<i>t</i>	The toolbar item was removed from the toolbar.
<i>nil</i>	The toolbar item could not be removed from the toolbar.

Example

```
hiDeleteToolbarItem( toolbar 'typein )
```

Related Topics

[Toolbar Functions](#)

[hiDeleteToolbarItems](#)

[hiAddToolbarItems](#)

[hiInsertToolbarItems](#)

[hiCreateToolbarComboBox](#)

[hiCreateToolbarTypein](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiDeleteToolbarItems

```
hiDeleteToolbarItems(  
    o_hiToolbar  
    l_toolbarItems  
)  
=> t / nil
```

Description

Removes the specified toolbar items from a toolbar.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>l_toolbarItems</i>	List of toolbar names specifying the toolbar items to be deleted from the toolbar.

Values Returned

<i>t</i>	The toolbar item was removed from the toolbar.
<i>nil</i>	The toolbar item could not be removed from the toolbar.

Examples

```
hiDeleteToolbarItems( toolbar list ( 'comboItem 'typein )
```

Related Topics

[Toolbar Functions](#)

[hiDeleteToolbarItem](#)

[hiAddToolbarItems](#)

[hiInsertToolbarItems](#)

[hiCreateToolbarComboBox](#)

[hiCreateToolbarTypein](#)

hiFocusToToolbarItem

```
hiFocusToToolbarItem(  
    ?toolbar S_toolbar  
    ?item S_toolbarItem  
    [?warpPointer g_warp]  
    [?window w_window]  
)  
=> t / nil
```

Description

Sets the keyboard focus on a visible, enabled, and editable toolbar type-in or combo item.

Arguments

?toolbar S_toolbar

Specifies the name of a toolbar symbol, a title string, or a tooltip string.

?item S_toolbarItem

Specifies the name symbol of a toolbar item or a tooltip string.

?warpPointer g_warp

When set to `t`, the mouse pointer warps to the field when the focus is set.

?window w_window

Specifies the window containing the toolbar. If not specified, the default is the current window.

The toolbar and item symbols and string for type-in and combo toolbar items can be listed with the SKILL function `hiListFocusableToolbarItems`.

Cadence User Interface SKILL Reference

Toolbar Functions

Value Returned

<code>t</code>	The operation is successful.
<code>nil</code>	The operation fails and a warning is issued when one of the following is true: ■ An invalid window argument is specified. ■ The specified toolbar or item cannot be found in the window. ■ The item specified is not a type-in or combo item. No warning is issued when the type-in or combo item is found, but is not enabled, editable, or visible.

Example

This function can be used to set a bindkey to quickly modify a toolbar value. The following command sets bindkey `Ctrl + S` to focus on the *Display Stop Level* toolbar item in the layout window:

```
> hiSetBindKey("Layout" "Ctrl<Key>S" "hiFocusToToolbarItem(?toolbar \"Options\"  
?item \"Display Stop Level\")")
```

Related Topics

[Toolbar Functions](#)

[hiListFocusableToolbarItems](#)

hiGetSharedIcon

```
hiGetSharedIcon(  
    t_callback  
)  
=> g_icon / nil
```

Description

Gets the icon associated with an action. The icon is in the format in which it was originally specified: an image file name or an hilcon list.

Action icons are maintained in a table and this function enables you to look them up by the callback name of the action.

When you create a menu item with a callback of the same name as an action callback, the icon associated with the action will be used for the menu item, regardless of whether you specify an icon for the menu item. However, instead of relying on that to share icons between a menu item created with `hiCreateMenuItem` and an action, you should use actions directly in menus.

Argument

<i>t_callback</i>	The name of the callback function that was specified for the action when the action was created with <code>hiCreateAction</code> .

Value Returned

<i>g_icon</i>	The action's icon, in the format in which it was specified when the action was created.
nil	An icon does not exist for the action or <i>t_callback</i> does not exist.

Related Topics

[Toolbar Functions](#)

[hiSetActionChecked](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiGetToolbarObjName

```
hiGetToolbarObjName (
    o_hiToolbar
)
=> t
```

Description

Gets the name that is used to store the toolbar window configuration information. This name is set with the `hiSetToolbarObjName` function.

Argument

<code>o_hiToolbar</code>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .

Value Returned

<code>t_name</code>	The toolbar name.

Example

```
hiGetToolbarObjName ( toolbar )
```

Related Topics

[Toolbar Functions](#)

[hiSetToolbarObjName](#)

[hiCreateToolbar](#)

[hiPlaceToolbar](#)

[hiInsertToolbar](#)

hiGetWindowToolbars

```
hiGetWindowToolbars(  
    w_windowId  
)  
=> l_toolbars
```

Description

Gets the list of toolbars that a window contains. The toolbars are listed in the order in which they were placed in the window with `hiPlaceToolbar`.

Argument

<i>w_windowId</i>	The ID of a window or session window.

Value Returned

<i>l_toolbars</i>	List of toolbar handles, in the order in which they were placed in the window with <code>hiPlaceToolbar</code> .

Example

```
hiGetWindowToolbars( window ( 1 ) )
```

Related Topics

[Toolbar Functions](#)

[hiSetToolbarObjName](#)

[hiCreateToolbar](#)

[hiPlaceToolbar](#)

[hiInsertToolbar](#)

hiHasToolbarBreak

```
hiHasToolbarBreak(  
    w_windowID  
    s_toolbar  
)  
=> t / nil
```

Description

Returns a Boolean value indicating whether there was a toolbar break before the specified toolbar.

Arguments

<i>w_windowID</i>	The ID of the window that contains the toolbar.
<i>s_toolbar</i>	The name of the toolbar you want to check.

Values Returned

<i>t</i>	There was a toolbar break before the specified toolbar.
<i>nil</i>	There was no toolbar break before the specified toolbar or there was an error.

Examples

```
hiHasToolbarBreak( window(1) toolbar )
```

Related Topics

[Toolbar Functions](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiHideToolbar

```
hiHideToolbar(  
    o_hiToolbar  
    [ g_detach ]  
)  
=> t / nil
```

Description

Hides a toolbar. If *g_detach* is set to *t*, this function removes the toolbar from the window.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>g_detach</i>	Removes toolbar from the window. If set to <code>nil</code> or not specified, the toolbar is made invisible but is not removed from the window.

Values Returned

<i>t</i>	The toolbar is hidden or removed.
<i>nil</i>	There was an error.

Examples

Hides the toolbar.

```
hiHideToolbar( toolbar nil )  
=>t
```

Related Topics

[Toolbar Functions](#)

[hiShowToolbar](#)

[hiCreateToolbar](#)

[hiPlaceToolbar](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiInsertToolbar

hiSetToolbarObjName

hiSetActionChecked

hiInsertToolbar

```
hiInsertToolbar(  
    w_windowID  
    o_hiToolbar  
    s_toolbar  
)  
=> t / nil
```

Description

Inserts a toolbar into the toolbar area of a window before *s_toolbar*.

Arguments

<i>w_windowID</i>	The window ID.
<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>s_toolbar</i>	The name of the toolbar before which you want to insert <i>o_hiToolbar</i> .

Values Returned

<i>t</i>	Inserted toolbar in the window.
<i>nil</i>	There was an error.

Example

```
hiInsertToolbar( window ( 1 ) toolbar 'Custom )
```

Related Topics

[Toolbar Functions](#)

[hiPlaceToolbar](#)

[hiCreateToolbar](#)

[hiShowToolbar](#)

[hiSetToolbarObjName](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiSetActionChecked

hiInsertToolbarBreak

```
hiInsertToolbarBreak(  
    w_windowID  
    s_toolbar  
)  
=> t / nil
```

Description

Inserts a toolbar break into the toolbar area of a window before the specified toolbar.

Arguments

<i>w_windowID</i>	The ID of the window that contains the toolbar.
<i>s_toolbar</i>	The name of the toolbar before which you want to insert the toolbar break.

Values Returned

<i>t</i>	The toolbar break was inserted in the window.
<i>nil</i>	The operation fails in case of an error.

Examples

Inserts a toolbar break before `toolbar`.

```
hiInsertToolbarBreak( window(1) toolbar )  
=>t
```

Related Topics

[Toolbar Functions](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiInsertToolbarItem

```
hiInsertToolbarItem(  
    o_hiToolbar  
    o_hiToolbarItem  
    s_toolbarItemBefore  
)  
=> t / nil
```

Description

Inserts a toolbar item before a specific item in a toolbar.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>o_hiToolbarItem</i>	Handle to the toolbar item returned by <code>hiCreateAction</code> , <code>hiCreateToobarComboBox</code> , <code>hiCreateToolbarTypein</code> , or <code>hiCreateToolbarSeparator</code> .
<i>s_toolbarItemBefore</i>	The name of the toolbar item before which to add the new toolbar item.

Values Returned

<i>t</i>	The toolbar item was added.
<i>nil</i>	There was an error.

Examples

Inserts `typein` before `comboItem`.

```
hiInsertToolbarItem(toolbar typein 'comboItem )  
=>t
```

Related Topics

[Toolbar Functions](#)

[hiInsertToolbarItems](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiAddToolBarItems

hiDeleteToolBarItem

hiDeleteToolBarItems

hiCreateToolBarTypein

hiCreateToolBarSeparator

Cadence User Interface SKILL Reference

Toolbar Functions

hiInsertToolbarItems

```
hiInsertToolbarItems(  
    o_hiToolbar  
    l_hiToolbarItems  
    s_toolbarItemBefore  
)  
=> t / nil
```

Description

Inserts toolbar items before the specified item in a toolbar.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>l_hiToolbarItems</i>	List of toolbar items returned by <code>hiCreateAction</code> , <code>hiCreateToolbarComboBox</code> , <code>hiCreateToolbarTypein</code> , or <code>hiCreateToolbarSeparator</code> .
<i>s_toolbarItemBefore</i>	The name of the toolbar item before which to add the new toolbar items.

Values Returned

<i>t</i>	The toolbar items were added to the toolbar.
<i>nil</i>	There was an error.

Examples

Inserts `typein` and `anotherTypein` before `comboItem`.

```
hiInsertToolbarItem(toolbar list ( typein anotherTypein ) 'comboItem )  
=>t
```

Related Topics

[Toolbar Functions](#)

[hiAddToolbarItems](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiDeleteToolBarItem

hiDeleteToolBarItems

hiCreateToolBarComboBox

hiCreateToolBarTypein

Cadence User Interface SKILL Reference

Toolbar Functions

hiListFocusableToolbarItems

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

Description

Lists all type-in and combo toolbar items that are currently defined in the specified window and can be used as arguments for the function `hiFocusToToolbarItem`.

Arguments

<code>w_window</code>	Specifies the window containing the type-in and combo toolbar items. If not specified, the default is the current window.

Value Returned

<code>t</code>	The operation is successful.
<code>nil</code>	The operation fails and a warning is issued if the specified window is invalid.

Examples

The following example shows partial listing from a layout window in which the *Display Stop Level* toolbar item is not visible due to overlapping of toolbars:

```
> hiListFocusableToolbarItems ()
Toolbar Name/Symbol      Item Name/Symbol      Type   Editable  Enabled  Visibility
-----
"Workspaces"             "Workspace Configuration"  combo    No        Yes      Visible
'deWorkspacesToolbar     'deWorkspacesCombo
-----
"Align"                  "Align Spacing Value"      typein   No        No       Toolbar not visible
'leAlignToolbar          'leAlignToolbarAlignSpacingValue
-----
"Orthogonal Spacing Value" typein   No        No       Toolbar not visible
'leAlignToolbarAlignOrthogonalSpacingValue
-----
"Options"                "Display Stop Level"      typein   Yes       Yes      Not visible
'lebOptionsToolbar       'lebDisp
=> t
```

Cadence User Interface SKILL Reference

Toolbar Functions

For `hiFocusToToolbarItem` to work with a Display Stop Level item, the toolbars need to be arranged in such a way that the item is visible. Here, the other three items are either not editable or not enabled. The function does not work with them even if they are made visible.

Related Topics

[Toolbar Functions](#)

[hiFocusToToolbarItem](#)

hiPlaceToolbar

```
hiPlaceToolbar(  
    w_windowID  
    o_hiToolbar  
    [ s_toolbarArea ]  
)  
=> t / nil
```

Description

Adds a toolbar to the specified toolbar area of a window. If the toolbar already exists on the window, this function moves it to the specified area. The toolbar is placed at the end of the current toolbars in that area.

Arguments

<i>w_windowID</i>	The window ID.
<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>s_toolbarArea</i>	The toolbar area of the window; one of: 'left, 'right, 'top, and 'bottom. Default value: 'top

Values Returned

<i>t</i>	The toolbar was added to the window.
<i>nil</i>	There was an error.

Examples

Adds toolbar to the top of the window.

```
hiPlaceToolbar( window ( 1 ) toolbar 'top )  
=>t
```

Related Topics

[Toolbar Functions](#)

[hiInsertToolbar](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiCreateToolbar

hiShowToolbar

hiSetToolbarObjName

hiPlaceToolbarBreak

```
hiPlaceToolbarBreak  
    w_windowID  
    s_toolbarArea  
)  
=> t / nil
```

Description

Adds a toolbar break to the specified toolbar area of a window.

Arguments

<i>w_windowID</i>	The ID of the window that contains the toolbar.
<i>s_toolbarArea</i>	The toolbar area of the window; one of: 'left', 'right', 'top', and 'bottom'. Default value: 'top'

Values Returned

t	The toolbar break was added to the window
nil	The operation fails in case of an error.

Examples

Adds a toolbar break to the `top` toolbar.

```
hiPlaceToolbarBreak( window(1) 'top)  
=>t
```

Related Topics

[Toolbar Functions](#)

hiReattachToolbar

```
hiReattachToolbar(  
    o_toolbar  
)  
=> t / nil
```

Description

Reattaches a given toolbar to its parent main window without showing it.

Intended for use with those toolbars that have been detached from a window using `hiHideToolbar`.

Argument

<code>o_toolbar</code>	The toolbar to be reattached.

Values Returned

<code>t</code>	The toolbar has been successfully reattached.
<code>nil</code>	The command failed.

Examples

Reattaches `mytoolbar` to its parent window.

```
( hiHideToolbar mytoolbar t/*detach*/ )  
( hiReattachToolbar mytoolbar ) ;; reattached but not shown
```

Related Topics

[Toolbar Functions](#)

[hiDeleteToolbarItem](#)

[hiHideToolbar](#)

[hiInsertToolbarItems](#)

[hiCreateToolbarComboBox](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiCreateToolbarTypein

Cadence User Interface SKILL Reference

Toolbar Functions

hiShowToolbar

```
hiShowToolbar(  
    o_hiToolbar  
)  
=> t / nil
```

Description

Displays a hidden toolbar.

If the toolbar was hidden and removed from the window by calling `hiHideToolbar ( toolbar t )`, it is added back to the window and then displayed.

Argument

<code>o_hiToolbar</code>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .

Values Returned

<code>t</code>	The toolbar is displayed.
<code>nil</code>	There was an error.

Examples

Displays the `toolbar`.

```
hiShowToolbar( toolbar )  
=>t
```

Related Topics

[Toolbar Functions](#)

[hiCreateToolbar](#)

[hiPlaceToolbar](#)

[hiInsertToolbar](#)

[hiSetToolbarObjName](#)

Cadence User Interface SKILL Reference

Toolbar Functions

hiSetActionChecked

hiSetToolbarObjName

```
hiSetToolbarObjName(  
    o_hiToolbar  
    t_name  
)  
=> t / nil
```

Description

Sets a name for the toolbar that is used for storing window configuration information about the toolbar. Every toolbar must have a unique name that does not change. Do not use this function with toolbars provided by Cadence.

The object name is initially set to the ?title specified when the toolbar was created, until you set a new object name with this function.

Arguments

<i>o_hiToolbar</i>	Handle to the toolbar returned by <code>hiCreateToolbar</code> .
<i>t_name</i>	The name you want to set for the toolbar.

Values Returned

<i>t</i>	The toolbar name was set.
<i>nil</i>	There was an error.

Examples

Sets the name "FileWin" for toolbar.

```
hiSetToolbarObjName( toolbar "FileWin" )  
=>t
```

Related Topics

Related Topics

Toolbar Functions

Cadence User Interface SKILL Reference

Toolbar Functions

hiSetActionChecked

hiCreateToolbar

hiPlaceToolbar

hiInsertToolbar

hiShowToolbar

hiSetActionChecked

```
hiSetActionChecked(  
    o_action  
    g_checkState  
    [ g_triggerCB ]  
)  
=> t / nil
```

Description

Sets checked state for an action and triggers the action callback, unless you specify otherwise by setting *g_triggerCB* to *nil*.

Arguments

<i>o_action</i>	Handle to the action returned by <i>hiCreateAction</i> .
<i>g_checkState</i>	Sets the state as checked.
<i>g_triggerCB</i>	Triggers the callback of an action when the state is changed with this function. The default value of this argument is <i>t</i> .

Values Returned

<i>t</i>	The state was changed.
<i>nil</i>	There was an error.

Related Topics

Toolbar Functions

hiSetActionChecked

Dialog Boxes

Dialog boxes are pop-up windows usually containing a brief message that is displayed to the user. Dialog boxes can be modal or modeless.

- Modal, or blocking, dialog boxes

A modal dialog box blocks access to the application as long as it is displayed. Users must finish using the dialog box and must close it before they can continue with the application. However, users can still execute window manager commands and use other applications.

- Modeless, or non-blocking, dialog boxes

A modeless dialog box does not block access to the application. It is often used to provide additional information or display a warning message. The dialog box remains on the screen until the user dismisses it.

Information Dialog Boxes

You can use the `hiDisplayAppDBox` or `hiDisplayUserDBox` functions to create dialog boxes that display information such as messages, warnings, and errors.

Dialog boxes created with `hiDisplayAppDBox` have a set of pre-defined buttons—a combination of *OK*, *Yes*, *No*, *Cancel*, *Close*, and *Help* buttons. These dialog boxes can be closed by clicking the *OK* or *Cancel* button or by pressing *Return* while the cursor is within the box, which is the equivalent of clicking the *OK* button.

Dialog boxes created with `hiDisplayUserDBox` (or with `hiDisplayAppDBox` with `s_buttonLayout` set to 'UserDefined') can have any buttons that you specify. You also specify what happens when the buttons are clicked.

Do not use the window manager to close dialog boxes or forms. Always close or cancel dialog boxes with the buttons in the dialog box.

File Selection Dialog Boxes

File selection dialog boxes let users select a file or a directory. They can be modal (blocking) or modeless (non-blocking).

Progress Dialog Boxes

Progress dialog boxes display the progress of a command. Progress dialog boxes are always modeless (non-blocking).

Related Topics

[General Dialog Box Functions](#)

[File Selection Dialog Box Functions](#)

[Progress Dialog Box Functions](#)

[Color Dialog Function](#)

[hiDisplayAppDBox](#)

[hiDisplayUserDBox](#)

General Dialog Box Functions

This topic provides a list of Cadence® SKILL functions associated with general dialog box.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiDisplayAppDBox](#)
- [hiDisplayUserDBox](#)
- [hiDBoxCancel](#)
- [hiDBoxOK](#)
- [hiSetDBoxDefaultLocation](#)
- [hiGetDBoxDefaultLocation](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDisplayAppDBox

```
hiDisplayAppDBox(  
    ?name s_dboxHandle  
    [ ?dboxBanner t_dboxBanner ]  
    [ ?dboxText t_dboxText ]  
    [ ?callback g_callback ]  
    [ ?dialogType x_dialogType ]  
    [ ?dialogStyle s_dialogStyle ]  
    [ ?buttonLayout s_buttonLayout ]  
    [ ?buttons g_buttons ]  
    [ ?dontPopdowns g_dontPopdowns ]  
    [ ?defaultButton x_defaultButton ]  
    [ ?location l_location ]  
    [ ?help g_help ]  
)  
=> t / nil / cancel
```

Description

Creates and displays a dialog box. You can add a pre-defined set of buttons to the dialog box or you can specify your own buttons. The dialog box is destroyed when it is removed from the screen.

Arguments

?name *s_dboxHandle*

SKILL handle to the dialog box. The *dboxHandle* symbol is set to point to the created dialog box; when the dialog box is dismissed, this symbol is reset to *nil*.

?dboxBanner *t_dboxBanner*

Text that appears in the window manager banner of the dialog box.

?dboxText *t_dboxText*

Message that is displayed in the dialog box.

?callback *g_callback*

Cadence User Interface SKILL Reference

Dialog Boxes

Text string specifying the SKILL procedure to be executed when the user clicks *OK* or *Yes* in the dialog box. This callback is called if *s_buttonLayout* is 'OKCancel, 'YesNo, or 'YesNoCancel. If *s_buttonLayout* is 'UserDefined and you use the *g_buttons* argument to define your own buttons, specify *g_callback* as a list of callbacks which corresponds to the list of buttons. Each element in the list is a string (name of the callback function) or *nil* (if you do not want to specify a callback for a button). The first callback in *g_callback* is the callback for the first button in *g_buttons*, the second callback is for the second button, and so on.

If you specified a *Help* button in *g_buttons* and if you want the standard Doc Assistant mechanism to be invoked when the user clicks the button, use the following procedure as the callback for the *Help* button:

```
procedure ( myHelp ( ) hiHelp ( 'dbox dBoxName ))
```

If *s_buttonLayout* is 'UserDefined and *s_dialogStyle* is 'modeless, you should specify a callback for each button, as the function returns *t* immediately after the dialog box is displayed (if there is no error). You can write the callback in a way that it returns the button index. For example, write the callback as:

```
procedure( myCallback ( button )  
when (button == 1 "Overwrite selected")
```

and, in the *hiDisplayAppDBox* function, pass the button index to the callback:

```
?callback ' ("myCallback(1)" "myCallback(2)"  
"myCallback(3)" )
```

```
?dialogType x_dialogType
```

Cadence User Interface SKILL Reference

Dialog Boxes

One of the following dialog box types, which determine the appearance of the dialog box:

- `hicWarningDialog` (default)
- `hicErrorDialog`
- `hicInformationDialog`
- `hicMessageDialog`
- `hicQuestionDialog`
- `hicWorkingDialog`

`?dialogStyle s_dialogStyle`

One of the following symbols, which determine the blocking behavior of the dialog box:

- `'modal`

Blocks access to the application while it is displayed. The user must dismiss the dialog box in order to continue using the application. (Access to window manager commands or other applications is not blocked.)

The `'modal` dialog box cannot be displayed before the initialization—which includes reading the `.cdsinit` file or other SKILL files—is complete. To delay displaying of the `'modal` dialog box until initialization is complete, use [`hiEnqueueCmd`](#).

- `'modeless`

Does not block access to the application. The user can continue using the application without having to close the dialog box.

The default value of this argument is `'modal`.

`?buttonLayout s_buttonLayout`

Cadence User Interface SKILL Reference

Dialog Boxes

Symbol specifying the buttons that appear in this dialog box.
Valid values are:

- 'OKCancel (default)
- 'YesNo
- 'YesNoCancel
- 'CloseHelp
- 'Close
- 'UserDefined

If you want to customize the buttons, specify 'UserDefined as the value of this argument and specify your buttons with the *g_buttons* argument.

A *Help* button appears with all choices except 'Close and 'UserDefined. (See the description of *g_buttons* if you want a *Help* button with 'UserDefined.)

The dialog box callback, if any, is executed only if *s_buttonLayout* is not of type 'Close or 'CloseHelp, which simply dismiss the dialog box with no action taken.

?buttons
g_buttons

List of buttons that you want to display in the dialog box. Specify this argument as a list of strings. You can specify any string for a button, for example, "Overwrite", "Update", or "Cancel".

A *Help* button is not added by default to your list of buttons. If you want a *Help* button, specify it in the *g_buttons* list and specify a callback for it in the *g_callbacks* argument. You can also invoke the standard Doc Assistant mechanism when the user clicks *Help*; see the description of *g_callbacks* for more information.

?dontPopdowns *g_dontPopdowns*

Cadence User Interface SKILL Reference

Dialog Boxes

A list, each element of which is `t` or `nil`, that determines whether the dialog box is closed when a button is clicked. Specify `t` if you want to continue to display the dialog box when the button is clicked; specify `nil` if you want the dialog box to be closed when the button is clicked.

The items in the list correspond to the buttons in `g_buttons`. The first item in `g_dontPopdowns` determines the behavior of the first button in `g_buttons`, the second item determines the behavior of the second button, and so on.

The default value of this argument is `nil` for all buttons.

`?defaultButton` *x_defaultButton*

Specifies the default button to be used when the *Return* key is pressed. The default button should be specified by its position. Possible choices can range from 1 through 4 depending on `s_buttonLayout`. For example, if the `s_buttonLayout` argument is set to 'YesNo, and the `x_defaultButton` is set to 2, the *No* button is set as the default button. If no value or an invalid value (such as "3" when there are only two buttons) is given for this argument, the first button is set as the default.

`?location` *l_location*

The location of the dialog box on the screen, specified as the x and y coordinates of the upper left corner of the box:

`list(x-coordinate y-coordinate)`

If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

Cadence User Interface SKILL Reference

Dialog Boxes

?help *g_help*

The value of this argument is used if *s_buttonLayout* is 'OKCancel, 'YesNo, 'YesNoCancel, or 'CloseHelp. This argument is ignored if *s_buttonLayout* is 'UserDefined, unless you have created a *Help* button that calls `hiHelp('dBox dBoxName)`.

If you specify a single string or symbol, it is used to set the `hiHelpAppName` property on the symbol that represents the dialog box name. If you specify a list of strings or a list of symbols, the first element in the list is used to set the `hiHelpAppName` property and the second element is used to set the `hiHelpSymName` property.

(If you want to specify the help symbol only, specify a list of two strings or symbols, with `nil` as the first element of the list and the help symbol as the second element. This sets the `hiHelpSymName` property without setting the `hiHelpAppName` property.)

When a user clicks *Help* on the dialog box, the `hiHelp` function is called with the following arguments:

appId helpTag

If the `hiHelpAppName` property has been set, it is used as the *appId*, otherwise the application ID of the current window is used as the *appId*. If the `hiHelpSymName` property has been set, it is used as the *helpTag*, otherwise the dialog box name is used as the *helpTag*.

The *appId* and *helpTag* determine the name of the tag file that is called (*appId.tgf*) and the help tag in the file that is accessed.

You can also use the `hiHelpAppName` and `hiHelpSymName` properties directly to specify the help file and help tag that are used to reference the help.

If you do not want to use the default `.tgf` help file mechanism, you can specify your own callback for the *Help* button.

Cadence User Interface SKILL Reference

Dialog Boxes

Values Returned

If *s_buttonLayout* is not 'UserDefined, this function returns one of the following values:

<i>t</i>	The <i>OK</i> , <i>Yes</i> , or <i>Close</i> button was selected. If <i>s_dialogStyle</i> is 'modeless, the function returns <i>t</i> immediately after the dialog box is displayed, if there is no error; it does not wait for a button to be selected.
<i>nil</i>	The <i>Cancel</i> or <i>No</i> button was selected, an error occurred, or one of the arguments is invalid.
'cancel	The <i>Cancel</i> button was selected and <i>s_buttonLayout</i> is 'YesNoCancel.

If *s_buttonLayout* is 'UserDefined and If *s_dialogStyle* is 'modal, this function returns one of the following values when a button is clicked:

<i>buttonIndex</i>	The index of the button that is clicked, if the button does not have a callback associated with it. The first button in the dialog box is 1, the second is 2, and so on.
<i>callbackReturnVal</i>	The return value of the callback that is associated with the button that is clicked.

If *s_buttonLayout* is 'UserDefined and *s_dialogStyle* is 'modeless, this function returns one of the following values immediately after the dialog box is displayed:

<i>t</i>	There is no error.
<i>nil</i>	There is an error.

Examples

Creates a reminder list and a warning dialog box for each item of the list.

```
;;; creating the reminder list
trReminders = '( "Project Meeting 10-11" "Workout 12-1" "Team Meeting 3-4" )
foreach( reminderText trReminders ; for each reminder
  hiDisplayAppDBox(
    ?name      gensym( 'trReminderDialogBox ) ; unique variable
    ?dboxBanner "Reminder"
    ?dboxText   reminderText
    ?callback   sprintf( nil "trReminderCB( \"%s\" )"  reminderText ) ;
dynamically built callback
    ?dialogType hicWarningDialog
```

Cadence User Interface SKILL Reference

Dialog Boxes

```
        ?dialogStyle 'modeless
        ?buttonLayout 'YesNo
    )
) ; foreach
procedure( trReminderCB( reminderText )
    printf( "%s completed" reminderText )
)
```

Creates different types of modal dialog boxes.

```
trDialogTypes = '(
    hicWarningDialog hicErrorDialog hicInformationDialog
    hicMessageDialog hicQuestionDialog hicWorkingDialog
)
```

Related Topics

[General Dialog Box Functions](#)

[hiDisplayUserDBox](#)

[hiSetDBoxDefaultLocation](#)

[hiGetDBoxDefaultLocation](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDisplayUserDBox

```
hiDisplayUserDBox(  
    ?name s_dboxHandle  
    [ ?dboxBanner t_dboxBanner ]  
    [ ?dboxText t_dboxText ]  
    [ ?location l_location ]  
    [ ?dialogType x_dialogType ]  
    [ ?dialogStyle s_dialogStyle ]  
    [ ?defaultButton x_defaultButton ]  
    ?buttons g_buttons  
    [ ?callbacks g_callbacks ]  
    [ ?dontPopdowns g_dontPopdowns ]  
    [ ?help g_help ]  
)  
=> buttonIndex / callbackReturnVal / t / nil
```

Description

Creates and displays a dialog box. This function lets you customize buttons in the dialog box. You can also choose whether the dialog box is removed from the screen or not when a button is pressed. Using this function is equivalent to using `hiDisplayAppDBox` with `s_buttonLayout` set to 'UserDefined.

Arguments

`?name s_dboxHandle`

SKILL handle to the dialog box. When the dialog box is dismissed, this symbol is reset to `nil`.

`?dboxBanner t_dboxBanner`

Text that appears within the window manager banner of the dialog box.

`?dboxText t_dboxText`

Text that is displayed in the dialog box.

`?location l_location`

Cadence User Interface SKILL Reference

Dialog Boxes

The location of the dialog box on the screen, specified as the x and y coordinates of the upper left corner of the box:

`list(x-coordinate y-coordinate)` . If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

?dialogType *x_dialogType*

One of the following dialog box types, which determine the appearance of the dialog box:

- `hicWarningDialog(default)`
- `hicErrorDialog`
- `hicInformationDialog`
- `hicMessageDialog`
- `hicQuestionDialog`
- `hicWorkingDialog`

?dialogStyle *s_dialogStyle*

One of the following symbols, which determines the blocking behavior of the dialog box:

`'modal`

Blocks access to the application while it is displayed. The user must dismiss the dialog box in order to continue using the application. (Access to window manager commands or other applications is not blocked.)

`'modeless`

Does not block access to the application. The user can continue using the application without having to close the dialog box.

The default value of this argument is `'modal`.

?defaultButton *x_defaultButton*

Cadence User Interface SKILL Reference

Dialog Boxes

The default button to be used when the *Return* key is pressed. Specify the default button by its position. The first button you specify in *g_buttons* is 1, the second one is 2, and so on.

If you do not specify a value or if you specify an invalid value (for example, if you specify 5 when the dialog box only has three buttons), the first button is the default.

?buttons *g_buttons*

List of button labels that you want to display in the dialog box. Specify this argument as a list of strings. You can specify any string for a button, for example, "Overwrite", "Update", or "Cancel".

A *Help* button is not added by default to your list of buttons. If you want a *Help* button, specify it in the *g_buttons* list and specify a callback for it in the *g_callbacks* argument. You can also invoke the standard Doc Assistant mechanism when the user clicks *Help*; see the description of *g_callbacks* for more information.

?callbacks *g_callbacks*

Cadence User Interface SKILL Reference

Dialog Boxes

List of callbacks that determine what happens when the buttons in the dialog box are clicked. The list of callbacks corresponds to the list of buttons specified for the *g_buttons* argument. The first callback is called when the first button in *g_buttons* is clicked, the second is called when the second button is clicked, and so on. If you do not want to specify a callback for a button, use *nil*.

The default value of this argument is *nil* for all buttons.

If you specified a *Help* button in *g_buttons* and if you want the standard Doc Assistant mechanism to be invoked when the user clicks the button, use the following procedure as the callback for the *Help* button:

```
procedure(myHelp())
hiHelp('dbox dBoxName)
```

If *s_dialogStyle* is *'modeless*, you should specify a callback for each button, as the function returns *t* immediately after the dialog box is displayed (if there is no error). You can write the callback in a way that it returns the button index. For example, write the callback as:

```
procedure( myCallback(button)
when (button == 1 "Overwrite selected")
```

and, in the *hiDisplayUserDBox* function, pass the button index to the callback:

```
?callback ' ("myCallback(1) " "myCallback(2) "
"myCallback(3) ")
```

?dontPopdowns g_dontPopdowns

A list, each element of which is *t* or *nil*, that determines whether the dialog box is closed when a button is clicked. Specify *t* if you want to continue to display the dialog box when the button is clicked; specify *nil* if you want the dialog box to be closed when the button is clicked.

The items in the list correspond to the buttons in *g_buttons*. The first item in *g_dontPopdowns* determines the behavior of the first button in *g_buttons*, the second item determines the behavior of the second button, and so on.

The default value of this argument is *nil* for all buttons.

Cadence User Interface SKILL Reference

Dialog Boxes

`?help g_help` This argument works the same way as the `g_help` argument of `hiDisplayAppDBox` if you specify a *Help* button in the `g_buttons` argument and if the button calls `hiHelp( 'dBox dBoxName)`. Otherwise, this argument is not used.

For more information about specifying a *Help* button and a callback for it, see the description of the `g_buttons` and `g_callbacks` arguments.

Values Returned

If `s_dialogStyle` is 'modal, this function returns one of the following values when a button is clicked:

<code>buttonIndex</code>	The index of the button that is clicked, if the button does not have a callback associated with it. The first button in the dialog box is 1, the second is 2, and so on.
<code>callbackReturnVal</code>	The return value of the callback that is associated with the button that is clicked.

If `s_dialogStyle` is 'modeless, this function returns one of the following values immediately after the dialog box is displayed:

<code>t</code>	There is no error.
<code>nil</code>	There is an error.

Related Topics

[General Dialog Box Functions](#)

[hiDisplayAppDBox](#)

[hiSetDBoxDefaultLocation](#)

[hiGetDBoxDefaultLocation](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDBoxCancel

```
hiDBoxCancel(  
    g_dboxID  
    [ g_cancelFromCancel ]  
)  
=> t / nil
```

Description

Function logged when either the *Cancel* or the *No* button is pressed on a dialog box. This function can also be called directly from SKILL to simulate pressing *No* or *Cancel*.

If *g_cancelFromCancel* is non-*nil* in the logged entry, that indicates that the *Cancel* button was pressed when both *No* and *Cancel* buttons are present in the dialog box.

Arguments

<i>g_dboxID</i>	ID of the dialog box being canceled.
<i>g_cancelFromCancel</i>	Optional argument to be used when a form has both a <i>No</i> and <i>Cancel</i> button. The valid values for this argument are: Any non-<i>nil</i> value: Simulate pressing <i>Cancel</i> if the dialog box has a <i>No</i> button. <i>nil</i>: Simulate pressing <i>No</i> if the dialog box has both <i>No</i> and <i>Cancel</i> buttons, otherwise simulate pressing <i>Cancel</i>. The default value of this argument is <i>nil</i>.

Values Returned

<i>t</i>	The function was logged.
<i>nil</i>	<i>g_dboxID</i> is invalid.

Related Topics

[General Dialog Box Functions](#)

[hiDisplayAppDBox](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDBoxOK

Cadence User Interface SKILL Reference

Dialog Boxes

hiDBoxOK

```
hiDBoxOK(  
    g_dboxID  
    [ g_dontUnmanage ]  
)  
=> t / nil
```

Description

Function logged when the *OK*, *Yes* or *Done* button is pressed in a dialog box. This function can also be called directly from SKILL to simulate pressing *OK*, *Yes* or *Done*.

Do not use the *g_dontUnmanage* optional argument, as it is only intended for special quick help dialog boxes that are currently unused.

Arguments

<i>g_dboxID</i>	ID of the dialog box.
<i>g_dontUnmanage</i>	Do not use this argument.

Values Returned

<i>t</i>	The <i>OK</i> , <i>Yes</i> , or <i>Done</i> button was selected or an <code>hiDBoxOK</code> call was made from SKILL.
<i>nil</i>	Returned in all other cases.

Related Topics

[General Dialog Box Functions](#)

[hiDisplayAppDBox](#)

[hiDBoxCancel](#)

hiSetDBoxDefaultLocation

```
hiSetDBoxDefaultLocation(  
    l_location  
    [ g_autoUpdate { t | nil } ]  
)  
=> t / nil
```

Description

Sets the default location for all dialog boxes created with `hiDisplayAppDBox`, `hiDisplayUserDBox`, `hiDisplayProgressBox`, and `hiDisplayFileDialog`. This setting is used if the `l_location` argument of those functions has not been set or has been set incorrectly.

If both this function and the `l_location` argument of a dialog box function has been set, the `l_location` argument is used. If neither has been set, the location is determined by the underlying toolkit that implements the dialog box and is usually the center of the screen.

Argument

<code>l_location</code>	The location of the dialog box on the screen, specified as the <code>x</code> and <code>y</code> coordinates of the upper left corner of the dialog box: <code>list(x-coordinate y-coordinate)</code>
<code>g_autoUpdate</code>	When set to <code>t</code> , each time a dialog box created using a HI function is moved from its default location to a new location, the default location of the dialog box is updated to the new location automatically. The new location that is saved is the position of the dialog box when it is dismissed. The default is <code>nil</code> .

Value Returned

<code>t</code>	The default location was set.
<code>nil</code>	There was an error.

Related Topics

General Dialog Box Functions

hiDisplayAppDBox

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetDBoxDefaultLocation

hiDisplayUserDBox

hiDisplayFileDialog

hiDisplayProgressBox

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetDBoxDefaultLocation

```
hiGetDBoxDefaultLocation()  
=> l_location / nil
```

Description

Gets the default location of all dialog boxes, which has been set with the `hiSetDBoxDefaultLocation` function.

Arguments

None

Value Returned

<code>l_location</code>	The default location, specified as the <code>x</code> and <code>y</code> coordinates of the upper left corner of the dialog box: <code>list(x-coordinate y-coordinate)</code>
<code>nil</code>	The default location is not set.

Related Topics

[General Dialog Box Functions](#)

[hiSetDBoxDefaultLocation](#)

[hiDisplayAppDBox](#)

[hiDisplayUserDBox](#)

[hiDisplayFileDialog](#)

[hiDisplayProgressBox](#)

File Selection Dialog Box Functions

This section describes the following functions that let you create file selection dialog boxes:

- [hiDisplayFileDialog](#)
- [hiFileDialogSelection](#)
- [hiFileDialogSetSelection](#)
- [hiFileDialogDone](#)
- [hiGetExistingFile](#)
- [hiGetExistingFiles](#)
- [hiGetAnyFile](#)
- [hiGetExistingDirectory](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDisplayFileDialog

```
hiDisplayFileDialog(  
    [ ?dialogName s_dialogName ]  
    [ ?workingDir t_workingDir ]  
    [ ?filter t_filter ]  
    [ ?mode s_mode ]  
    [ ?modal g_modal ]  
    [ ?defaultValue t_defaultValue ]  
    [ ?callback t_callback ]  
    [ ?userData s_userData ]  
    [ ?caption t_caption ]  
    [ ?location l_location ]  
    [ ?acceptMode s_acceptMode ]  
    [ ?labelText t_labelText ]  
    [ ?confirmOverwrite g_confirmOverwrite ]  
)  
=> l_userSelection / nil
```

Description

Creates and displays a file selection dialog box, which lets users select a file or directory. The dialog box is destroyed when it is removed from the screen.

If *s_dialogName* is a valid file dialog box handle, which indicates that the dialog is already displayed, the dialog is brought to the front of the screen.

The file dialog box can be either modal (blocking) or modeless (non-blocking). If the dialog box is modal, the application that displays it cannot continue until the user clicks one of the buttons on the dialog box. If the dialog box is modeless, the application can continue.

With the file dialog SKILL functions, an application can display a file dialog, retrieve the file or directory selections made by the user, and replay the process. Full interactive functionality of the file dialog box is not supported programmatically—for example, while a user can create directories or sort files in the file dialog box interactively, these actions are not logged.

Arguments

?dialogName *s_dialogName*

SKILL symbol that is the handle to the file dialog box.

When the dialog box is destroyed (that is, when the `hiFileDialogDone` command is executed), this symbol is reset to `nil`. Do not reference a deleted file dialog; it can cause undefined behavior.

Cadence User Interface SKILL Reference

Dialog Boxes

?workingDir *t_workingDir*

Text string specifying the working directory. This is the directory that appears in the *Look in* field.

The default value of this argument is " ." (the current working directory).

?filter *t_filter*

Text string specifying the filters that can be used in the *File type* field. For example:

```
"Images (*.png *.xpm *.jpg) "
```

If you want to specify multiple filters, separate them with two semicolons. For example:

```
"Images (*.png *.xpm *.jpg);;Text Files  
(*.txt);;XML Files (*.xml) "
```

If you do not specify a filter, the default is "All Files (*)".

If *s_mode* is 'directory or 'directoryOnly, this argument is ignored.

?mode *s_mode*

Symbol specifying what the user is allowed to select in the file dialog box. Legal values are:

- 'anyFile: Selects any file, whether it exists or not.

This option is typically used for "Save As" dialog boxes.

- 'existingFile: Selects an existing file.

This option is typically used for Open File dialog boxes.

- 'existingFiles: Selects zero or more existing files

- This option is typically used for Open File dialog boxes.

- 'directory: Selects a directory; the file dialog displays both files and directories

- 'directoryOnly: Selects a directory; the file dialog displays both files and directories

Cadence User Interface SKILL Reference

Dialog Boxes

The default value is `'existingFile`.

If `s_mode` is set to `'anyFile` and `s_acceptMode` equals `'save`, then the first suffix in the selected name filter is added to the filename if no other suffix has been specified in the file dialog. For example, if the selected name filter is `"Images (*.png *.xpn *.jpg)"`, and any type-in field is `"/tmp/foobar"`, then `hiFileDialogSelection` is returned `("/tmp/foobar.png")`. Note however that if the type-in field is `"/tmp/foobar.txt"`, Qt considers that a suffix already exists and HI would not be able to return `"/tmp/foobar.txt.png"`.

`?modal g_modal`

`non-nil` or `nil`. Specify any non-`nil` value if you want the dialog box to be modal (blocking) or `nil` if you want the dialog box to be modeless (non-blocking).

If the dialog box is modal, calls to `hiDisplayFileDialog` do not return until the user clicks one of the buttons on the dialog box. In this case, you can either handle the user selection immediately, or in a callback as you would with a modeless dialog box.

If the dialog box is modeless, calls to `hiDisplayFileDialog` return a value of `nil` immediately after the dialog is displayed. Therefore, you must specify a callback to handle the user selection.

`?defaultValue t_defaultValue`

Text string specifying a file or directory that is selected by default.

`?callback t_callback`

Cadence User Interface SKILL Reference

Dialog Boxes

Text string specifying the SKILL function to be called when the user makes a selection in the dialog box.

The callback function is passed either two or three arguments. The first argument is *&_dialog*, the handle to the file dialog box.

The second argument is *s_event*, indicating which button the user clicked after making a selection. *s_event* is one of 'done or 'cancel, corresponding to the *Open/Save/OK* and *Cancel* buttons.

The third argument, *l_userData*, is passed only if you specify the *s_userData* argument to this function.

If the dialog box is modeless, you must specify a callback to handle the user selection, as `hiDisplayFileDialog` returns *nil* immediately after the dialog box is displayed.

?userData *s_userData*

Symbol specifying user data for the callback. If you specify this argument, an additional argument, *userData*, is passed to the callback function. Otherwise, the callback is passed only two arguments.

?caption *t_caption*

Text string that is displayed in the file dialog box banner.

?location *l_location*

The location of the dialog box on the screen, specified as a list of the x and y coordinates of the upper left corner of the box:

`list(x-coordinate y-coordinate)`

If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

?acceptMode *s_acceptMode*

Cadence User Interface SKILL Reference

Dialog Boxes

One of the following symbols: 'open or 'save, that specify the button to display on the dialog box. Specify 'open if you are creating a dialog for opening files (an *Open* button appears on the dialog) or 'save if you are creating a dialog for saving files (a *Save* button appears on the dialog).

Default value: 'save if ?mode is 'anyFile, else 'open

?labelText *t_labelText*

Specify the optional argument ?labelText as an association list of s_key/t_value pairs. Accepted keys are 'LookIn, 'FileName, 'FileType, 'Accept, and 'Reject. The 'Accept and 'Reject keys are for specifying *Open/Save* and *Cancel* button labels respectively. The default value is nil.

?confirmOverwrite *g_confirmOverwrite*

Set this optional argument to nil if you do not want to display a dialog box that asks for a file overwrite confirmation when an existing file is selected , and argument *s_mode* equals 'anyFile.

Set it to t to display the overwrite confirmation dialog box but this only applies when the argument *s_mode* equals 'anyFile.

Default value: t

Cadence User Interface SKILL Reference

Dialog Boxes

Values Returned

<i>l_userSelection</i>	Returns <i>l_userSelection</i> if the file dialog box is modal and the user makes a selection. <i>l_userSelection</i> is a list of two elements, each of which is a string: <pre>("dir" "file")</pre> <i>dir</i> is a path relative to <i>t_workingDir</i> if the selection is either in <i>t_workingDir</i> or in a directory ascending or descending from <i>t_workingDir</i>. Otherwise, <i>dir</i> is an absolute path. <i>file</i> is the file that was selected. If the <i>s_mode</i> argument is 'directory or 'directoryOnly, <i>file</i> is an empty string. If <i>s_mode</i> is 'existingFiles, <i>l_userSelection</i> is a list of directory-file pairs, such as: <pre>(("dir1" "file1") ("dir2" "file2") ...)</pre>
<i>nil</i>	Returns <i>nil</i> if the file dialog box is modeless, or if the file dialog box is modal and has been canceled.

Example

```
( hiDisplayFileDialog
  ?dialogName 'myFileDialog
  ?callback "myFileDialogCallback"
  ?caption "Open Existing File"
  ?workingDir "/usr/local"
  ?mode 'existingFile
  ?modal nil
  ?userData 'myUserData
)

;; ...

( defun myFileDialogCallback ( dialog event userData "&ss" )
  ( if ( eq event 'done )
    ( printf "hiFileDialogSelection returns, %L\n"
      (hiFileDialogSelection dialog ) )
    ( printf "myFileDialog is canceled\n" ) ) )
```

The callback gets invoked as

```
( myFileDialogCallback myFileDialog 'done 'myUserData )
```

or

```
( myFileDialogCallback myFileDialog 'cancel 'myUserData )
```

depending on which button was clicked.

Cadence User Interface SKILL Reference

Dialog Boxes

Related Topics

[File Selection Dialog Box Functions](#)

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiFileDialogDone](#)

[hiGetExistingFile](#)

[hiGetExistingFiles](#)

[hiGetAnyFile](#)

[hiGetExistingDirectory](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiFileDialogSelection

```
hiFileDialogSelection(  
    &_dialogName  
)  
=> l_userSelection / nil
```

Description

Returns the selection made by the user in the file dialog box. This function is typically called from a user callback.

Argument

<i>&_dialogName</i>	The handle to the file dialog box, which was specified as the <i>s_dialogName</i> argument of <code>hiDisplayFileDialog</code> when the dialog box was created.

Values Returned

<i>l_userSelection</i>	<i>l_userSelection</i> is a list of two elements, each of which is a string: <pre>("dir" "file")</pre> <i>dir</i> is a path relative to the working directory if the selection is either in the working directory or in a directory ascending or descending from it. (The working directory is specified in the <i>t_workingDir</i> argument of <code>hiDisplayFileDialog</code> .) Otherwise, <i>dir</i> is an absolute path. <i>file</i> is the file that was selected. If the <i>s_mode</i> argument of <code>hiDisplayFileDialog</code> is 'directory or 'directoryOnly, <i>file</i> is an empty string. If <i>s_mode</i> is 'existingFiles, <i>l_userSelection</i> is a list of directory-file pairs: <pre>(("dir" "file") ("dir" "file") ...)</pre>
<i>nil</i>	Nothing was selected in the file dialog box or <i>&_dialogName</i> is not a valid file dialog ID.

Cadence User Interface SKILL Reference

Dialog Boxes

Related Topics

[File Selection Dialog Box Functions](#)

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiFileDialogDone](#)

[hiGetExistingFile](#)

[hiGetExistingFiles](#)

[hiGetAnyFile](#)

[hiGetExistingDirectory](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiFileDialogSetSelection

```
hiFileDialogSetSelection(  
    &_dialogName  
    l_selection  
    s_event  
)  
=> t / nil
```

Description

This function is logged when a selection is made from a file dialog.

Applications do not usually call this function.

Arguments

<i>&_dialogName</i>	The file dialog identity that is set into the <i>s_dialogName</i> argument of <code>hiDisplayFileDialog</code> when the file dialog is created.
<i>l_selection</i>	List of selections that can be in the same format as the return value of <code>hiDisplayFileDialog</code> , which is a list of lists of directory-path/file-name pairs. Here, the directory-path should be the same for every file, or a list of directory-path and file-name, or a list of directory-paths and a list of file names. If <i>l_selection</i> includes multiple files, only the first file in the list is shown in the file dialog.
<i>s_event</i>	Symbol indicating which the button the user clicked after making a selection. Legal values are 'done, 'apply, and 'cancel, which correspond to the <i>Open/Save/OK</i> , <i>Apply</i> , and <i>Cancel</i> buttons.

Values Returned

t	The argument <i>&_dialogName</i> is a valid file dialog identity.
nil	<i>&_dialogName</i> is not a valid file dialog ID.

Examples

```
( hiFileDialogSetSelection hiEditFileDialog '( "/tmp" "foo.txt" ) 'done )
```

Cadence User Interface SKILL Reference

Dialog Boxes

```
( hiFileDialogSetSelection hiEditFileDialog '( ( "/tmp" "foo.txt" ) ( "/tmp"
"bar.txt" ) ) 'done )
( hiFileDialogSetSelection hiEditFileDialog '( "/tmp" ( "foo.txt" "bar.txt" ) )
'done )
```

Related Topics

File Selection Dialog Box Functions

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiFileDialogDone](#)

[hiGetExistingFile](#)

[hiGetExistingFiles](#)

[hiGetAnyFile](#)

[hiGetExistingDirectory](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiFileDialogDone

```
hiFileDialogDone(  
    &_dialogName  
    g_accept  
)  
=> t / nil
```

Description

This function is logged when a file dialog box is dismissed. Applications do not need to call this function; it is provided for replay purposes.

This function destroys the file dialog box and resets the value of the *&_dialogName* symbol to *nil*. Do not reference a deleted file dialog box; it can cause undefined behavior.

Arguments

<i>&_dialogName</i>	The handle to the file dialog box, which was specified as the <i>s_dialogName</i> argument of <i>hiDisplayFileDialog</i> when the dialog box was created.
<i>g_accept</i>	<i>nil</i> if the file dialog box was canceled, non- <i>nil</i> otherwise.

Values Returned

<i>t</i>	The function was logged.
<i>nil</i>	<i>&_dialogName</i> is not a valid file dialog ID.

Related Topics

[File Selection Dialog Box Functions](#)

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiGetExistingFile](#)

[hiGetExistingFiles](#)

[hiGetAnyFile](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetExistingDirectory

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetExistingFile

```
hiGetExistingFile(  
    [ ?workingDir t_workingDir ]  
    [ ?filter t_filter ]  
    [ ?dialogName s_dialogName ]  
    [ ?defaultValue t_defaultValue ]  
    [ ?callback t_callback ]  
    [ ?userData s_userData ]  
    [ ?caption t_caption ]  
    [ ?location l_location ]  
    [ ?modal g_modal ]  
)  
=> l_userSelection / nil
```

Description

Calls `hiDisplayFileDialog` with the mode set to 'existingFile'. Use this function to ask the user to select a file to open.

This function is provided as a quick and convenient way to display a file dialog by specifying only the working directory; default values for the rest of the arguments are provided.

Arguments

?workingDir t_workingDir

Text string specifying the working directory. This is the directory that appears in the *Look in* field.

The default value of this argument is "." (the current working directory).

filter t_filter

Text string specifying the filters that can be used in the *File type* field. For example:

```
"Images (*.png *.xpm *.jpg)"
```

If you want to specify multiple filters, separate them with two semicolons. For example:

```
"Images (*.png *.xpm *.jpg);;Text Files  
(*.txt);;XML Files (*.xml)"
```

If you do not specify a filter, the default is "All Files (*)".

?dialogName s_dialogName

Cadence User Interface SKILL Reference

Dialog Boxes

SKILL symbol that is the handle to the file dialog box.

When the dialog box is destroyed (that is, when the `hiFileDialogDone` command is executed), this symbol is reset to `nil`. Do not reference a deleted file dialog box; it can cause undefined behavior.

?defaultValue t_defaultValue

Text string specifying the file that is selected by default.

?callback t_callback

Text string specifying the SKILL function to be called when the user makes a selection in the dialog box.

The callback function is passed either two or three arguments. The first argument is `&_dialog`, the handle to the file dialog box.

The second argument is `s_event`, indicating which button the user clicked after making a selection. `s_event` is one of 'done', 'apply', or 'cancel', corresponding to the *Open*, *Apply*, and *Cancel* buttons.

The third argument, `l_userData`, is passed only if you specify the `s_userData` argument to this function.

Note: If the dialog box is modeless, you must specify a callback to handle the user selection, as this function returns `nil` immediately after the dialog box is displayed.

?userData s_userData

Symbol specifying user data for the callback. If you specify this argument, an additional argument, `userData`, is passed to the callback function. Otherwise, the callback is passed only two arguments.

?caption t_caption

Specifies caption which gets displayed in the file dialog box banner. The default caption is `Choose a File`.

?location l_location

Cadence User Interface SKILL Reference

Dialog Boxes

The location of the dialog box on the screen, specified as a list of the x and y coordinates of the upper left corner of the box:

```
list(x-coordinate y-coordinate)
```

If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

?modal g_modal

Specifies any non-`nil` value if you want the dialog box to be modal (blocking) or `nil` if you want the dialog box to be modeless (non-blocking).

If the dialog box is modal, calls to `hiDisplayFileDialog` do not return until the user clicks one of the buttons on the dialog box. In this case, you can either handle the user selection immediately, or in a callback as you would with a modeless dialog box.

If the dialog box is modeless, calls to `hiDisplayFileDialog` return a value of `nil` immediately after the dialog is displayed. Therefore, you must specify a callback to handle the user selection.

Cadence User Interface SKILL Reference

Dialog Boxes

Values Returned

<i>l_userSelection</i>	Returns <i>l_userSelection</i> if the file dialog box is modal and the user makes a selection. <i>l_userSelection</i> is a list of two elements, each of which is a string: <code>("dir" "file")</code> <i>dir</i> is a path relative to <i>t_workingDir</i> if the selected file is either in <i>t_workingDir</i> or in a directory ascending or descending from <i>t_workingDir</i> . Otherwise, <i>dir</i> is an absolute path. <i>file</i> is the file that was selected.
<i>nil</i>	Returns <i>nil</i> if the file dialog box is modeless, or if the file dialog box is modal and has been canceled.

Example

```
(edit (buildString (hiGetExistingFile) "/"))
```

Related Topics

File Selection Dialog Box Functions

hiFileDialogSelection

hiFileDialogSetSelection

hiFileDialogDone

hiGetExistingFiles

hiGetAnyFile

hiGetExistingDirectory

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetExistingFiles

```
hiGetExistingFiles(  
    [ ?workingDir t_workingDir ]  
    [ ?filter t_filter ]  
    [ ?dialogName s_dialogName ]  
    [ ?callback t_callback ]  
    [ ?userData s_userData ]  
    [ ?caption t_caption ]  
    [ ?location l_location ]  
    [ ?modal g_modal ]  
)  
=> l_userSelection / nil
```

Description

Calls `hiDisplayFileDialogBox` with the mode set to 'existingFiles'. Use this function to ask the user to select the files to open.

This function is provided as a quick and convenient way to display a file dialog by specifying only the working directory; default values for the rest of the arguments are provided.

Arguments

`?workingDir t_workingDir`

Text string specifying the working directory. This is the directory that appears in the *Look in* field.

The default value of this argument is "." (the current working directory).

`?filter t_filter`

Text string specifying the filters that can be used in the *File type* field. For example:

```
"Images (*.png *.xpm *.jpg)"
```

If you want to specify multiple filters, separate them with two semicolons. For example:

```
"Images (*.png *.xpm *.jpg);;Text Files  
(*.txt);;XML Files (*.xml)"
```

If you do not specify a filter, the default is "All Files (*)".

`?dialogName s_dialogName`

Cadence User Interface SKILL Reference

Dialog Boxes

SKILL symbol that is the handle to the file dialog box.

When the dialog box is destroyed (that is, when the `hiFileDialogDone` command is executed), this symbol is reset to `nil`. Do not reference a deleted file dialog box; it can cause undefined behavior.

?callback *t_callback*

Text string specifying the SKILL function to be called when the user makes a selection in the dialog box.

The callback function is passed either two or three arguments. The first argument is `&_dialog`, the handle to the file dialog box.

The second argument is `s_event`, indicating which button the user clicked after making a selection. `s_event` is one of 'done, 'apply, or 'cancel, corresponding to the *Open*, *Apply*, and *Cancel* buttons.

The third argument, `l_userData`, is passed only if you specify the `s_userData` argument to this function.

If the dialog box is modeless, you must specify a callback to handle the user selection, as this function returns `nil` immediately after the dialog box is displayed.

?userData *s_userData*

Symbol specifying user data for the callback. If you specify this argument, an additional argument, `userData`, is passed to the callback function. Otherwise, the callback is passed only two arguments.

?caption *t_caption*

Text string that gets displayed in the file dialog box banner. The default caption is `Choose File(s)`.

?location *l_location*

Cadence User Interface SKILL Reference

Dialog Boxes

The location of the dialog box on the screen, specified as a list of the x and y coordinates of the upper left corner of the box:

```
list(x-coordinate y-coordinate)
```

If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

?modal *g_modal*

non-`nil` or `nil`. Specify any non-`nil` value if you want the dialog box to be modal (blocking) or `nil` if you want the dialog box to be modeless (non-blocking).

If the dialog box is modal, calls to `hiDisplayFileDialog` do not return until the user clicks one of the buttons on the dialog box. In this case, you can either handle the user selection immediately, or in a callback as you would with a modeless dialog box.

If the dialog box is modeless, calls to `hiDisplayFileDialog` return a value of `nil` immediately after the dialog is displayed. Therefore, you must specify a callback to handle the user selection.

Values Returned

l_userSelection

Returns *l_userSelection* if the file dialog box is modal and the user makes a selection.

l_userSelection is a list of lists. Each list is a directory-file pair, indicating the files that were selected.

```
(("dir" "file") ("dir" "file") ...)
```

dir is a path relative to *t_workingDir* if the selected file is either in *t_workingDir* or in a directory ascending or descending from *t_workingDir*. Otherwise, *dir* is an absolute path.

file is the file that was selected. .

`nil`

Returns `nil` if the file dialog box is modeless, or if the file dialog box is modal and has been canceled.

Cadence User Interface SKILL Reference

Dialog Boxes

Example

```
hiGetExistingFiles( ?filter "*il" )
```

Related Topics

[File Selection Dialog Box Functions](#)

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiFileDialogDone](#)

[hiGetExistingFiles](#)

[hiGetAnyFile](#)

[hiGetExistingDirectory](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetAnyFile

```
hiGetAnyFile(  
    [ ?workingDir t_workingDir ]  
    [ ?filter t_filter ]  
    [ ?dialogName s_dialogName ]  
    [ ?defaultValue t_defaultValue ]  
    [ ?callback t_callback ]  
    [ ?userData s_userData ]  
    [ ?caption t_caption ]  
    [ ?location l_location ]  
    [ ?modal g_modal ]  
    [ ?acceptMode s_acceptMode ]  
    [ ?confirmOverwrite g_confirmOverwrite ]  
)  
=> l_userSelection / nil
```

Description

Calls `hiDisplayFileDialogBox` with the mode set to 'anyFile'. Use this function to prompt the user to specify the name of a file to open or save.

This function is provided as a quick and convenient way to display a file dialog by specifying only the working directory; default values for the rest of the arguments are provided.

Arguments

`?workingDir t_workingDir`

Text string specifying the working directory. This is the directory that appears in the *Look in* field.

The default value of this argument is "." (the current working directory).

`?filter t_filter`

Text string specifying the filters that can be used in the *File type* field. For example:

```
"Images (*.png *.xpm *.jpg) "
```

If you want to specify multiple filters, separate them with two semicolons. For example:

```
"Images (*.png *.xpm *.jpg);;Text Files  
(*.txt);;XML Files (*.xml) "
```

If you do not specify a filter, the default is "All Files (*)".

`?dialogName s_dialogName`

Cadence User Interface SKILL Reference

Dialog Boxes

SKILL symbol that is the handle to the file dialog box. When the dialog box is destroyed (that is, when the `hiFileDialogDone` command is executed), this symbol is reset to `nil`. Do not reference a deleted file dialog box; it can cause undefined behavior.

?defaultValue *t_defaultValue*

Text string specifying a file that is selected by default.

?callback *t_callback*

Text string specifying the SKILL function to be called when the user makes a selection in the dialog box.

The callback function is passed either two or three arguments. The first argument is `&_dialog`, the handle to the file dialog box.

The second argument is `s_event`, indicating which button the user clicked after making a selection. `s_event` is one of 'done', 'apply', or 'cancel', corresponding to the *Save*, *Apply*, and *Cancel* buttons.

The third argument, `l_userData`, is passed only if you specify the `s_userData` argument to this function.

Note: If the dialog box is modeless, you must specify a callback to handle the user selection, as this function returns `nil` immediately after the dialog box is displayed.

?userData *s_userData*

Symbol specifying user data for the callback. If you specify this argument, an additional argument, `userData`, is passed to the callback function. Otherwise, the callback is passed only two arguments.

?caption *t_caption*

Text string that gets displayed in the file dialog box banner. The default caption is *Specify a File*.

?location *l_location*

Cadence User Interface SKILL Reference

Dialog Boxes

The location of the dialog box on the screen, specified as a list of the x and y coordinates of the upper left corner of the box:

```
list(x-coordinate y-coordinate)
```

If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

`?modal g_modal`

`non-nil` or `nil`. Specify any `non-nil` value if you want the dialog box to be modal (blocking) or `nil` if you want the dialog box to be modeless (non-blocking).

If the dialog box is modal, calls to `hiDisplayFileDialog` do not return until the user clicks one of the buttons on the dialog box. In this case, you can either handle the user selection immediately, or in a callback as you would with a modeless dialog box.

If the dialog box is modeless, calls to `hiDisplayFileDialog` return a value of `nil` immediately after the dialog is displayed. Therefore, you must specify a callback to handle the user selection.

`?acceptMode s_acceptMode`

One of the following symbols: `'open` or `'save`, that specify the button to display on the dialog box. Specify `'open` if you are creating a dialog for opening files (an *Open* button appears on the dialog) or `'save` if you are creating a dialog for saving files (*Save* button appears on the dialog).

Default value: `'save`

`?confirmOverwrite g_confirmOverwrite`

Set this optional argument to `nil` if you do not want to display a dialog box that asks for a file overwrite confirmation when an existing file is selected, and argument `s_mode` equals `'anyFile`.

Set it to `t` to display the overwrite confirmation dialog box but this only applies when the argument `s_mode` equals `'anyFile`.

Default value: `t`

Cadence User Interface SKILL Reference

Dialog Boxes

Values Returned

<i>l_userSelection</i>	Returns <i>l_userSelection</i> if the file dialog box is modal and the user makes a selection. <i>l_userSelection</i> is a list of two elements, each of which is a string: <code>("dir" "file")</code> <i>dir</i> is a path relative to <i>t_workingDir</i> if the selected file is either in <i>t_workingDir</i> or in a directory ascending or descending from <i>t_workingDir</i> . Otherwise, <i>dir</i> is an absolute path. <i>file</i> is the file that was selected.
<i>nil</i>	Returns <i>nil</i> if the file dialog box is modeless, or if the file dialog box is modal and has been canceled.

Example

```
(edit (buildString (hiGetAnyFile) "/"))  
hiGetAnyFile(?acceptMode 'open)
```

Related Topics

[File Selection Dialog Box Functions](#)

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiFileDialogDone](#)

[hiGetExistingFiles](#)

[hiGetExistingDirectory](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetExistingDirectory

```
hiGetExistingDirectory(  
    [ ?workingDir t_workingDir ]  
    [ ?dirOnly g_dirOnly ]  
    [ ?dialogName s_dialogName ]  
    [ ?defaultValue t_defaultValue ]  
    [ ?callback t_callback ]  
    [ ?userData s_userData ]  
    [ ?caption t_caption ]  
    [ ?location l_location ]  
    [ ?modal g_modal ]  
)  
=> l_userSelection / nil
```

Description

Calls `hiDisplayFileDialogBox` with the mode set to 'dirOnly'. If the `?dirOnly` argument is `nil`, calls `hiDisplayFileDialogBox` with the mode set to 'directory'. Use this function to ask the user to select a directory.

This function is provided as a quick and convenient way to display a file dialog by specifying only the working directory; default values for the rest of the arguments are provided.

Arguments

`?workingDir t_workingDir`

Text string specifying the working directory. This is the directory that appears in the *Look in* field.

The default value of this argument is `" . "` (the current working directory).

`?dirOnly g_dirOnly`

`t` or `nil`. If you specify `t`, the file dialog displays only directories, if you specify `nil`, the file dialog displays both directories and files. The default value is `t`.

`?dialogName s_dialogName`

SKILL symbol that is the handle to the file dialog box.

When the dialog box is destroyed (that is, when the `hiFileDialogDone` command is executed), this symbol is reset to `nil`. Do not reference a deleted file dialog box; it can cause undefined behavior.

Cadence User Interface SKILL Reference

Dialog Boxes

?defaultValue *t_defaultValue*

Text string specifying the directory that is selected by default.

?callback *t_callback*

Text string specifying the SKILL function to be called when the user makes a selection in the dialog box.

The callback function is passed either two or three arguments. The first argument is *&_dialog*, the handle to the file dialog box.

The second argument is *s_event*, indicating which button the user clicked after making a selection. *s_event* is one of 'done, 'apply, or 'cancel, corresponding to the *OK*, *Apply*, and *Cancel* buttons.

The third argument, *l_userData*, is passed only if you specify the *s_userData* argument to this function.

If the dialog box is modeless, you must specify a callback to handle the user selection, as this function returns *nil* immediately after the dialog box is displayed.

?userData *s_userData*

Symbol specifying user data for the callback. If you specify this argument, an additional argument, *userData*, is passed to the callback function. Otherwise, the callback is passed only two arguments.

?caption *t_caption*

Text string that gets displayed in the file dialog box banner. The default caption is *Open a Directory*.

?location *l_location*

The location of the dialog box on the screen, specified as a list of the x and y coordinates of the upper left corner of the box:

list(x-coordinate y-coordinate)

If this argument is not specified or if it is specified incorrectly, then the default location set by [hiSetDBoxDefaultLocation](#) is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

Cadence User Interface SKILL Reference

Dialog Boxes

`?modal` *g_modal* non-`nil` or `nil`. Specify any non-`nil` value if you want the dialog box to be modal (blocking) or `nil` if you want the dialog box to be modeless (non-blocking).

If the dialog box is modal, calls to `hiDisplayFileDialog` do not return until the user clicks one of the buttons on the dialog box. In this case, you can either handle the user selection immediately, or in a callback as you would with a modeless dialog box.

If the dialog box is modeless, calls to `hiDisplayFileDialog` return a value of `nil` immediately after the dialog is displayed. Therefore, you must specify a callback to handle the user selection.

Values Returned

l_userSelection Returns *l_userSelection* if the file dialog box is modal and the user makes a selection.

l_userSelection is a list of two elements, each of which is a string:

```
("dir" "file")
```

dir is a path relative to *t_workingDir* if the selection is either in *t_workingDir* or in a directory ascending or descending from *t_workingDir*. Otherwise, *dir* is an absolute path.

file is an empty string.

`nil` Returns `nil` if the file dialog box is modeless, or if the file dialog box is modal and has been canceled.

Related Topics

[File Selection Dialog Box Functions](#)

[hiFileDialogSelection](#)

[hiFileDialogSetSelection](#)

[hiFileDialogDone](#)

[hiGetExistingFiles](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetAnyFile

hiGetExistingDirectory

Progress Dialog Box Functions

This topic provides a list of Cadence® SKILL functions associated with progress dialog boxes and progress bars.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiDisplayColorDialog](#)
- [hilsProgressBoxCancelled](#)
- [hiDisplayProgressBox](#)
- [hiGetProgress](#)
- [hiGetProgressTotalSteps](#)
- [hiGetProgressBarTotalSteps](#)
- [hilsProgressBoxCancelled](#)
- [hiResetProgressBox](#)
- [hiSetProgress](#)
- [hiSetProgressText](#)
- [hiSetProgressTotalSteps](#)
- [hiSetProgressAndText](#)
- [hiSetProgressButtonText](#)
- [hiSetProgressBannerText](#)
- [hiSetProgressBar](#)
- [hiSetProgressLabel](#)
- [hiShowProgressBar](#)

hiAdvanceProgressBarOneStep

```
hiAdvanceProgressBarOneStep(  
    w_window  
)  
=> t / nil
```

Description

Increments the progress bar by one step for the specified window.

Argument

<i>w_window</i>	The window from which you want to advance the progress bar by one step.

Values Returned

<i>t</i>	The progress bar successfully advanced by one step.
<i>nil</i>	There was an error.

Examples

Increments the progress bar for window (1).

```
hiAdvanceProgressBarOneStep( window ( 1 ) )  
=>t
```

Related Topics

[Progress Dialog Box Functions](#)

[hiShowProgressBar](#)

[hiSetProgressBar](#)

[hiSetProgressLabel](#)

[hiGetProgressBarTotalSteps](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiCancelProgressBox

```
hiCancelProgressBox(  
    g_dboxID  
)  
=> t / nil
```

Description

Programmatically cancels a progress dialog box. Calls the callback procedure for the dialog box, if any.

Argument

<i>g_dboxID</i>	The ID of the progress dialog box that you want to cancel.

Values Returned

<i>t</i>	The progress dialog box was canceled.
<i>nil</i>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDisplayProgressBox

```
hiDisplayProgressBox(  
    ?name s_name  
    [ ?banner t_banner ]  
    [ ?text t_text ]  
    [ ?location l_location ]  
    [ ?callback g_callback ]  
    [ ?totalSteps x_totalSteps ]  
    [ ?delay x_delay ]  
    [ ?autoClose g_autoClose ]  
)  
=> t / nil
```

Description

Creates a progress dialog box that displays the progress of a command. The progress dialog box is always modeless (non-blocking).

The progress dialog box has a progress bar, text that is displayed above the progress bar, and a *Cancel* button.

The progress bar displays the percentage of steps completed, you need to provide the total number of steps with the `?totalSteps` argument and the number of steps completed with the `hiSetProgress` function.

The button text can be changed with the `hiSetProgressButtonText` function.

You can also specify the location of the progress dialog box on the screen, the delay before it appears, and whether you want the dialog box to close automatically when the progress bar reaches 100%. You can also provide a callback function that is called when the dialog box is closed.

Arguments

<code>?name <i>s_name</i></code>	The progress box ID, specified as a symbol.
<code>?banner <i>t_banner</i></code>	The banner text that is displayed on the dialog box frame.
<code>?text <i>t_text</i></code>	The text that is displayed above the progress bar.
<code>?location <i>l_location</i></code>	

Cadence User Interface SKILL Reference

Dialog Boxes

The location of the dialog box on the screen, specified as the xy coordinates of the upper left corner of the box:

`list(x-coordinate y-coordinate)`

If this argument is not specified or if it is specified incorrectly, then the default location set by `hiSetDBoxDefaultLocation` is used. If the default location has not been set, then the toolkit that implements the dialog box determines the location (usually the center of the screen).

`?callback g_callback`

Text string specifying the SKILL procedure to be called when the dialog box is closed. The callback is called whenever the dialog box is closed—whether it is automatically closed when the progress bar reaches 100%, canceled with the *Cancel* button, or canceled programatically with the `hiCancelProgressBar` function.

`?totalSteps x_totalSteps`

The total number of steps on the progress bar. The default number of steps is 100.

`?delay x_delay`

The time delay, in milliseconds, before the progress box appears. The default value is 0.

If the delay is set and the command is executed in a time less than the delay, then the progress box is not displayed. This is useful in cases where you want to prevent the progress box from appearing, such as while loading a simple design.

`?autoClose g_autoClose`

`t` or `nil`, specifying whether the progress box should be closed when the progress bar reaches 100%. The default value is `t`.

Values Returned

`t`

The progress dialog box was created and displayed.

`nil`

There was an error.

Example

```
RECT_COUNT = 100000
hiDisplayProgressBar(?name 'pb
```

Cadence User Interface SKILL Reference

Dialog Boxes

```
?banner "Progress Box Test"
?text "Drawing Rectangles..."
?totalSteps RECT_COUNT
?autoClose nil
)
i=0
while(i<RECT_COUNT
    ;paint a rectangle on a graphics window
    i=i+1;
    if(i==RECT_COUNT/4 hiSetProgressText(pb "1/4 done !"))
    if(i==RECT_COUNT/2 hiSetProgressText(pb "1/2 done !"))
    if(i==RECT_COUNT*3/4 hiSetProgressText(pb "3/4 done !"))
    if(i==RECT_COUNT-1 hiSetProgressText(pb "Complete !!"))
    if(hiSetProgress(pb i)==nil i=RECT_COUNT)
)
```

Related Topics

Progress Dialog Box Functions

hilsProgressBoxCancelled

hiGetProgress

hiGetProgressTotalSteps

hiSetProgress

hiSetProgressText

hiSetProgressAndText

Cadence User Interface SKILL Reference

Dialog Boxes

hiGetProgress

```
hiGetProgress(  
    g_dboxID  
)  
=> x_currentProgress
```

Description

Gets the current progress (that is, the current step) from a progress dialog box.

Argument

g_dboxID The ID of the progress dialog box.

Value Returned

x_currentProgress The current progress (that is, the current step).

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiGetProgressTotalSteps](#)

[hiSetProgress](#)

[hiSetProgressText](#)

[hiSetProgressTotalSteps](#)

hiGetProgressBarCurrentStep

```
hiGetProgressBarCurrentStep(  
    w_window  
)  
=> t / nil
```

Description

Returns the current step of the progress bar from the window specified.

Argument

<i>w_window</i>	The window from which you want to get the current progress bar step.

Value Returned

<i>t</i>	The progress bar current step successfully returned.
<i>nil</i>	There was an error.

Example

```
hiGetProgressBarCurrentStep( window ( 1 ) )
```

Related Topics

[Progress Dialog Box Functions](#)

[hiShowProgressBar](#)

[hiSetProgressBar](#)

[hiSetProgressLabel](#)

[hiGetProgressBarTotalSteps](#)

hiGetProgressBarTotalSteps

```
hiGetProgressBarTotalSteps(  
    w_window  
)  
=> t / nil
```

Description

Returns the maximum number of total steps for the progress bar from the window specified.

Argument

<i>w_window</i>	The window from which you want to get the maximum number of total progress bar steps.

Value Returned

<i>t</i>	The progress bar total steps successfully returned.
<i>nil</i>	There was an error.

Example

```
hiGetProgressBarTotalSteps( window ( 1 ) )
```

Related Topics

[Progress Dialog Box Functions](#)

[hiShowProgressBar](#)

[hiSetProgressBar](#)

[hiSetProgressLabel](#)

[hiGetProgressBarTotalSteps](#)

hiGetProgressTotalSteps

```
hiGetProgressTotalSteps(  
    g_dboxID  
)  
=> x_totalSteps
```

Description

Gets the total number of steps of the progress bar in a progress dialog box.

Argument

<i>g_dboxID</i>	The ID of the progress dialog box.

Value Returned

<i>x_totalSteps</i>	The total number of steps on the progress bar.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

[hiSetProgressTotalSteps](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiHideProgressBar

```
hiHideProgressBar(  
    w_windowID  
)  
=> t / nil
```

Description

Removes the progress bar (and its label) from the window you specify.

Argument

<code>w_windowID</code>	The window from which you want to remove the progress bar.

Value Returned

<code>t</code>	The progress bar was removed.
<code>nil</code>	There was an error.

Example

```
hiHideProgressBar( window ( 3 ) )
```

Related Topics

[Progress Dialog Box Functions](#)

[hiShowProgressBar](#)

[hiSetProgressBar](#)

[hiSetProgressLabel](#)

[hiGetProgressBarTotalSteps](#)

[hiDisplayColorDialog](#)

hIsProgressBoxCancelled

```
hIsProgressBoxCancelled(  
    g_dboxID  
)  
=> t / nil
```

Description

Checks whether a progress box is canceled. A progress box is considered canceled if it is canceled with the *Cancel* button, canceled with the `hiCancelProgressBox` function, closed from the window manager, or automatically closed when the progress bar reaches 100% (if it was created with the `?autoClose` argument set to `t`).

Argument

<code>g_dboxID</code>	The ID of the progress dialog box.

Value Returned

<code>t</code>	The progress box has been canceled.
<code>nil</code>	The progress box has not been canceled.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hIsProgressBoxCancelled](#)

[hiGetProgress](#)

[hiGetProgressTotalSteps](#)

[hiSetProgress](#)

[hiSetProgressText](#)

[hiSetProgressTotalSteps](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgress

```
hiSetProgress(  
    g_dboxID  
    x_step  
)  
=> t / nil
```

Description

Sets the progress bar in a progress dialog box to the step you specify. Note, however, that the text above the progress bar remains the same; use `hiSetProgressText` to update the text or `hiSetProgressAndText` to update both the step and the text.

Arguments

<code>g_dboxID</code>	The ID of the progress dialog box.
<code>x_step</code>	The step to which you want to set the progress bar. If the number you specify is greater than the total number of steps in the progress bar, it is set to the highest step.

Value Returned

<code>t</code>	The progress bar was set to the step you specified.
<code>nil</code>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgressText](#)

[hiSetProgressTotalSteps](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgressText

```
hiSetProgressText (  
    g_dboxID  
    t_text  
)  
=> t / nil
```

Description

Sets the text that is displayed above the progress bar in a progress dialog box.

Arguments

<i>g_dboxID</i>	The ID of the progress dialog box.
<i>t_text</i>	The text that is to be displayed above the progress bar.

Value Returned

<i>t</i>	The text was set.
<i>nil</i>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressTotalSteps](#)

hiSetProgressTotalSteps

```
hiSetProgressTotalSteps(  
    g_dboxID  
    x_totalSteps  
    x_step  
)  
=> t / nil
```

Description

Sets the total number of steps on the progress bar in a progress dialog box. This function also sets the current step.

Arguments

<i>g_dboxID</i>	The ID of the progress dialog box.
<i>x_totalSteps</i>	The total number of steps that you want on the progress bar.
<i>x_step</i>	The current step to which you want set the progress bar. If the number you specify is greater than the total number of steps in the progress bar, it is set to the highest step.

Values Returned

<i>t</i>	The values you specified were set.
<i>nil</i>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hiIsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgressAndText

```
hiSetProgressAndText (  
    g_dboxID  
    x_step  
    t_text  
)  
=> t / nil
```

Description

In a progress dialog box, sets the current step of the progress bar as well as the text that is displayed above the progress bar.

Arguments

<i>g_dboxID</i>	The ID of the progress dialog box.
<i>x_step</i>	The current step to which you want to set the progress bar. If the number you specify is greater than the total number of steps in the progress bar, it is set to the highest step.
<i>t_text</i>	The text that is to be displayed above the progress bar.

Values Returned

<i>t</i>	The values you specified were set.
<i>nil</i>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hiIsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

hiSetProgressButtonText

```
hiSetProgressButtonText(  
    g_dboxID  
    t_buttonText  
)  
=> t / nil
```

Description

Changes the text of the button on a progress dialog box. Progress dialog boxes, created with `hiDisplayProgressBox`, have one button, whose default text is *Cancel*.

Arguments

<code>g_dboxID</code>	The ID of the progress dialog box.
<code>t_buttonText</code>	The new text to be displayed on the button.

Values Returned

<code>t</code>	The button text was set.
<code>nil</code>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgressBannerText

```
hiSetProgressBannerText (  
    g_dboxID  
    t_banner  
)  
=> t / nil
```

Description

Sets the text that is displayed on the window banner of a progress dialog box.

Arguments

<i>g_dboxID</i>	The ID of the progress dialog box.
<i>t_banner</i>	The text to be displayed on the banner of the progress dialog box.

Values Returned

<i>t</i>	The banner text was set.
<i>nil</i>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiResetProgressBox

```
hiResetProgressBox(  
    g_pboxID  
    x_totalSteps  
    t_text  
    g_autoClose  
)  
=> t / nil
```

Description

Resets a progress dialog box to the new values you specify. The progress bar is also reset to 0%.

Arguments

<i>g_dboxID</i>	The ID of the progress dialog box.
<i>x_totalSteps</i>	The total number of steps on the progress bar.
<i>t_text</i>	The text to be displayed above the progress bar.
<i>g_autoClose</i>	<i>t</i> or <i>nil</i> , specifying whether the progress dialog box should be closed when the progress bar reaches 100%.

Value Returned

<i>t</i>	The progress dialog box was reset.
<i>nil</i>	There was an error.

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

hiShowProgressBar

```
hiShowProgressBar(  
    w_windowID  
    [ x_totalSteps ]  
    [ x_delay ]  
    [ t_label ]  
  
    )  
=> t / nil
```

Desecription

Displays a progress bar in the status bar of a window, typically in the bottom-right corner. The progress bar displays the progress of a command.

This function sets the total number of steps in the progress bar as well as the delay before the progress bar appears on the window. To set the current progress, use `hiSetProgressBar`.

Arguments

<i>w_windowID</i>	The window to which you want to add the progress bar.
<i>x_totalSteps</i>	The total number of steps on the progress bar.
<i>x_delay</i>	The time delay, in milliseconds, before the progress box appears.
<i>t_label</i>	The progress bar label, that is, the text that is displayed on the left of the progress bar.

You can also use `hiSetProgressLabel` to set the label.

Values Returned

<i>t</i>	The progress bar was displayed.
<i>nil</i>	There was an error.

Example

```
hiShowProgressBar(window(3) 200 0)  
hiSetProgressBar(window(3) 50)
```


Cadence User Interface SKILL Reference

Dialog Boxes

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

[hiSetProgressLabel](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgressBar

```
hiSetProgressBar(  
    w_windowID  
    x_step  
)  
=> t / nil
```

Description

Sets the progress (the current step) of the progress bar in the window you specify.

Arguments

<i>w_windowID</i>	The window in which you want to set the progress bar.
<i>x_step</i>	The current step to which you want to set the progress bar. The percentage displayed is relative to the total number of steps, which is set by the <code>hiShowProgressBar</code> function.

Values Returned

<i>t</i>	The progress bar was set.
<i>nil</i>	There was an error.

Example

```
hiShowProgressBar( window ( 3 ) 200 0 )  
hiSetProgressBar( window ( 3 ) 50 )
```

Related Topics

[Progress Dialog Box Functions](#)

[hiDisplayProgressBox](#)

[hilsProgressBoxCancelled](#)

[hiSetProgress](#)

[hiSetProgressText](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgressLabel

Cadence User Interface SKILL Reference

Dialog Boxes

hiSetProgressLabel

```
hiSetProgressLabel(  
    w_window  
    t_label  
)  
=> t / nil
```

Description

Sets the label of the progress bar in the specified window. An empty string removes the label.

You can use `hiDisplayColorDialog` to remove both the progress bar and its label.

Arguments

<code>w_windowID</code>	The window that contains the progress bar.
<code>t_label</code>	The progress bar label, that is, the text that is displayed on the left of the progress bar. To remove the label, specify an empty string (" ").

Values Returned

<code>t</code>	The label was set.
<code>nil</code>	There was an error.

Examples

```
hiSetProgressLabel( window ( 3 ) "Loading design" )
```

Related Topics

[Progress Dialog Box Functions](#)

[hiShowProgressBar](#)

[hiSetProgressBar](#)

[hiDisplayColorDialog](#)

[hiGetProgressBarTotalSteps](#)

Color Dialog Function

This topic provides Cadence® SKILL function which is associated with color dialog.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- hiDisplayColorDialog

hiDisplayColorDialog

```
hiDisplayColorDialog(  
    [ ?initialColor l_initialColor ]  
    [ ?alpha g_alpha ]  
)  
=> l_RGBandAlpha / nil
```

Description

Displays the Select color dialog, which lets you select a color. This function returns the RGB (Red, Green, and Blue) value of the color you select as well as the alpha channel value, if the `?alpha` argument was `t`. The Select color dialog is a modal dialog.

In the dialog, you can pick a color from the *Basic colors* or *Custom colors* table, or by moving the cross in the color spectrum to the color you want, and/or modifying the *Hue*, *Sat* (Saturation), *Val* (Value), *Red*, *Green*, and *Blue* fields. You can also specify transparency with the *Alpha channel* field. To save a custom color, click *Add to Custom Colors*.

Hue represents the color and has a range of 0 to 359; saturation represents the intensity of the color and has a range of 0 to 255; and value represents the brightness of the color and has a range of 0 to 255. The alpha channel of a color specifies the transparency effect and has a range of 0 to 255, where 0 represents a fully transparent color while 255 represents a fully opaque color.

The *Alpha channel* field is only displayed in the Select color dialog if you call this function with the `?alpha` argument set to `t`.

Arguments

`?initialColor` *l_initialColor*

Cadence User Interface SKILL Reference

Dialog Boxes

Specifies the default color that is already selected when the color dialog appears. Specify *l_initialColor* as a list of three or four elements. The first three elements represent the RGB value of the color. For example:

```
list(255 85 0)
```

The fourth optional element specifies the initial value for the *Alpha channel* field (which is displayed only if the *g_alpha* argument is *t*). This field sets the transparency effect and can have a value between 0 and 255, where 0 represents a fully transparent color while 255 represents a fully opaque color. For example:

```
list(255 85 0 110)
```

Default value: *nil*

?alpha *g_alpha*

Specifies whether to display the *Alpha channel* option on the color dialog. This option is used to control transparency.

Default value: *nil*

Values Returned

l_RGBandAlpha

A list containing the values selected in the color dialog. The first three elements of the list represent the RGB values of the color that was selected and the fourth element represents the *Alpha channel* value selected. If the *g_alpha* argument was *nil* when the function was called, the list contains only three elements.

nil

Returns *nil* if you cancel the color dialog.

Examples

Displays the Select dialog box with RGB values 0 85 255

```
hiDisplayColorDialog( )  
=> (0 85 255)
```

```
hiDisplayColorDialog( ?initialColor list ( 0 85 255 ) ?alpha t )  
=> ((255 85 0 230))
```

Old Dialog Box Functions

The following functions have been replaced with [hiDisplayAppDBox](#):

- [hiDisplayBlockingDBox](#)
- [hiDisplayModalDBox](#)
- [hiDisplayModelessDBox](#)
- [hiDisplayNonBlockingDBox](#)

Cadence User Interface SKILL Reference

Dialog Boxes

hiDisplayBlockingDBox

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

Description

Use [hiDisplayAppDBox](#), which is the recommended dialog box interface function.

hiDisplayModalDBox

```
hiDisplayModalDBox(  
    s_dBoxHandle  
    t_dBoxBannerText  
    t_dBoxMsg  
    g_dBoxHelp  
    t_confirmAction  
    [ l_location ]  
    [ x_dboxType ]  
)  
=> t / nil
```

Description

Use [hiDisplayAppDBox](#), which is the recommended dialog box interface function.

hiDisplayModelessDBox

```
hiDisplayModelessDBox(  
    s_dBoxHandle  
    t_dBoxBannerText  
    t_dBoxMsg  
    g_dBoxHelp  
    t_confirmAction  
    [ l_location ]  
    [ x_dboxType ]  
)  
=> t / nil
```

Description

Use [hiDisplayAppDBox](#), which is the recommended dialog box interface function. Additionally, set *?dialogStyle* to 'modeless.

Cadence User Interface SKILL Reference

Dialog Boxes

hiDisplayNonBlockingDBox

hiDisplayNonBlockingDBox()

Description

Use hiDisplayAppDBox, which is the recommended dialog box interface function.

List Box Functions

This topic provides a list of Cadence® SKILL functions associated with list boxes.

List boxes display a list of user-defined strings in a scrollable window. The window has a vertical and horizontal scroll bar and the size of the window is constant. You can select a string (or strings) using the left mouse button.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiDisplayListBox](#)
- [hiGetListBoxFieldFit](#)
- [hiListBoxCancel](#)
- [hiListBoxDone](#)
- [hiShowListBox](#)

hiDisplayListBox

```
hiDisplayListBox(  
    s_listBoxHandle  
    t_listBoxTitle  
    l_listItems  
    t_help  
)  
=> t / nil
```

Description

Creates and displays a list box. The list box displays a list of strings. The user can select one of these strings by clicking over the item with the left mouse button. The list box remains on the screen until the user selects *OK* or *Cancel*. The list box is destroyed when it is removed from the screen

Arguments

<i>s_listBoxHandle</i>	A unique global SKILL handle to the created list box. The selected item can be retrieved from the <i>listBoxHandle</i> . For example, if <i>listBoxHandle</i> was <code>myListBox</code> , the selected item would be: <i>myListBox -> value</i> This value is a list of two items: the integer representing the position of the selected item in the item list and the selected item string.
<i>t_listBoxTitle</i>	Title of the list box displayed in the <i>WindowManager</i> banner.
<i>l_listItems</i>	List of strings that is displayed in the scrolling window.
<i>t_help</i>	String containing information about this list box.

Values Returned

<i>t</i>	The <i>OK</i> button was selected.
<i>nil</i>	The <i>Cancel</i> button was selected.

Related Topics

List Box Functions

Cadence User Interface SKILL Reference

List Box Functions

hiShowListBox

Cadence User Interface SKILL Reference

List Box Functions

hiGetListBoxFieldFit

```
hiGetListBoxFieldFit(  
    x_numRows  
    x_numColumns  
    [ g_ignored ]  
)  
=> l_widthHeight
```

Description

Returns the width and height necessary for displaying a list box with the specified number of rows and columns and the specified font.

Arguments

<i>x_numRows</i>	Number of rows to display in the list box.
<i>x_numColumns</i>	Number of columns (characters) to display in the list box.
<i>g_ignored</i>	Optional. Any value that exists for backwards compatibility.

Value Returned

<i>l_widthHeight</i>	List of the width and height of the list box. The width for the variable width fonts is based on the size of a numeric character.

Examples

Creates a form with a list box field large enough to show four rows of values and as many columns as the largest value in the value list.

```
let( (lbValues (lbColumns 0) lbSize (lbPrompt "My List") lbPromptSize  
    (lbFont hiGetFont("text")))  
    lbValues = list( "this is my first value string"  
                    "this is my second value string"  
                    "this is my third value string"  
                    "this is my fourth value string"  
                    "this is my fifth value string"  
                    "this is my sixth value string"  
                    "this is my seventh value string"  
                    "this is my eighth value string" )  
    foreach( x lbValues  
        let( (y)  
            y = strlen( x )  
            when( y > lbColumns lbColumns = y )
```


Cadence User Interface SKILL Reference

List Box Functions

```
)
)
lbPromptSize = hiGetTextWidth( lbFont lbPrompt )
lbSize = hiGetListBoxFieldFit( 4 lbColumns lbFont )
MyListBoxField = hiCreateListBoxField(
    ?name 'MyListBoxField
    ?prompt lbPrompt
    ?choices lbValues )
MyListBoxForm = hiCreateAppForm(
    ?name 'MyListBoxForm
    ?fields list( list( MyListBoxField 5:5
                        list( car(lbSize) + lbPromptSize
                            cadr(lbSize))
                        lbPromptSize ) )
    )
)
```

Related Topics

List Box Functions

hiDisplayListBox

hiGetNumVisibleItems

hiListBoxCancel

```
hiListBoxCancel(  
    o_listBox  
)  
=> t / nil
```

Description

Performs the same action as the *Cancel* button on the listbox.

If the listbox is mapped, it gets unmapped. The listbox callback is called with a first argument of `nil` to indicate that *Cancel* was pressed.

Argument

<code>o_listBox</code>	SKILL handle to the listbox. This is the value of the <code>listBoxHandle</code> symbol passed into <code>hiShowListBox</code> .

Values Returned

<code>t</code>	The list box is canceled.
<code>nil</code>	The list box is not canceled.

Related Topics

[List Box Functions](#)

[hiShowListBox](#)

Cadence User Interface SKILL Reference

List Box Functions

hiListBoxDone

```
hiListBoxDone(  
    o_listBox  
)  
=> t / nil
```

Description

Performs the same action as the *OK* button on the listbox.

Argument

<i>o_listBox</i>	SKILL handle to the listbox. This is the value of the <i>listBoxHandle</i> symbol passed into <i>hiShowListBox</i> .

Values Returned

<i>t</i>	The list box is unmapped.
<i>nil</i>	The list box is not unmapped.

Related Topics

List Box Functions

hiShowListBox

Cadence User Interface SKILL Reference

List Box Functions

hiShowListBox

```
hiShowListBox(  
    ?name s_listBoxHandle  
    ?choices lt_listItems  
    ?callback s_callback  
    [?title t_listBoxTitle]  
    [?multipleSelect g_multiSelect]  
    [?value lt_selectedItem]  
    [?appData g_appData]  
    [?applyButton g_applyButton]  
)  
=> t / nil
```

Description

Creates and displays a list box. The list box displays a list of strings. The list box can be created so that one item is selectable or multiple items are selectable. While the list box is displayed on the screen, the user may change the items in the list box, or change the item(s) selected. The list box remains on the screen until the user selects *OK* or *Cancel*. The list box is destroyed when it is removed from the screen.

Arguments

?name *s_listBoxHandle*

A unique global SKILL handle to the created list box. The selected item and other useful information can be retrieved from the *listBoxHandle*. For example, if *listBoxHandle* was 'myListBox, the following could be accessed and/or set:

myListBox->??	;all list box
properties (read-only)	
myListBox->hiListBoxSym	;s_listBoxHandle
(read-only)	
myListBox->title	;list box title
(read-only)	
myListBox->value	;selected item(s)
myListBox->choices	;item(s) displayed
in list box	
myListBox->appData	;application data

Cadence User Interface SKILL Reference

List Box Functions

?choices *lt_listItems*

List of strings that is displayed in the scrolling window. This list can be referenced **or** reset through:

listBoxHandle->choices

When the list box choices have been changed, any item(s) previously selected gets unselected (for example, *listBoxHandle->value* is set to *nil*).

?callback *s_callback*

The user-defined SKILL procedure that is run when either the *OK* or *Cancel* button of the list box is pressed. This procedure must be in the following format:

myListBoxCB(g_OKpressed o_listBox)

where *g_OKpressed* is *t* if the user pressed the *OK* button and is *nil* if the user pressed *Cancel*. The argument *o_listBox* is the SKILL structure of the list box that was dismissed. Various elements can be accessed through this list box structure, as described above.

?title *t_listBoxTitle*

Optional title of the list box displayed in the *WindowManager* banner. This title can also be referenced through:

listBoxHandle->title

?multipleSelect *g_multiSelect*

If set to *t*, multiple items in the list box can be selected; if set to *nil*, at most one item can be selected at a time. The default value is *nil*.

?value *lt_selectedItem*

List of string item(s) that needs to appear as already selected when the list box is displayed. These strings must also be specified in *lt_listItems*. If not specified, nothing is pre-selected. If this list contains more than one string, the argument *g_multiSelect* must be set to *t*. The currently selected item(s) can be referenced **or** set through:

listBoxHandle->value

?appData *g_appData*

Cadence User Interface SKILL Reference

List Box Functions

Used to store any application-specific data associated with the list box. It can be referenced **or** set through:

```
listBoxHandle->appData
```

```
?applyButton g_applyButton
```

If set to `t`, places an *Apply* button in the list box. The default is `nil` (no *Apply* button).

There is also an `applying` property which, if queried during the callback of the list box (via `myListBox->applying`), returns `t` if the *Apply* button was the reason for the callback, or `nil` if the callback was called because the user clicked *OK*.

Values Returned

<code>t</code>	The <i>OK</i> button was selected.
<code>nil</code>	The <i>Cancel</i> button was selected.

The list box symbol is reset to `nil` after the callback has been executed, and the list box is dismissed from the screen.

Examples

Creates and display a list box showing various flowers. The user should be able to select all of his/her favorite flowers.

```
hiShowListBox(?name 'myLB
?title "Pick your favorite flowers"
?choices list("daffodil" "lily" "freesia" "rose" "carnation")
?value list("rose" "freesia")
?callback 'myListBoxCB
?multipleSelect t )

; This procedure is called when the user selects OK or
; Cancel on the list box
procedure(myListBoxCB(okPressed listbox "go")
    ; if user pressed OK, save the selected flowers
    if( okPressed saveFlowers(listbox->value) )
)

; Change the list of flowers being displayed
myLB->choices = list("petunia" "tiger lily"
"bird of paradise" "chrysanthemum")

; Select a flower
myLB->value = list("bird of paradise")
```

Creates a simple list box.

Cadence User Interface SKILL Reference

List Box Functions

```
;;; trShowListBox generates a new list box displaying aList
procedure( trShowListBox( aList )
  let( ( listBoxSymbol )
    listBoxSymbol = gensym( 'trExampleListBox )
    hiShowListBox(
      ?name listBoxSymbol
      ?choices aList
      ?callback 'trExampleListBoxCB
      ?title sprintf( nil "List Box: %L" listBoxSymbol )
      ?multipleSelect t
      ?applyButton t
    )
  ) ; let
) ; procedure

procedure( trExampleListBoxCB( ok theListBox )
  "The callback routine"
  println( list( ok theListBox->value ) )
) ; procedure
```

To test your application, create two different list boxes and verify that the two list boxes behave independently:

```
trShowListBox( '( "apple" "orange" "kiwi" ) )
trShowListBox( '( "wood" "water" "metal" ) )
```

Creates a list box that displays all the functions matching the specified pattern.

```
procedure( trFunctionListBox( pattern )
  "uses trListFunctions to compose a Show List Box"
  hiShowListBox(
    ?name gensym( 'trFunctionListBoxHandle )
    ?title sprintf( nil "Matching: %s" pattern )
    ?choices trListFunctions( pattern )
    ?callback 'trFunctionListBoxCB
    ?applyButton t
    ?multipleSelect t
  ) ; hiShowListBox
) ; procedure

procedure( trListFunctions( pattern )
  "returns a list of functions for the matching pattern"
  foreach( mapcar fun listFunctions( pattern )
    get_pname( fun )
  )
) ; procedure

procedure( trFunctionListBoxCB( ok theListBox )
  let( ( theArglist )
    when( ok
      foreach( funName theListBox->value
        theArglist = arglist( makeSymbol( funName ) )
        if( theArglist
          then
            printf( "\n%s%L" funName theArglist )
          else
            printf( "\n%s()" funName )
          ) ; if
        ) ; foreach
      ) ; when
    ) ; let
  ) ; procedure
```

Cadence User Interface SKILL Reference

List Box Functions

To find all the functions that start with "hiGet", type the following command in the CIW:

```
trFunctionListBox( "^hiGet" )
```

Additional Information

You can select mechanism for multiple selection in list boxes using `hiShowListBox:`

To select a continuous range of items:

1. Press the left mouse button.
2. Drag the cursor over the items you wish to select.
3. Release the mouse button.

Each time the left mouse button is pressed, a new selection process begins, and the previously selected items are unselected.

To modify a selection:

1. Press and hold the *Shift* key.
2. Move to the new endpoint of selected items.
3. Press the left mouse button.

The items between the initial start point and the new endpoint are selected; any previous selections are not unselected.

To add or delete items from the selected set:

1. Press and hold the *Control* key.
2. Move to the item you wish to add or delete.
3. Press the left mouse button (inverts the selection state of this item).
4. Drag the mouse button, if you wish to add or delete a range of items.

The item(s) selected or unselected gets added to or deleted from the selected set. Any previous selections remain unaffected.

Related Topics

List Box Functions

Cadence User Interface SKILL Reference

List Box Functions

hiDisplayListBox

hiListBoxCancel

hiListBoxDone

Cadence User Interface SKILL Reference

List Box Functions

Icon Functions

This topic provides a list of Cadence® SKILL functions associated with icons creation that are used in user interface (hi) functions.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [colorIndex](#)
- [hiAddIconOverrides](#)
- [hiCreateColorArray](#)
- [hiGetIconOverrides](#)
- [hiLoadIconFile](#)
- [hiLoadImageFile](#)
- [hiLoadIconData](#)
- [hiMatchColor](#)
- [hiMatchColorByName](#)
- [hiMatchColorByName](#)
- [hiRemoveIconOverrides](#)
- [hiSaveIconData](#)
- [hiStringToIcon](#)
- [hiSetDisabledIcon](#)
- [nameToColor](#)

colorIndex

```
colorIndex(  
    l_rgb  
)  
=> x_index
```

Description

Returns the pixel value for the colormap index of the specified RGB value. This function is an alias for `hiMatchColor`.

Arguments

<i>l_rgb</i>	Specifies the RGB value.

Value Returned

<i>x_index</i>	The pixel value for the colormap index.

Related Topics

[Icon Functions](#)

[hiStringToIcon](#)

hiAddIconOverrides

```
hiAddIconOverrides
    l_list
)
=> t / nil
```

Description

Adds names to the list of icons that are loaded using CSF search results directly, bypassing the hierarchy files. If the CSF search fails, the files in the hierarchy are loaded.

Icons not loaded using full file paths are loaded from the files located in the hierarchy at /share/cdssetup/icons/{16x16,24x24,32x32,...}. Any icons not found here are searched using CSF.

Arguments

<i>l_list</i>	List of icon names.

Values Returned

t	The listed icon names are added.
nil	No icon names were added because the input list contains an element that is not a string.

Example

Adds "point.png" and "rotate.png" to the list of HI icons that gets loaded later by calls to hiCreateAction, hiLoadImageFile, and so on by the CSF search to bypass the files in share/cdssetup/icon.

```
hiAddIconOverrideShareNames( ' ("point.png" "rotate.png") )
t
```

Related Topics

[Icon Functions](#)

[hiRemoveIconOverrides](#)

Cadence User Interface SKILL Reference

Icon Functions

hiGetIconOverrides

hiCreateColorArray

```
hiCreateColorArray(  
    )  
    => a_iconColorArray
```

Description

Returns a 27-element color array that can be used in a call to `hiStringToIcon`. The first 26 items correspond to the letters in the alphabet; the 27th item corresponds to a period (.).

Example: `colorArray[0]` is represented by `a` or `A`; `colorArray[12]` is represented by `m` or `M`; `colorArray[26]` is represented by a period (.).

To set these array elements, use `hiMatchColorByName` or `hiMatchColor`.

Arguments

None

Value Returned

a_iconColorArray A 27-element SKILL array.

Related Topics

[Icon Functions](#)

[hiStringToIcon](#)

[hiMatchColorByName](#)

[hiStringToIcon](#)

[hiSetDisabledIcon](#)

hiGetIconOverrides

```
hiGetIconOverrides(  
    )  
=> l_list
```

Description

Returns the list of icon names that are loaded by the CSF search results directly, bypassing the hierarchy files. If the CSF search fails, the files in the hierarchy are loaded.

Icons not loaded using full file paths are loaded from the files located in the hierarchy at /share/cdssetup/icons/{16x16,24x24,32x32,...}. Any icons not found here are searched using CSF.

Arguments

None

Value Returned

<code>l_list</code>	List of icon names from previous calls to <code>hiAddIconOverrides</code> and <code>hiRemoveIconOverrides</code> .

Examples

Returns a list of HI icon names for loading icons by using the CSF search and bypassing the share/cdssetup/icons files.

```
hiAddIconOverrides( '("point.png" "rotate.png") )  
t  
hiRemoveIconOverrides( '("point.png" ) )  
t  
hiGetIconOverrides()  
("rotate.png")
```

Related Topics

[Icon Functions](#)

[hiAddIconOverrides](#)

Cadence User Interface SKILL Reference

Icon Functions

hiRemoveIconOverrides

hiLoadIconFile

```
hiLoadIconFile(  
    t_fileName  
    x_width  
    x_height  
)  
=> l_hiIcon / nil
```

Description

Loads an icon file of any format, such as PNG, and returns the icon in a format that can be used with `hi` functions that accept icons. If `t_fileName` is a relative path, the Cadence Search File mechanism (CSF) is used to search for the file.

Multiple resolutions of the icon are loaded, if found. The appropriate resolution is used when the icon is used.

This function provides you the ability to add your own icons for custom toolbars, menus, and forms. You can place these icons in a subdirectory of your home directory. Similarly, central groups can provide icons for your entire site in a different location. Any icons you provide must be placed in an `icons/16x16`, `icons/24x24`, `icons/32x32`, or `icons/48x48` subdirectory of a directory found by CSF. These directories should represent the resolution of the image files in them.

Follow these guidelines when you create icons:

- While any format supported by Qt is supported, PNG is preferred.
- Limit icons to 256 color.
- Use a consistent icon palette.

Cadence User Interface SKILL Reference

Icon Functions

Arguments

<i>t_fileName</i>	The name of the icon file to load. If <i>t_fileName</i> is an absolute path (starting with /), then the file is loaded directly. Only a single resolution is available in this case. If <i>t_fileName</i> is a relative path, then CSF rules are used to search for all resolutions of the file. For each location in your <code>setup.loc</code> file, the following subdirectories are searched. The names of these subdirectories represent the resolution of the image files in them. <code>icons/16x16</code> <code>icons/24x24</code> <code>icons/32x32</code> <code>icons/48x48</code> Whichever resolutions are found are loaded into the icon. When the icon is used, the appropriate resolution is used.
<i>x_width</i>	The desired width of the icon. The icon that is returned is not sized to the width and height specified; these values are added to the return value of this function and applied when the icon is used.
<i>x_height</i>	The desired height of the icon. The icon that is returned is not sized to the width and height specified; these values are added to the return value of this function and applied when the icon is used.

Values Returned

<i>l_hiIcon</i>	The icon handle, which can be used with any user interface function that accepts icons. <i>l_hiIcon</i> is of the following form: <code>list(handle width height)</code>
<i>nil</i>	<i>t_fileName</i> is not found or there was an error in loading it.

Examples

```
?itemIcon hiLoadIconFile( "wire.png" 16 16 )
```

Cadence User Interface SKILL Reference

Icon Functions

Related Topics

[Icon Functions](#)

[hiLoadIconFile](#)

[hiLoadImageFile](#)

[hiLoadIconData](#)

[hiSaveIconData](#)

[hiStringToIcon](#)

[hiMatchColorByName](#)

hiLoadImageFile

```
hiLoadImageFile(  
    t_fileName  
    x_width  
    x_height  
)  
=> l_hiImage / nil
```

Description

Loads an image file of any format, such as PNG, and returns the image in a format that can be used with `hi` functions that accept icons. If `t_fileName` is a relative path, the Cadence Search File mechanism (CSF) is used to search for the file.

Multiple resolutions of the icon are loaded, if found. The appropriate resolution is used when the icon is used.

This function provides you the ability to add your own images (icons) for custom toolbars, menus, and forms. You can place these icons in a subdirectory of your home directory. Similarly, central groups can provide icons for your entire site in a different location. Any icons you provide must be placed in an `icons/16x16`, `icons/24x24`, `icons/32x32`, or `icons/48x48` subdirectory of a directory found by CSF. These directories should represent the resolution of the image files in them.

Follow these guidelines when you create icons:

- While any format supported by Qt is supported, PNG is preferred.
- Limit icons to 256 color.
- Use a consistent icon palette.

To this point, the above use description is identical to that of `hiLoadIconFile`. However if the file being referred to has changed on disk, a new image referring to the updated image is returned, rather than the cached image that is returned by `hiLoadIconFile`.

The `hiLoadIconFile` command returns the most recently cached version of a file, although multiple calls to `hiLoadIconFile` returns the same value, even if the file has changed on disk, if `hiLoadImageFile` is called for this same icon, and it has changed on disk, then the next call to `hiLoadIconFile` returns the newly-cached version, not the original version.

Cadence User Interface SKILL Reference

Icon Functions

The `hiLoadIconFile` command always return the cached version of an icon for performance reasons (the filesystem search for an icon should only occur once during the initial call to `hiLoadIconFile`). Alternatively, `hiLoadImageFile` searches for an updated version of the icon upon each invocation. It should therefore only be used in cases where the image file may change during the lifetime of the program, and it is desirable to always have the most recent version.

The most common use then of `hiLoadImageFile` is to load thumbnail images for cellviews, as these are typically updated when a view is saved, and therefore changes during the lifetime of the program. Displaying an incorrect thumbnail image may be misleading, hence it is desirable to call `hiLoadImageFile` to check for updates, rather than always using the original image returned by `hiLoadIconFile`.

Unless there is a good reason to perform such filesystem checks, you should use `hiLoadIconFile` in order to avoid repeated checks.

Arguments

t_filename

The name of the image file to load.

If *t_filename* is an absolute path (starting with /), then the file is loaded directly. Only a single resolution is available in this case.

If *t_filename* is a relative path, then CSF rules are used to search for all resolutions of the file. For each location in your `setup.loc` file, the following subdirectories are searched. The names of these subdirectories represent the resolution of the image files in them.

`icons/16x16`

`icons/24x24`

`icons/32x32`

`icons/48x48`

Whichever resolutions are found are loaded into the image. When the image is used, the appropriate resolution is used.

x_width

The desired width of the image.

The image that is returned is not sized to the width and height specified; these values are added to the return value of this function and applied when the icon is used.

Cadence User Interface SKILL Reference

Icon Functions

x_height

The desired height of the image.

The image that is returned is not sized to the width and height specified; these values are added to the return value of this function and applied when the image is used.

Values Returned

l_hiImage

The image handle, which can be used with any user interface function that accepts icons. *l_hiImage* is of the following form:

list(handle width height)

nil

t_fileName is not found or there was an error in loading it.

Examples

Loads the thumbnail for myLib/myCell/myView into an icon:

```
myView = ddGetObj( myLib myCell myView)
hiLoadImageFile( sprintf(nil "%s/%s" ddGetObjReadPath(myView)
hiGetThumbnailFilename())) 64 64)
```

Related Topics

[Icon Functions](#)

[hiLoadIconFile](#)

[hiLoadImageFile](#)

[hiLoadIconData](#)

[hiSavelconData](#)

[hiStringToIcon](#)

[hiMatchColorByName](#)

hiLoadIconData

```
hiLoadIconData(  
    t_fileName  
    [ t_formatName ]  
)  
=> l_hiIcon / nil
```

Description

Loads a pixmap and returns it in a format that can be used with any `hi` function that accepts icons, such as `hiCreateMenuItem` or `hiCreateButton`. This function can load files that are in any of the image formats supported by Qt; currently these are: BMP, JPG, PNG, PPM, XBM, and XPM. PNG is the recommended format.

Arguments

<i>t_fileName</i>	The pixmap file, in one of the following formats: BMP, JPG, PNG, PPM, XBM, and XPM.
<i>t_formatName</i>	The image format name. The default value of this argument is "PNG".

Values Returned

<i>l_hiIcon</i>	The icon handle, which can be used with any user interface function that accepts icons. <i>l_hiIcon</i> is of the following form: <code>list(handle width height)</code>
<i>nil</i>	There was an error in loading <i>t_fileName</i> .

Examples

```
zoomPixmap = hiLoadIconData( "zoom.png" "png" )  
hiCreateMenuItem ( ?name 'zoom ?itemIcon zoomPixmap )
```

Related Topics

Icon Functions

hiLoadIconFile

Cadence User Interface SKILL Reference

Icon Functions

hiLoadImageFile

hiLoadIconData

hiSaveIconData

hiStringToIcon

hiMatchColorByName

hiMatchColor

```
hiMatchColor(  
    l_RGBcolor  
)  
=> x_colormapIndex
```

Description

Returns a layer color index corresponding to the specified RGB color. It creates a new color index if the color specified does not have any corresponding index.

Argument

<i>l_RGBcolor</i>	List of red, green, and blue integer coordinates in the range 0–1000 representing a color, such that it can be returned by the <code>nameToColor</code> function.

Value Returned

<i>x_colormapIndex</i>	A color index representing the layer color that matches the color specification. Returns the new color index, if the color specified does not have any corresponding index.

Examples

Returns a color index corresponding to a list of specified integer coordinates.

```
hiMatchColor('(1000 0 0))  
=>6 20
```

Returns a color index corresponding to the color, `red`.

```
hiMatchColor(nameToColor("red"))  
=>8 20
```

Returns a color index corresponding to the RGB string, `#FF0000`.

```
hiMatchColor(nameToColor("#FF0000"))  
=>10 20
```

Cadence User Interface SKILL Reference

Icon Functions

Related Topics

[Icon Functions](#)

[nameToColor](#)

[hiStringToIcon](#)

hiMatchColorByName

```
hiMatchColorByName(  
    t_colorName  
)  
=> x_colormapIndex
```

Description

Returns a layer color index corresponding to the specified color name.

Argument

<i>t_colorname</i>	Specify the color name, such as <code>red</code> or <code>green</code> or an RGB string, such as <code>#ff0000</code> (red) or <code>#00ff00</code> (green).

Value Returned

<i>x_colormapIndex</i>	Returns a layer color index if the color name is valid. Otherwise, a warning a message is displayed and the index of the editor background color is returned.

Examples

Returns the layer color index for the color, `red`.

```
hiMatchColorByName("red")  
=> 6 20
```

Returns the layer color index for the RGB string, `#ff0000`.

```
hiMatchColorByName("#ff0000")  
=> 8 20
```

Returns the warning message because the color name is not valid.

```
hiMatchColorByName("bad name")  
=> 10 *WARNING* nameToColor: could not parse color string "bad name"  
=> 11 *WARNING* hiMatchColorByName: defaulting to editor background color for color  
name "bad name"
```

Cadence User Interface SKILL Reference

Icon Functions

Related Topics

[Icon Functions](#)

[hiStringToIcon](#)

[hiSetDisabledIcon](#)

hiRemoveIconOverrides

```
hiRemoveIconOverrides  
    l_list  
)  
=> t / nil
```

Description

Removes names from the list of icons that are loaded using the CSF search results directly, bypassing the hierarchy files. If the CSF search fails, the files in the hierarchy are loaded.

Note: Icons not loaded using full file paths are loaded from the files located in the hierarchy at `/share/cdssetup/icons/{16x16,24x24,32x32,...}`. Any icons not found here are searched using CSF.

Arguments

<code>l_list</code>	List of icon names to be removed.

Value Returned

<code>t</code>	Specified icon names were removed.
<code>nil</code>	No icon names were removed because the input list contains an element that is not a string.

Examples

Removes "point.png" and "rotate.png" from the list of HI icons previously added by `hiAddIconOverrides`. They use the `share/cdssetup/icons` files instead of the CSF search results.

```
hiRemoveIconOverrides ( ("point.png" "rotate.png") )  
=>t
```

Related Topics

[Icon Functions](#)

[hiAddIconOverrides](#)

Cadence User Interface SKILL Reference

Icon Functions

hiGetIconOverrides

hiSaveIconData

```
hiSaveIconData (
    l_hiIcon
    t_fileName
    [ t_formatName ]
)
=> t / nil
```

Description

Saves a loaded pixmap to a file in the specified format. This function accepts an `hi` pixmap description and converts it to any of the image formats supported by Qt; currently, these are: BMP, JPG, PNG, PPM, XBM, and XPM. However, PNG is the recommended format.

Arguments

<i>l_hiIcon</i>	The pixmap handle, which is the return value of <code>hiStringToIcon</code> or <code>hiLoadIconData</code> .
<i>t_fileName</i>	The name of the file to which you want to save the pixmap.
<i>t_formatName</i>	The format in which you want to save the pixmap.

Values Returned

<i>t</i>	The pixmap was saved in the format you specified.
<i>nil</i>	There was an error.

Examples

Saves a loaded pixmap by the name `legacy.png` in the `png` format.

```
hiSaveIconData ( legacyPixmap "legacy.png" "png" )
```

Related Topics

[Icon Functions](#)

[hiStringToIcon](#)

[hiLoadIconFile](#)

Cadence User Interface SKILL Reference

Icon Functions

hiLoadImageFile

hiLoadIconData

hiSaveIconData

hiStringToIcon

hiMatchColorByName

hiStringToIcon

```
hiStringToIcon(  
    a_colorArray  
    t_iconString  
    x_width  
    x_height  
)  
=> l_hiIcon / nil
```

Description

Returns an icon description that can be used in hi functions that accept icons, such as `hiCreatePulldownMenu`, `hiCreateMenuItem`, `hiCreateSliderMenuItem`, `hiCreateButton`, `hiCreateCyclicField`, `hiCreateLabel`, and `hiAddCyclicChoice`.

Arguments

<i>a_colorArray</i>	Array created by <code>hiCreateColorArray</code> . Each element of this array describes a pixel color to use within the icon.
<i>t_iconString</i>	ASCII description of the icon. The maximum string length for <i>t_iconString</i> is 8K (8192) characters.
<i>x_width</i>	Width of the icon.
<i>x_height</i>	Height of the icon.

Values Returned

<i>l_hiIcon</i>	The icon handle, which can be used with any hi function that accepts icons. <i>l_hiIcon</i> is of the following form: <code>list(handle width height)</code>
<i>nil</i>	An error occurred.

Examples

For convenience, the character '0' represents white; the character '1' represents black.
Example: to describe an 8x8 pixel icon of a red square surrounded by blue, then thistle on a black background, execute the following (here, spaces are ignored):

Cadence User Interface SKILL Reference

Icon Functions

```
myColorArray = hiCreateColorArray( )
myColorArray[0] = hiMatchColorByName( "thistle")
; 'a' denotes thistle
myColorArray[12] = hiMatchColor( list( 1000 0 0 ))
; 'm' denotes red
myColorArray[4] = hiMatchColor( list( 0 0 1000))
; 'e' denotes blue
iconString = "    1    1    1    1    1    1    1    1\
1    a    a    a    a    a    a    1\
1    a    e    e    e    e    a    1\
1    a    e    m    m    e    a    1\
1    a    e    m    m    e    a    1\
1    a    e    e    e    e    a    1\
1    a    a    a    a    a    a    1\
1    1    1    1    1    1    1    1"
myIcon = hiStringToIcon (myColorArray iconString 8 8)
```

Related Topics

[Icon Functions](#)

[hiCreateColorArray](#)

[hiMatchColorByName](#)

[hiStringToIcon](#)

[hiSetDisabledIcon](#)

hiSetDisabledIcon

```
hiSetDisabledIcon(  
    g_icon  
    g_disabledIcon  
)  
=> t / nil
```

Description

Specifies a disabled icon for an icon, which is displayed whenever the icon is disabled. This function only applies to icons used in menus, not to icons used in form fields or buttons.

This function must be called before the menu is instantiated for the disabled icon to be used.

Arguments

<i>g_icon</i>	Specifies the icon.
<i>g_disabledIcon</i>	Specifies the disabled icon.

Values Returned

t	The function was successful.
nil	There was an error.

Related Topics

[Icon Functions](#)

[hiMatchColorByName](#)

[hiStringToIcon](#)

nameToColor

```
nameToColor(  
    S_colorName  
)  
=> l_RGBColor / nil
```

Description

Specifies the color name as string or symbol.

This function returns a list of three integers representing the red, green, and blue (RGB) components of the color in Virtuoso color coordinates (ranging from 0 to 1000, as opposed to the standard RGB coordinates that range from 0 to 255).

Argument

<i>S_colorName</i>	A color name string or symbol.

Values Returned

<i>l_RGBColor</i>	A list of the RGB values in Virtuoso color coordinates, which can be passed to the <code>hiMatchColor</code> function or the other Virtuoso color functions that take an RGB list.
<i>nil</i>	The color name is not recognized and the warning message is displayed.

Examples

Returns the list of RGB values for the color, `red`.

```
nameToColor("red")  
=> (1000 0 0)
```

Returns the list of RGB values for the color, `blue`.

```
nameToColor("blue")  
=> (0 0 1000)
```

Returns the list of RGB values for the color symbol, `#ff0000`.

```
nameToColor("#ff0000")  
=> (1000 0 0)
```

Cadence User Interface SKILL Reference

Icon Functions

Returns `nil` and displays the warning message because the color name is not valid.

```
nameToColor("seafoam")  
=> *WARNING* nameToColor: could not parse color string "seafoam"  
=> nil
```

Returns the list of RGB values for the color, `seagreen`.

```
nameToColor("seagreen")  
=> (180 545 341)
```

Related Topics

[Icon Functions](#)

[hiStringToIcon](#)

Form Functions

This topic provides a list of Cadence® SKILL functions associated with forms.

The SKILL functions that let you manage requirements in the forms can be broadly classified in the following categories:

- hiLayout Form Functions
- Form and Field Functions

To access form and field data through SKILL, the “→” operator is supported. This may be used to access (read only) the following data (where applicable) from:

```
forms:      hiFormSym
fieldList
_formType
_doneAction
_font
dialogStyle

form fields: hiFieldSym
prompt
value
lastValue
defValue
fieldType
choices
modified
```

Do not create fields with any of these symbol names, as this can cause a conflict in the form creation.

The following field data may be set through *formHandle->fieldHandle*.

```
value
defValue
```

For example:

```
hiFormHandle->hiFieldHandle->value = newValue
```

The new settings are immediately reflected in the appearance of the form.

Cadence User Interface SKILL Reference

Form Functions

In the above examples, the *formHandle* and *fieldHandle* symbols reference data in the form. They are the SKILL symbols that were specified as arguments to `hiCreateAppForm()` and `hiCreate...Field`.

Form Layout

Form layout is determined by the presence of optional two-dimensional (2D) attributes when the fields of the form are specified. A form or menu is 2D if attributes are specified for each field. If you specify 2D attributes for one field, you must specify them for all fields. If these attributes are not specified, the form is laid out in a one-dimensional column. Refer to the following sections for a discussion of laying out two-dimensional forms.



Video

To view the demonstration on creating modular and re-sizable forms, see [How to use the hi*.Layout Functions to Create Modular and Re-sizable Forms](#) video.

Access to this video depends on the availability of a web browser and a Cadence Online Support account.

One Dimensional Form Layout

The following functions create and display a one-dimensional in standard form:

```
someInt = hiCreateIntField(  
?name 'numScoops  
?prompt "Number of scoops (0..5)"  
?value 2  
?defValue 1  
?range '(0 5) )  
  
myCyclic = hiCreateCyclicField(  
?name 'flavor  
?prompt "Flavor"  
?value "chocolate"  
?choices list("chocolate"  
"strawberry" "vanilla"  
"peppermint" "coffee"  
"green_tea") )  
  
mytoggle = hiCreateToggleField(  
?name 'toppings  
?prompt "Toppings"  
?choices list('wCream "Whipped  
cream?")  
list('nuts "any nuts?")  
'(jimmies))  
?value '(t t nil)  
?numSelect 3 )  
  
aRadio = hiCreateRadioField(  
?name 'coneSize  
?prompt "Cone Size"  
?value "Large"  
?defValue "Small"  
  
?choices list("Small" "Medium"  
"Large" ) )
```

Cadence User Interface SKILL Reference

Form Functions

```
hiCreateAppForm( ?name 'iceCreamForm
?formTitle "Ice Cream"
?callback "buildIceCreamCone()"
?fields list( someInt myCyclic mytoggle aRadio )
?help "cream" )

status = hiDisplayForm( iceCreamForm )
```

The user can change the values of this form through the Command Interpreter Window. The user has a handle into the form and each of its fields through what is created by `hiCreateAppForm` or `hiCreateForm( )`. For example, the user can change the number of scoops from 2 to 5 by typing

```
iceCreamForm->numScoops->value = 5
```

The cone size can be altered to medium by typing

```
iceCreamForm->coneSize->value = "medium"
```

Two Dimensional Form Layout

The two-dimensional layout of form fields may be obtained by specifying 2D attributes. The specification of these attributes is optional. A form or menu is 2D if attributes are specified for each field description when creating a form or two-dimensional menu. They are not specified when creating fields themselves. If you specify 2D attributes for one field, you must specify them for all fields. If 2D attributes are not specified, form fields are sequentially laid out in a one-dimensional column.

Through two-dimensional attributes, you are given the flexibility of exact placement of fields. No geometry management is provided, and the fields appear as specified. 2D fields may therefore be specified in any order. Since field coordinates are implicitly obeyed, it is possible to place fields on top of others. Keep this in mind while specifying field locations.

2D attributes are specified in pixel values. The attributes for each field may vary, depending upon how much flexibility you are given. The minimal set of required attributes includes *x*, *y*, *width*, and *height*. The attributes *x* and *y* are coordinates of the upper-left corner of the field, relative to the upper-left (0,0) corner of the form (below the form banner) or menu. The attributes *width* and *height* specify the width and height of the field. An additional, optional attribute lets you specify the prompt box width or title height. These attributes are described in more detail below.

2D attributes are specified at the time the form or menu is created. The attributes should be appended to a list, where the first item of the list is the form field. For example, a 2D field attribute is specified in the following list:

```
list( r_fieldDescription
      x_xpos : x_ypos
```

Cadence User Interface SKILL Reference

Form Functions

```
x_width : x_height  
[x_promptBoxWidthOrTitleHeight] )
```

r_fieldDescription

Field description returned from one of the field creation routines.

x_xpos Upper-left x coordinate in pixels of the field relative to the upper-left corner of the form.

x_ypos Upper-left y coordinate in pixels of the field relative to the upper-left corner of the form.

x_width Width of the field in pixels.

x_height Height of the field in pixels.

x_promptBoxWidthOrTitleHeight

For fields that have a prompt and that are laid out horizontally, such as a string field or spin box field, this attribute is the width of the prompt box in pixels. For fields that have a top-down layout, such as the report field and the scroll region field, this attribute is the height of the title box in pixels (the title is displayed at the top of the title box).

Use this attribute only if the prompt or title is not " ". If the prompt or title is " ", no prompt or title box is created. If a prompt or title is specified, this attribute is required.

The value of this attribute should be less than the total width of the field (specified by *x_width*) or less than the total height of the field (specified by *x_height*), as applicable, and it cannot be a negative number.

This attribute should not be used for those fields that have no prompts, such as `hiCreateLabel` and `hiCreateButton`.

This attribute is optional for fields created with `hiCreateBooleanButton`. If you do not specify it, the right edge of the toggle button appears at the right edge of the field — $x\_xpos + x\_width$. However, if you set the `?buttonLocation` argument of `hiCreateBooleanButton` to 'left, the toggle button is placed at the left edge of the field.

Cadence User Interface SKILL Reference

Form Functions

For certain 2D attributes, you may specify a constant which takes on a value determined by the size of the form. For this purpose, `hicFormHeight`, `hicFormWidth`, and `hicLineHeight` are available.

<code>hicFormHeight</code>	Used to indicate that the height of this field should span the height of the form. This value is computed at the time the form is created.
<code>hicFormWidth</code>	Used to indicate that the width of this field should span the width of the form. This value is computed at the time the form is created.
<code>hicLineHeight</code>	Used to indicate a default line height value.

Field Attachments

Use field attachments if you want the fields in a two-dimensional form or scroll region to be automatically resized when the form is resized. One-dimensional forms do not support field attachments.

Field attachments are used in conjunction with two-dimensional attributes; they determine how the two-dimensional attributes of a field are used to position that field in the form. Like two-dimensional attributes, field attachments need to be specified when you create a form.

If you want to use field attachments, you must do the following when you create a form:

- Specify the initial size of the form by providing a list of width and height for the `g_initialSize` argument.
- Specify two-dimensional attributes for each field entry in the `l_fieldEntries` argument.
- Provide field attachments for each field listed in `l_fieldEntries`.

You specify field attachments by using the `?attachmentList` `l_fieldAttachments` argument in the function `hiCreateAppForm`. This argument takes a list of attachments, which are applied to the corresponding fields in the `l_fieldEntries` argument. Each attachment is a bitwise OR of a list of constants. Each constant is the attachment specification for a side of the field—left, right, top, or bottom.

You can specify either positional attachments or percentage attachments for the sides of a field.

If you want the left and top edges of a field to be at a fixed distance from the upper left corner of the form, but the width and height to grow as a percentage of the size of the form, use the following field attachments for the field:

Cadence User Interface SKILL Reference

Form Functions

`hicTopPositionSet | hicLeftPositionSet | hicRightPercentSet | hicBottomPercentSet`

The following routine creates a one-dimensional form containing a button field and a cyclic field:

```
hiCreateAppForm( ?name 'myForm
  ?formTitle "formName"
  ?fields list( buttonField cyclicField )
  ?callback "doneCallback()"
  ?help "formHelp" )
```

The following routine creates a two-dimensional form containing a field and a cyclic field:

```
hiCreateAppForm( ?name 'myForm
  ?formTitle "formName"
  ?fields list(
    list( buttonField x:y width:height )
    list( cyclicField x2:y2 width2:height2
          promptWidth ))
  ?callback "doneCallback()"
  ?help "formHelp" )
```

The following routine creates a two-dimensional form with field attachments:

```
hiCreateAppForm( ?name 'myForm
  ?formTitle "formName"
  ?fields list(
    list( buttonField x:y width:height )
    list( cyclicField x2:y2 width2:height2 promptWidth)
  )
  ?attachmentList list( hicTopPositionSet|hicLeftPositionSet|
    hicRightPercentSet|hicBottomPercentSet nil)
  ?initialSize list( width height)
  ?callback "doneCallback()"
  ?help "formHelp" )
```

The following routine creates a 2D menu:

```
hiCreate2DMenu( 'myMenu
  "menuName"
  list( list( buttonField x:y width:height )
    list( labelField x2:y2 width2:height2 ))
  "menuHelp" )
```

In the above examples, *buttonField* is the value returned from `hiCreateButton`, *cyclicField* is the value returned from `hiCreateCyclicField`, and *labelField* is the value returned from `hiCreateLabel`. The creation of these fields is shown below.

Specifying fields for 2D menus is the same as that for 2D forms, but only button and label fields created with `hiCreateButton` and `hiCreateLabel` are accepted. 2D menus are displayed in the same way as popup menus, with `hiDisplayMenu`.

```
buttonField = hiCreateButton( ?name 'someButton
  ?buttonText "More Options"
  ?callback "printf(\"pressed\")" )
cyclicField = hiCreateCyclicField( ? name 'cyclic
  ?prompt "Cycle through:"
```

Cadence User Interface SKILL Reference

Form Functions

```
?value "a"
?choices '("a" "b" "c" ) )
labelField = hiCreatelabel( ?name 'myLabel
                           ?labelText "I'm a label"
                           ?justification 'center )
```

To query the extent of the button field, call `hiGetFieldInfo` and look at the contents of *extentList*. For example:

```
extentList = hiGetFieldInfo( myForm 'cyclic )
```

where *extentList* would be of the form

```
(( x2 y2 ) ( width2 height2 ) promptWidth )
```

Positional Attachments

If you specify a positional attachment for a side of the field, that side is placed at a fixed offset from the same side of the form. When the form is resized, that side of the field maintains the fixed distance from the same edge of the form. The form width and height are obtained from the *g_initialSize* argument of the `hiCreateAppForm` function.

Positional attachments are applied to the two-dimensional attributes of a field in the following way:

Field attachment	Action
<code>hiLeftPositionSet</code>	Left edge is placed at <i>x</i> and remains there.
<code>hiRightPositionSet</code>	Right edge is placed at <i>x+width</i> and then the distance between the right edge of the field and the right edge of the form is kept constant.
<code>hiTopPositionSet</code>	Top edge is placed at <i>y</i> and remains there.
<code>hiBottomPositionSet</code>	Bottom edge is placed at <i>y+height</i> and then the distance between the bottom edge of the field and the bottom edge of the form is kept constant.

where *x*, *y*, *width*, and *height* are obtained from the two-dimensional attributes of the field, which are specified in the *l_fieldEntries* argument of `hiCreateAppForm`.

If you do not specify an attachment for either the left or right side (or both), the field width remains constant. If you do not specify an attachment for the top or bottom side (or both), the field height remains constant.

If you specify both positional and percentage attachments for a side, the positional attachment takes precedence.

Cadence User Interface SKILL Reference

Form Functions

Percentage Attachments

If you specify a percentage attachment for a side, that side is initially placed the same way as with a positional attachment; however, when the form is resized, that side maintains the same percentage placement within the form. For example, if you specify `hicRightPercentSet`, and the x location of the field is 10, the width of the field is 90, and the width of the form (as specified by `g_initialSize`) is 200, then the right side of the field is initially placed at 100 and when the form is resized the right side of the field always be kept halfway between the left and right sides of the form, that is, at the 50% position.

Percentage attachments are applied to the two-dimensional attributes of a field in the following way:

Field attachment	Action
<code>hicLeftPercentSet</code>	Left edge is placed at x , and remains at $(x / \text{initialFormWidth}) * \text{formWidth}$.
<code>hicRightPercentSet</code>	Right edge is placed at $x + \text{width}$, and remains at $((x + \text{width}) / \text{initialFormWidth}) * \text{formWidth}$.
<code>hicTopPercentSet</code>	Top edge is placed at y , and remains at $(y / \text{initialFormHeight}) * \text{formHeight}$.
<code>hicBottomPercentSet</code>	Bottom edge is placed at $y + \text{height}$, and remains at $((y + \text{height}) / \text{initialFormHeight}) * \text{formHeight}$.

where x , y , width and height are obtained from the two-dimensional attributes of the field (specified in the `l_fieldEntries` argument of `hiCreateAppForm`), and `formWidth` and `formHeight` are obtained from the `g_initialSize` argument of `hiCreateAppForm`.

If you do not specify an attachment for either the left or right side (or both), the field width remains constant. If you do not specify an attachment for the top or bottom side (or both), the field height remains constant.

If you specify both positional and percentage attachments for a side, the positional attachment takes precedence.

Related Topics

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiLayout Form Functions

Form and Field Functions

hiLayout Form Functions

This topic provides a list of Cadence® SKILL functions associated with hiLayout forms.

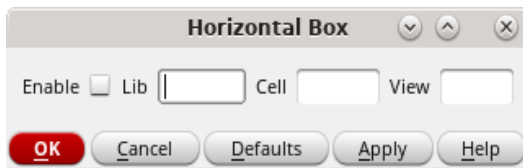
Use these forms to specify the fields of a layout container which gets implemented using Qt QLayout class objects. Layout containers can be specified hierarchically, allowing complex forms to be easily created without specifying pixel locations or sizes. The Qt layout manager automatically adjusts the field sizes and positions to accommodate the font size and form size changes, and it adds scrollbars, if needed.

The hiLayout form is a 2D form, the relationship between the field locations and sizes are controlled by the specification of layout containers within which the fields and other layout objects get placed.

The hiLayout form uses one of the following layouts, which act as a container containing fields or other layouts nested within them:

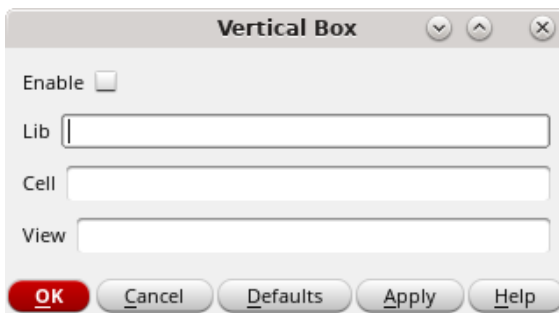
- **Horizontal Box Layout**

Arranges fields and layouts horizontally. The related SKILL function is [hiCreateHorizontalBoxLayout](#).



- **Vertical Box Layout**

Arranges fields and layouts vertically. The related SKILL function is [hiCreateVerticalBoxLayout](#).



- **Grid Layout**

Cadence User Interface SKILL Reference

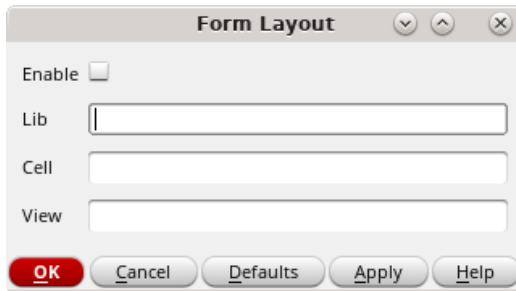
Form Functions

Arranges fields and layouts in a table with fields aligned in a grid. The related SKILL function is [hiCreateGridLayout](#).



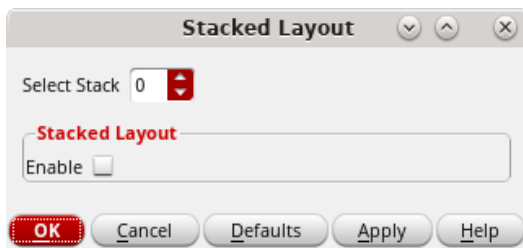
■ Form Layout

Creates a 1D form without specifying the coordinates. The related SKILL function is [hiCreateFormLayout](#).



■ Stacked Layout

Controls the stacking of fields or layouts to make them visible one at a time. The related SKILL function is [hiCreateStackedLayout](#).



The advantage of `hiLayout` forms over the older style 2D forms is that the field layout and automatic resizing is handled automatically without calculating pixel coordinates and sizes. Font size changes are handled automatically. Additionally, improved support for navigating fields is provided using the keyboard only.

Cadence User Interface SKILL Reference

Form Functions

The disadvantage is that the fields with associated prompts may not get properly aligned for the complex form layouts. This can be mitigated by not specifying a prompt for a field and creating a separate label field for the prompt, particularly when using grid layouts.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- hiDeleteField
- hiAddField
- hiAddFields
- hiCreateFormLayout
- hiCreateGridLayout
- hiCreateHorizontalBoxLayout
- hiCreateStackedLayout
- hiCreateVerticalBoxLayout
- hiCreateLayoutForm
- hiDeleteField
- hiDeleteFields
- hiDeleteItemFromLayout

hiAddItemToLayout

```
hiAddItemToLayout(  
    g_formRegionOrLayout  
    g_item  
    [ ?layoutControls l_layoutControls ]  
    [ ?toLayout s_layoutName ]  
)  
=> t / nil
```

Description

Adds the required controls to the layout.

Arguments

g_formRegionOrLayout

Any of the following:

- An instantiated hiLayout form
- Scroll region or tab page in the hiLayout form
- hiLayout

g_item

A field, hiLayout, or spacer item to be added to the layout.

The spacer items are:

- `'col_min_width x_column x_width`: Specifies the minimum width for a column.
- `'row_min_height x_row x_height`: Specifies the minimum height for a row.
- `'col_stretch x_column x_stretch_factor`: Specifies the horizontal stretch factor for a column.
- `'row_stretch x_row x_stretch_factor`: Specifies the vertical stretch factor for a row.

?layoutControls *l_layoutControls*

Cadence User Interface SKILL Reference

Form Functions

The controls appropriate to the layout type to which the items are being added. If these items are being added to a grid layout, then the row and column controls must be specified.

The required layout controls are:

- `'row x_row`: Specifies the row in which the field or layout is placed. The topmost row is 0.
- `'col x_column`: Specifies the column in which the field or layout is placed. The leftmost column is 0.

`?toLayout s_layoutName`

The `hiLayout` symbol for a `hiLayout` contained within the form or region.

The optional layout control is:

`'prompt t | nil | t_prompt`:

- If `t` is specified then the label portion of the form layout row stays occupied by the field prompt (if any) or an empty string if either the field has no prompt or it is a layout object.
- If `nil` is specified then the field prompt gets ignored (if any) and the field or layout spans both the label and field column.
- If `t_prompt` is specified then it becomes the label for the row and any prompt defined for the field is ignored.
- If no value is specified then the field prompt is used (if any) or a blank prompt is assumed.

This argument is ignored if `g_formRegionOrLayout` is an `hiLayout`. If `g_formRegionOrLayout` is a form or region and `s_layoutName` is not specified, then the main `hiLayout` of the form or region is used.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The operation is successfully completed.
nil	The operation is not completed.

Related Topics

[hiLayout Form Functions](#)

hiAddField

```
hiAddField(  
    g_formRegionOrLayout  
    g_field  
    [ x_attachment | s_layout [ l_field_controls ] ]  
)  
=> t / nil
```

Description

Adds a field to a form. The form must already exist, and the field must have been created with one of the field creation routines.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in the hiLayout form
- hiLayout

g_field

The field defstruct, user type, hiLayout defstruct, or list of fields, such as *x:y* location, *width:height*, and prompt width when being added to a 2D form.

x_attachment

The attachment flags if the field is being added to an attachment form or region.

s_layout

The symbol of the layout, field, or hiLayout is being added to instead of the main layout of the form or region.

l_field_controls

The hiLayout field controls specified for the field or hiLayout are added to the field only if these controls are added to the hiLayout form.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The field is added.
nil	The field is not added. An error message is displayed.

Related Topics

[hiLayout Form Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiAddFields

```
hiAddFields(  
    g_formRegionOrLayout  
    l_items  
    [ g_attachmentsOrLayout ]  
)  
=> t / nil
```

Description

Adds fields to a form. The form must already exist, and the fields must have been created with one of the field creation routines.

Arguments

g_formRegionOrLayout

The form, scroll region, tab page, or hiLayout that the fields and/or layouts are added to.

l_items

The *l_items* list is in the same format as fields list provided to hiCreateAppForm for the form type being added to in the 1D, 2D, or attachment form.

If the form is an hiLayout form, the *l_items* list is in the same format as for hiCreate*Layout functions used to create the layout in which the items are being added to.

g_attachmentsOrLayout

If *g_attachmentsOrLayout* is specified, it must be an attachment list if the form is an attachment form. Otherwise, for an hiLayout form, it can be the layout within the specified *g_formRegionOrLayout* to which the items gets added. For hiLayout forms, the new fields are added to the end of the layout unless it is a grid layout in which the grid location controls must be included in the *l_items* list.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The field is added.
nil	The field is not added.

Related Topics

[hiLayout Form Functions](#)

hiCreateFormLayout

```
hiCreateFormLayout(  
    s_name  
    [ ?items l_items ]  
    [ ?frame g_frame ]  
    [ ?margins l_margins ]  
    [ ?spacing g_spacing ]  
    [ ?label_align g_labelAlign ]  
    [ ?horiz_align g_horizAlign ]  
    [ ?vert_align g_vertAlign ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?scrollable g_scrollable ]  
)  
=> r_hiLayout
```

Description

Creates a form layout by specifying the labels, margins, spacing, and alignment.

Arguments

<i>s_name</i>	The symbol name of hiLayout.
?items <i>l_items</i>	Lists the fields and hiLayouts available in the hiLayout.
?frame <i>g_frame</i>	Any of the following values: ■ <i>nil</i>: (Default) For an unframed layout. ■ <i>t</i>: For an untitled frame. ■ <i>t_title</i>: For a framed layout with a title.
?margins <i>l_margins</i>	Accepts integer values or <i>nil</i> for 'left', 'top', 'right', and 'bottom' (in the same sequence) used to apply margins. If <i>nil</i> is specified, then the default margins are applied.
?spacing <i>g_spacing</i>	

Cadence User Interface SKILL Reference

Form Functions

Any of the following values:

`nil`: To apply the default spacing, which is inherited from its parent `hiLayout` or from the style.

Single integer: To specify the same horizontal and vertical spacing

List of two integers: To specify horizontal and vertical spacing (in the same sequence).

`?label_align g_labelAlign`

Specifies the horizontal alignment for the labels in the "label" column. If `nil` is specified, then the default label alignment is determined by the style.

Values: 'none, 'left (default), 'center, 'right, or 'justify. 'justify is specifically used only for text.

`?horiz_align g_horizAlign`

Specifies the horizontal alignment for the "field" column and can be set to 'none, 'left, 'center, 'right, or 'justify. 'justify is specifically used only for text.

Default value: none

`?vert_align g_vertAlign`

Specifies the vertical alignment for the "field" column and can be set to 'none, 'top, 'center, or 'bottom.

Default value: 'none

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?scrollable g_scrollable`

Adds a scroll bar in the layout if the content is large and cannot be accommodated. The default value is `nil`.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

r_hiLayout The defstruct for the horizontal box hiLayout and can be placed in an hiLayout form, scroll region, or tab page.

Examples

Creates a form layout that contains three text input fields for the first, middle, and last names, with a frame titled "Enter Full Name" using the default margins, spacing, and alignment

```
MyNameLayout = hiCreateFormLayout( 'MyNameLayout ?frame "Enter Full Name" ?items
list (
    hiCreateStringField( ?name 'MyFirstName ?prompt "First" )
    hiCreateStringField( ?name 'MyMiddleName ?prompt "Middle" )
    hiCreateStringField( ?name 'MyLastName ?prompt "Last" ) ) )
```

Specifies the field options of the Form Layout form.

```
procedure (ExFormLayout ()
    let ((formLay)
        formLay=hiCreateFormLayout (
            'formLay
            ?items
            list (
                hiCreateBooleanButton(
                    ?name 'enable
                    ?buttonText "Enable"
                )
                hiCreateStringField(
                    ?name 'lib
                    ?prompt "Lib"
                )
                hiCreateStringField(
                    ?name 'cell
                    ?prompt "Cell"
                )
                hiCreateStringField(
                    ?name 'view
                    ?prompt "View"
                )
            )
        )
        hiCreateLayoutForm (
            'ExFormLayoutForm
```

Cadence User Interface SKILL Reference

Form Functions

```
"Form Layout"  
formLay  
)  
hiDisplayForm (ExFormLayoutForm)  
)  
)
```

Related Topics

[hiLayout Form Functions](#)

hiCreateGridLayout

```
hiCreateGridLayout(  
    s_name  
    [ ?items l_items ]  
    [ ?frame g_frame ]  
    [ ?margins l_margins ]  
    [ ?spacing g_spacing ]  
    [ ?horiz_align g_horizAlign ]  
    [ ?vert_align g_vertAlign ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?scrollable g_scrollable ]  
)  
=> r_hiLayout
```

Description

Creates a grid layout by specifying the margins, spacing, and alignment.

Arguments

<i>s_name</i>	The symbol name of hiLayout.

Cadence User Interface SKILL Reference

Form Functions

`?items l_items`

The list of fields, hiLayouts, and spacer items contained within hiLayout.

Each entry in the items list can be a list containing the field or hiLayout followed by the required and optional field controls, or a list containing the spacer items.

The required field controls are:

- `'row x_row`: Specifies the row in which the field or layout is placed. The topmost row is 0.
- `'col x_column`: Specifies the column in which the field or layout is placed. The leftmost column is 0.

The optional field controls are:

- `'row_span x_row_span`: The number of rows to span starting at `x_row`. A value of -1 indicates that the span extends to the bottom row.
- `'col_span x_column_span`: The number of columns to span starting at `x_column`. A value of -1 indicates that the span extends to the rightmost column.
- `'horiz_align s_horiz_alignment`: The `s_horiz_alignment` can be set to 'none', 'left', 'right', 'center, or 'justify (for text).
- `'vert_align x_vert_alignment`: The `s_vert_alignment` can be set to 'none', 'top', 'bottom, or 'center.

The optional spacer items are:

- `'col_min_width x_column x_width`: Specifies the minimum width for a column.
- `'row_min_height x_row x_height`: Specifies the minimum height for a row.
- `'col_stretch x_column x_stretch_factor`: Specifies the horizontal stretch factor for a column.
- `'row_stretch x_row x_stretch_factor`: Specifies the vertical stretch factor for a row.

Cadence User Interface SKILL Reference

Form Functions

<code>?frame</code> <i>g_frame</i>	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.
<code>?margins</code> <i>l_margins</i>	Accepts integer values or <code>nil</code> for 'left', 'top', 'right', and 'bottom' (in the same sequence) used to apply margins. If <code>nil</code> is specified, then the default margins are applied.
<code>?spacing</code> <i>g_spacing</i>	Specifies the integer value to apply for horizontal spacing. If <code>nil</code> is specified, then the default spacing is applied, which is inherited from its parent <code>hiLayout</code> or from the style.
<code>?horiz_align</code> <i>g_horizAlign</i>	Specifies the horizontal alignment and can be set to 'none', 'left', 'center', 'right', or 'justify'. 'justify' is used only for text. Default value: <code>none</code>
<code>?vert_align</code> <i>g_vertAlign</i>	Specifies the vertical alignment and can be set to 'none', 'top', 'center', or 'bottom'. Default value: <code>'none</code>
<code>?enabled</code> <i>g_enabled</i>	Specifies whether the fields in the layout are enabled. The default value is <code>t</code>.
<code>?invisible</code> <i>g_invisible</i>	Specifies whether the layout and the fields within are invisible. The default value is <code>nil</code>.
<code>?scrollable</code> <i>g_scrollable</i>	Adds a scroll bar in the layout if the content is large and cannot be accommodated. The default value is <code>nil</code>.

Cadence User Interface SKILL Reference

Form Functions

Value Returned

r_hiLayout The defstruct for the horizontal box hiLayout and can be placed in an hiLayout form, scroll region, or tab page.

Examples

Creates a grid layout using the default margins, spacing, and alignment. It contains three text input fields for the first, middle, and last names with a frame title, Enter Full Name.

```
MyNameLayout = hiCreateGridLayout( 'MyNameLayout ?frame "Enter Full Name" ?items
list (
    list( hiCreateLabel( ?name 'MyFirstLabel ?labelText "First" ) 'row 0 'col 0 )
    list( hiCreateStringField( ?name 'MyFirstName ) 'row 0 'col 1 )
    list( hiCreateLabel( ?name 'MyMiddleLabel ?labelText "Middle" ) 'row 1 'col 0 )
    list( hiCreateStringField( ?name 'MyMiddleName ) 'row 1 'col 1 )
    list( hiCreateLabel( ?name 'MyLastLabel ?labelText "Last" ) 'row 2 'col 0 )
    list( hiCreateStringField( ?name 'MyLastName ) 'row 2 'col 1 )
    list( 'col_stretch 1 1 )
) )
```

Specifies the field options of the Grid Layout form.

```
procedure (ExGrid())
let ((grid)
    grid=hiCreateGridLayout (
        'grid
        ?items
        list(
            list(
                hiCreateLabel(
                    ?name 'enableLabel
                    ?labelText "Enable"
                )
                'row 0 'col 0
            )
            list(
                hiCreateBooleanButton(
                    ?name 'enable
                    ?buttonText " "
                )
                'row 0 'col 1
            )
            list(
                hiCreateLabel(
```

Cadence User Interface SKILL Reference

Form Functions

```
        ?name 'libLabel
        ?labelText "Lib"
    )
    'row 1 'col 0
)
list(
    hiCreateStringField(
        ?name 'lib
    )
    'row 1 'col 1
)
list(
    hiCreateLabel(
        ?name 'cellLabel
        ?labelText "Cell"
    )
    'row 2 'col 0
)
list(
    hiCreateStringField(
        ?name 'cell
    )
    'row 2 'col 1
)
list(
    hiCreateLabel(
        ?name 'viewLabel
        ?labelText "View"
    )
    'row 3 'col 0
)
list(
    hiCreateStringField(
        ?name 'view
    )
    'row 3 'col 1
)
list(
    hiCreateLabel(
        ?name 'explanation
        ?labelText "Fields can also span across multiple columns or rows"
```

Cadence User Interface SKILL Reference

Form Functions

```
        )
        'row 4 'col 0 'col_span 2
    )
    ; only the entry fields are stretchable
    ; the labels do not stretch
    list('col_stretch 0 0)
    list('col_stretch 1 1)
    )
    ?frame "Grid Layout"
)
hiCreateLayoutForm(
    'ExGridLayout
    "Grid"
    ; surround the grid with a vertical box layout
    ; to provide a stretch_item below to prevent fields
    ; spreading vertically when the form is stretched
    hiCreateVerticalBoxLayout(
        'vbox
        ?items
        list(
            grid
            list('stretch_item 1)
        )
    )
)
hiDisplayForm(ExGridLayout)
)
```

Related Topics

[hiLayout Form Functions](#)

hiCreateHorizontalBoxLayout

```
hiCreateHorizontalBoxLayout(  
    s_name  
    [ ?items l_items ]  
    [ ?frame g_frame ]  
    [ ?margins l_margins ]  
    [ ?spacing g_spacing ]  
    [ ?horiz_align g_horizAlign ]  
    [ ?vert_align g_vertAlign ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?scrollable g_scrollable ]  
)  
=> r_hiLayout
```

Description

Creates a horizontal box layout by specifying the margins, spacing, and alignment.

Arguments

<i>s_name</i>	The symbol name of the hiLayout.

Cadence User Interface SKILL Reference

Form Functions

<code>?items l_items</code>	The list of fields, hiLayouts, and spacer items contained within hiLayout. Each entry in the items list can be a field or hiLayout, a list containing the field or hiLayout followed by the optional field controls, or a list containing the spacer items. The optional field controls are: ■ <code>'stretch x_stretch_factor</code>: Stretchiness compared with other fields or layouts in the direction of the box layout. The default value is 0. ■ <code>'horiz_align s_horiz_alignment</code>: The <code>s_horiz_alignment</code> can be set to 'none', 'left', 'right', 'center, or 'justify (for text). ■ <code>'vert_align x_vert_alignment</code>: The <code>s_vert_alignment</code> can be set to 'none', 'top', 'bottom, or 'center. The spacer items are: ■ <code>'spacer_item x_size</code>: Specifies the additional fixed-size space between two items. ■ <code>'stretch_item x_stretch_factor</code>: Specifies the additional stretchy-space with minimum size of 0 between two items.
<code>?frame g_frame</code>	It accepts 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.
<code>?margins l_margins</code>	Accepts integer values or <code>nil</code> for 'left', 'top', 'right, and 'bottom (in the same sequence) used to apply margins. If <code>nil</code> is specified, then the default margins are applied.
<code>?spacing g_spacing</code>	Specifies the integer value to apply horizontal spacing. If <code>nil</code> is specified then the default spacing is applied, which is inherited from its parent hiLayout or from the style.
<code>?horiz_align g_horizAlign</code>	

Cadence User Interface SKILL Reference

Form Functions

Specifies the horizontal alignment and can be set to 'none, 'left, 'center, 'right, or 'justify. 'justify is specifically used only for text.

Default value: 'none

`?vert_align g_vertAlign`

Specifies the vertical alignment and can be set to 'none, 'top, 'center, or 'bottom.

Default value: 'none

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?scrollable g_scrollable`

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`.

Value Returned

`r_hiLayout`

The defstruct for the horizontal box `hiLayout` and can be placed in an `hiLayout` form, scroll region, or tab page.

Examples

Creates a horizontal box layout using the default margins, spacing, and alignment. It contains three text input fields for the first, middle, and last names with a frame title, Enter Full Name.

```
MyNameLayout = hiCreateHorizontalBoxLayout( 'MyNameLayout ?frame "Enter Full Name"
?items list(
    hiCreateStringField( ?name 'MyFirstName ?prompt "First" )
    hiCreateStringField( ?name 'MyMiddleName ?prompt "Middle" )
    hiCreateStringField( ?name 'MyLastName ?prompt "Last" ) ) )
procedure (ExHorizontalBox())
let ( (hbox)
    hbox=hiCreateHorizontalBoxLayout (
```

Cadence User Interface SKILL Reference

Form Functions

```
'hbox
?items
  list(
    hiCreateBooleanButton(
      ?name 'enable
      ?buttonText "Enable"
    )
    list(
      hiCreateStringField(
        ?name 'lib
        ?prompt "Lib"
      )
      'stretch 1
    )
    list(
      hiCreateStringField(
        ?name 'cell
        ?prompt "Cell"
      )
      'stretch 1
    )
    list(
      hiCreateStringField(
        ?name 'view
        ?prompt "View"
      )
      'stretch 1
    )
  )
)
hiCreateLayoutForm(
  'ExHorizBox
  "Horizontal Box"
  hbox
)
hiDisplayForm(ExHorizBox)
)
```


Cadence User Interface SKILL Reference

Form Functions

Related Topics

[hiLayout Form Functions](#)

hiCreateStackedLayout

```
hiCreateStackedLayout(  
    s_name  
    [ ?items l_items ]  
    [ ?frame g_frame ]  
    [ ?margins l_margins ]  
    [ ?currentIndex g_currentIndex ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?scrollable g_scrollable ]  
)  
=> r_hiLayout
```

Description

Creates a stacked layout. A stacked layout only shows one of the fields at a time, which can be set using the `currentIndex` argument of the layout.

Arguments

<i>s_name</i>	The symbol name for the layout.
?items <i>l_items</i>	The list of fields and layouts contained within the layout.
?frame <i>g_frame</i>	Any of the following values: ■ <code>nil</code>: (Default) For no frame. ■ <code>t</code>: For an untitled frame. ■ <code>t_title</code>: For a titled frame.
?margins <i>l_margins</i>	Accepts integer values or <code>nil</code> for 'left', 'top', 'right, and 'bottom (in the same sequence) used to apply margins. If <code>nil</code> is specified, then the default margins are applied.
?currentIndex <i>g_currentIndex</i>	Specifies an integer value other than <code>nil</code> that specifies the initial index to display. Default value: 0
?enabled <i>g_enabled</i>	

Cadence User Interface SKILL Reference

Form Functions

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?scrollable g_scrollable`

Adds a scroll bar in the layout if the content is large and cannot be accommodated. The default value is `nil`.

Value Returned

`r_hiLayout` The `hiLayout` defstruct for the stacked layout.

Examples

Creates a stacked layout destruct with `MyStringField` as the first field in the stack, keeping the side margins as the default with the top margin as 5 and the bottom margin as 8.

If `MyForm` contains a grid layout that was created as follows:

```
MyStackedLayout = hiCreateStackedLayout( 'MyStackedLayout ?margins list( nil 5 nil
8 ) ?items list( MyStringField MyReportField MySubLayout ) )
```

Specifies the field options of the Stacked Layout form.

```
procedure(ExStacked())
let((stacked)
  stacked=hiCreateStackedLayout(
    'stacked
    ?items
    list(
      hiCreateBooleanButton(
        ?name 'enable
        ?buttonText "Enable"
      )
      hiCreateStringField(
        ?name 'lib
        ?prompt "Lib"
      )
      hiCreateStringField(
```

Cadence User Interface SKILL Reference

Form Functions

```
        ?name 'cell
        ?prompt "Cell"
    )
    hiCreateStringField(
        ?name 'view
        ?prompt "View"
    )
)
?frame "Stacked Layout"
)
hiCreateLayoutForm(
    'ExStackedForm
    "Stacked Layout"
    hiCreateVerticalBoxLayout(
        'vbox
        ?items
        list(
            ; spin box is to allow the selection of which field
            ; to display in the stacked layout
            hiCreateSpinBox(
                ?name 'stackSelector
                ?prompt "Select Stack"
                ?range 0:3
                ?callback
                "ExStackedForm->stacked->currentIndex=ExStackedForm->stackSelector->value"
            )
            stacked
        )
    )
)
)
hiDisplayForm(ExStackedForm)
)
)
```

Related Topics

hiLayout Form Functions

hiCreateVerticalBoxLayout

```
hiCreateVerticalBoxLayout(  
    s_name  
    [ ?items l_items ]  
    [ ?frame g_frame ]  
    [ ?margins l_margins ]  
    [ ?spacing g_spacing ]  
    [ ?horiz_align g_horizAlign ]  
    [ ?vert_align g_vertAlign ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?scrollable g_scrollable ]  
)  
=> r_hiLayout
```

Description

Creates a vertical box layout by specifying the margins, spacing, and alignment.

Arguments

<i>s_name</i>	The symbol name of hiLayout.

Cadence User Interface SKILL Reference

Form Functions

<code>?items l_items</code>	The list of fields, hiLayouts, and spacer items contained within hiLayout. Each entry in the items list can be a field or hiLayout, a list containing the field or hiLayout followed by the optional field controls, or a list containing the spacer items. The optional field controls are: ■ <code>'stretch x_stretch_factor</code>: Stretchiness compared with other fields or layouts in the direction of the box layout. The default value is 0. ■ <code>'horiz_align s_horiz_alignment</code>: The <code>s_horiz_alignment</code> can be set to 'none', 'left', 'right', 'center, or 'justify (for text). ■ <code>'vert_align x_vert_alignment</code>: The <code>s_vert_alignment</code> can be set to 'none', 'top', 'bottom, or 'center. The spacer items are: ■ <code>'spacer_item x_size</code>: Specifies the additional fixed-size space between two items. ■ <code>'stretch_item x_stretch_factor</code>: Specifies the additional stretchy-space with minimum size of 0 between two items.
<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.
<code>?margins l_margins</code>	Accepts integer values or <code>nil</code> for 'left', 'top', 'right, and 'bottom (in the same sequence) used to apply margins. If <code>nil</code> is specified, then the default margins are applied.
<code>?spacing g_spacing</code>	Specifies the integer value to apply for horizontal spacing. If <code>nil</code> is specified then the default spacing is applied, which is inherited from its parent hiLayout or from the style.
<code>?horiz_align g_horizAlign</code>	

Cadence User Interface SKILL Reference

Form Functions

Specifies the horizontal alignment and can be set to 'none, 'left, 'center, 'right, or 'justify. 'justify is specifically used only for text.

Default value: 'none

`?vert_align g_vertAlign`

Specifies the vertical alignment and can be set to 'none, 'top, 'center, or 'bottom.

Default value: none

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?scrollable g_scrollable`

Adds a scroll bar in the layout if the content is large and cannot be accommodated. The default value is `nil`.

Value Returned

`r_hiLayout`

The defstruct for the vertical box `hiLayout` and can be placed in an `hiLayout` form, scroll region, or tab page.

Examples

Creates a vertical box layout using the default margins, spacing, and alignment. It contains three text input fields for the first, middle, and last names with a frame title, Enter Full Name.

```
MyNameLayout = hiCreateVerticalBoxLayout( 'MyNameLayout ?frame "Enter Full Name"
?items list(
    hiCreateStringField( ?name 'MyFirstName ?prompt "First" )
    hiCreateStringField( ?name 'MyMiddleName ?prompt "Middle" )
    hiCreateStringField( ?name 'MyLastName ?prompt "Last" ) ) )
```

Specifies the field options of the Vertical Box form.

```
procedure (ExVerticalBox())
let ((vbox)
```

Cadence User Interface SKILL Reference

Form Functions

```
vbox=hiCreateVerticalBoxLayout(  
    'vbox  
    ?items  
        list(  
            hiCreateBooleanButton(  
                ?name 'enable  
                ?buttonText "Enable"  
            )  
            hiCreateStringField(  
                ?name 'lib  
                ?prompt "Lib"  
            )  
            hiCreateStringField(  
                ?name 'cell  
                ?prompt "Cell"  
            )  
            hiCreateStringField(  
                ?name 'view  
                ?prompt "View"  
            )  
            ; swallow up any remaining space vertically  
            list('stretch_item 1)  
        )  
    )  
    hiCreateLayoutForm(  
        'ExVertBox  
        "Vertical Box"  
        vbox  
    )  
    hiDisplayForm(ExVertBox)  
)
```

Related Topics

[hiLayout Form Functions](#)

hiCreateLayoutForm

```
hiCreateLayoutForm(  
    s_name  
    t_title  
    r_layout  
    [ ?callback g_callback ]  
    [ ?mapCB g_mapCallback ]  
    [ ?help g_help ]  
    [ ?unmapAfterCB g_unmapAfterCB ]  
    [ ?buttonLayout g_buttonLayout ]  
    [ ?buttonDisabled l_buttonDisabled ]  
    [ ?formType s_formType ]  
    [ ?dialogStyle s_dialogStyle ]  
    [ ?baseName s_baseName ]  
    [ ?plist g_plist ]  
    [ ?initialSize l_initialSize ]  
    [ ?minSize l_minimumSize ]  
    [ ?maxSize l_maximumSize ]  
    [ ?sizePolicy s_sizePolicy ]  
  
    )  
=> r_form / nil
```

Description

Returns the SKILL representation of a layout form with the specified field entries and sets it to the specified name.

Arguments

<i>s_name</i>	The global SKILL symbol used to reference this form.
<i>t_title</i>	Specifies the form title, which is displayed on the top of the form.
<i>r_layout</i>	The main hiLayout of the form.
?callback <i>g_callback</i>	

Cadence User Interface SKILL Reference

Form Functions

Specifies the SKILL functions (callbacks) that execute after you select *Apply*, *OK*, or *Cancel* in the form.

- *g_callback* can be in one of two formats:
 - *g_okAction* or' (*g_okAction*
g_cancelAction)
- *g_okAction* specifies the SKILL function to be executed when the *Apply*, *OK*, *Close*, or *Quit* button is pressed in a form; *g_cancelAction* specifies that the SKILL function to be executed when the *Cancel* button is pressed. Here, if *g_cancelAction* is not specified, no callback is called when the form is canceled.

Both *g_okAction* and *g_cancelAction* can be in one of the following formats:

- *t_callback*, *s_callback*, or *u_callback*
- *t_callback* is the string representation of the SKILL function to be executed. If you specify *t_callback*, the string is evaluated and the callback function is not passed any argument.
- *s_callback* is the symbol of the SKILL function to be called. If you specify *s_callback*, the callback function is passed and *r_form* returned from *hiCreateLayoutForm* as its only parameter.
- *u_callback* is a callback of the function object type. The callback function is passed and *r_form* returned from *hiCreateLayoutForm* as its only parameter.
- After the form is instantiated, you can change the callback with the *hiChangeFormCallback* function.

?mapCB *g_mapCallback*

Specifies the callback to be invoked when the form is mapped, that is, when *hiDisplayForm* is called for a form that is not currently mapped. Specify one of the following:

- The name (string),
- Symbol of the map callback function,
- The function object.

Cadence User Interface SKILL Reference

Form Functions

?help *g_help*

Specify *g_help* as one of the following:

- Single string
- Single symbol
- List of strings
- List of symbols that are used to refer a .tgf help file.

If you do not want to use the default .tgf help file mechanism, you can specify your own callback for the *Help* button.

If you specify a single string or symbol, it is used to set the `hiHelpAppName` property on the symbol that represents the form name. If you specify a list of strings or a list of symbols, the first element in the list is used to set the `hiHelpAppName` property and the second element is used to set the `hiHelpSymName` property.

If you want to specify only the help symbol, specify a list of two strings or symbols, with `nil` as the first element of the list and the help symbol as the second element. This sets the `hiHelpSymName` property without setting the `hiHelpAppName` property.)

When a user clicks *Help* on the form, the `hiHelp` function is called with the following arguments:

appId helpTag

If the `hiHelpAppName` property has been set, it is used as the *appId*, otherwise the application ID of the current window is used as the *appId*. If the `hiHelpSymName` property has been set, it is used as the *helpTag*, otherwise the form name is used as the *helpTag*.

The *appId* and *helpTag* determine the name of the tag file that is called (*appId.tgf*) and the help tag in the file that is accessed.

You can also use the `hiHelpAppName` and `hiHelpSymName` properties directly to specify the help file and help tag that are used to reference the help.

?unmapAfterCB *g_unmapAfterCB*

Cadence User Interface SKILL Reference

Form Functions

If set to `t`, the form is removed (unmapped) from the screen after the callback is executed. If set to `nil`, the form is removed before the callback is invoked. The default value is `nil`. If you use `hiSetCallbackStatus` in your callback routine, you must set the value of `g_unmapAfterCB` to `t` or the callback status is ignored.

?buttonLayout *g_buttonLayout*

Specifies the buttons that appear at the bottom of the form. *g_buttonLayout* can be in one of the following formats:

■ *s_buttonLayout*

Displays a predefined set of buttons, followed by the Help button. You can select from a list of button layouts.

■ `('s_buttonLayout (s_customButtonText
s_customButtonCB) ...)`

Displays a predefined set of buttons, followed by your custom buttons and the Help button. Custom buttons are specified as button text/button callback pairs.

■ `('Empty (s_customButtonText
s_customButtonCB) ...)`

Displays only custom buttons, followed by the Help button.

Specify either *OK*, *Close*, or *Quit* as one of the button names in your list of custom buttons so the form can be closed. You can use `hiFormClose()` as the callback for these buttons.

■ `'Empty`

Does not display any buttons in the form.

The form can be closed either through the window manager or by calling `hiFormDone()`.

Cadence User Interface SKILL Reference

Form Functions

s_buttonLayout specifies a predefined set of buttons. The *Help* button is automatically added with each of these combinations. Select one of the following button layouts:

For non-options forms:

```
'OKCancel  
'OKCancelDef  
'OKCancelLast  
'OKCancelDefLast  
'OKCancelApply  
'ApplyCancel  
'ApplyCancelDef  
'ApplyCancelLast  
'ApplyCancelDefLast  
'Close/'Quit  
'OKCancelDefApply (default)
```

You can use the *Quit* button instead of the *Close* button with the same behavior.

For options forms:

```
'HideCancel  
'HideCancelDef  
'HideCancelLast  
'HideCancelDefLast (default)  
'ApplyHideCancel  
'ApplyHideCancelDef  
'ApplyHideCancelLast  
'ApplyHideCancelDefLast
```

where, *OK* denotes the *OK* button (*hiFormDone* is called when this button is pressed, and *hiFormDone* calls the *g_okAction* callback);

Apply denotes the *Apply* button (*hiFormApply* is called when this button is pressed, and *hiFormApply* calls the *g_okAction* callback);

Cadence User Interface SKILL Reference

Form Functions

`Cancel` denotes the *Cancel* button (`hiFormCancel` is called when this button is pressed, and `hiFormCancel` calls the `g_cancelAction` callback);

`Close` denotes the *Close* or *Quit* button

(`hiFormClose` is called when this button is pressed, and `hiFormClose` calls the `g_okAction` callback. The *Close* and *Quit* buttons are similar to the *OK* button; `hiFormClose` is an alias for `hiFormDone`); `Last` denotes the *Last* button; `Def` denotes the *Defaults* button; `Hide` denotes the *Hide* button.

If there is no *Cancel* button on a form but there is a *Close* or *Quit* button, when you close the form from the window manager or by typing `hiFormCancel` in the CIW, `hiFormDone` is called.

For custom buttons, `s_buttonLayout` or `'Empty` should be listed consisting of button text/button callback pairs:

```
'(s_customButtonText s_customButtonCB)
```

- `s_customButtonText` is the symbol whose printname is the button text. For example, to have a button "Press Me", `s_customButtonText` would be specified as `'Press\Me`. You can use `concat("string")` to get the symbol of a string.
- `s_customButtonCB` is the callback for the button and can be specified as a symbol or string. If it is a symbol, the callback function is passed and `r_form` returned from `hiCreateAppForm` as its only argument. If it is a string, the unchanged string is passed to SKILL to interpret.

Press *Enter* to perform the same action as the first button of the form button list. This is usually `hiFormDone()` for non-options forms and `hiToggleEnterForm()` for options forms. If you specify your own button list, you need to be aware that the action of `<Enter>` is changed. For example, if your button list is `'ApplyCancelDef`, the action of `<Enter>` in a type-in field performs the `hiFormApply()` action.

Use `formDefaultAction` to specify the `buttonText` of a button to link to the `Enter` Key.

```
?buttonDisabled l_buttonDisabled
```

Cadence User Interface SKILL Reference

Form Functions

List of buttons that are disabled when the form appears. You can change this value with the `hiSetFormButtonEnabled` function.

?formType *s_formType*

Possible choices include 'nonoptions (default), or 'options. Options forms must be modeless.

?dialogStyle *s_dialogStyle*

One of the following symbols, which determines the blocking behavior of the form:

'modal

Blocks access to the application while it is displayed. You must dismiss the form in order to continue using the application. (Access to window manager commands or other applications is not blocked.)

'modeless

Does not block access to the application. You can continue using the application without having to close the form.

The default value of this argument is 'modeless.

?baseName *s_baseName*

A string or symbol that defines the base name of the form.

g_plist

Indicates whether or not this is a "property list" form.

?initialSize *l_initialSize*

A list of width and height values. The form is sized to the specified width and height, or to the size needed to accommodate all the fields, whichever is less.

?minSize *l_minimumSize*

The minimum size of the form, specified as a list of width and height or *width:height* values. The minimum size excludes the window manager's title and border. The default minimum width is a few pixels wider than the width of the button banner and the default minimum height is a few pixels more than the height of the button banner.

To change the minimum size, use the `hiSetFormMinMaxSize` function.

Cadence User Interface SKILL Reference

Form Functions

?maxSize *l_maximumSize*

The maximum size of the form, specified as a list of width and height or *width:height*. The maximum size excludes the window manager's title and border. The default maximum size is the size of the screen.

To change the maximum size, use the `hiSetFormMinMaxSize` function.

?sizePolicy *s_sizePolicy*

Controls how the forms can be resized. It includes the following options:

- 'expanding: Allows resizing of form layout in both the directions.
- 'horizontalExpanding: Allows horizontal resizing of the form layout.
- 'verticalExpanding: Allows vertical resizing of the form layout.
- 'fixed: Resizing is not allowed.

Default value: 'expanding

It includes only the interactive resizing of the form. The `sizePolicy` cannot be changed once the form is instantiated or displayed.

Values Returned

r_form Returns the handle of the form. the SKILL representation of a layout form

nil The handle of the form is not returned.

Additional Information

When creating a `hiLayout` form, instead of a list of fields, a single layout container is passed to `hiCreateLayoutForm`. The layouts themselves contain the details of the fields within. A specified layout have a name, which must be unique within the form, tab page or scroll region. Each layout contains not only a list of fields, but options relevant to the layout type for describing spacer items or stretch items and how alignment works within the layout. Fields

Cadence User Interface SKILL Reference

Form Functions

can be added and deleted to layouts similar to 1D and 2D forms, but specifying coordinates are not necessary.

The fields within a layout can be accessed without using the layout container. For example, a field `myField` within a layout `myLayout` on a form `myForm` can be accessed using:

```
myForm->myField->value
```

which makes it simpler to convert existing code using a 2D form to use `hiLayout` containers. Instead, the code does not need to be updated to reflect any new containers which have been inserted. Alternatively, you can use `myForm->myLayout->myField->value`.

The only field type not supported in an `hiLayout` form is a frame field. This is replaced by the `?frame` argument for each layout creation function.

Related Topics

[hiSetFormMinMaxSize](#)

[hiSetFormButtonEnabled](#)

[formDefaultAction](#)

[hiLayout Form Functions](#)

hiDeleteField

```
hiDeleteField(  
    g_formRegionOrLayout  
    s_fieldOrLayout  
)  
=> t / nil
```

Description

Deletes a field from a form.

To delete more than one field from the same form, use `hiDeleteFields`. This is especially true if a contiguous set of one-dimensional form fields are deleted.

Deleting a field from a 2D form removes the instance of the field from the form, and assumes that there is only one instance of this field contained within the form. The field still exists in any other form which contains it.

In a two-dimensional form or menu, no geometry management is performed to `fill` the freed area of the screen. If the field is deleted from a one-dimensional form, the field is removed and the other fields adjusted appropriately.

Arguments

g_formRegionOrLayout

Specifies a form, scroll region, tab page, or hiLayout from which the field is deleted.

s_fieldOrLayout

The symbol name of the field or hiLayout to be deleted. This does not delete the field or layout defstruct or other instantiations of the field or layout.

Values Returned

t

The field is deleted.

nil

The field is not deleted. An error message is displayed.

Related Topics

[hiLayout Form Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiDeleteFields

hiDeleteFields

```
hiDeleteFields(  
    g_formRegionOrLayout  
    l_fieldsOrLayouts  
)  
=> t / nil
```

Description

Deletes multiple fields from a form.

To delete more than one field from the same form, this function should be used instead of multiple calls to `hiDeleteField`.

For one-dimensional forms, `hiDeleteFields` optimizes the deleting of a contiguous list of fields (given in top-to-bottom order) by moving each remaining field once, after all fields have been deleted.

Deleting a field from a 2D form removes the instance of the field from the form, and assumes that there is only one instance of this field contained within the form. The field still exists in any other form which contains it.

In a two-dimensional form or menu, no geometry management is performed to fill the freed area of the screen. If a field is deleted from a one-dimensional form, the field is removed and the other fields adjusted appropriately.

Arguments

g_formRegionOrLayout

Specifies a form, scroll region, tab page, or `hiLayout` from which the field is deleted.

l_fieldsOrLayouts

The symbol name of the field or `hiLayout` to be deleted. This does not delete the field or layout `defstruct` or other instantiations of the field or layout.

Values Returned

`t`

The field is deleted.

`nil`

The field is not deleted. An error message is displayed.

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[hiLayout Form Functions](#)

hiDeleteItemFromLayout

```
hiDeleteItemFromLayout(  
    x_index  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> x_index / nil
```

Description

Deletes an item from hiLayout using the specified index.

Arguments

<i>x_index</i>	The index position at which the item needs to be deleted.
<i>g_formRegionOrLayout</i>	Any of the following: ■ hiLayout form ■ Scroll region or tab page in hiLayout form ■ hiLayout
<i>s_layoutName</i>	Specifies a symbol name of a hiLayout, field in the form, or region.

Values Returned

<i>x_index</i>	The description of all the items deleted from hiLayout.
<i>nil</i>	Specified hiLayout do not exist.

Related Topics

[hiLayout Form Functions](#)

Traversing Fields

You can use the `Tab` key to traverse editable fields in a form. The traversal order is determined by the following (listed in order of precedence, highest-to-lowest):

- The *nextField* property of a field. This is set with the *nextField* argument when the field is created and defines the next field to bring into focus when the `Tab` key is pressed. The *nextField* property has the highest precedence.
- The *tabOrderIsAddOrder* argument value. This is set when a form, scroll region, or tab field is created. When set to `t`, *tabOrderIsAddOrder* sets the *nextField* property on all fields (except those that have the *nextField* property already set to a non-`nil` value) to the order in which the fields are added to the form.
- If *nextField* is not specified or *tabOrderIsAddOrder* is not set, then the traversal order is based on the form layout —left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

You can use `SHIFT+TAB` to traverse the fields backwards. Note, however, that traversing backwards might not always be predictable. For example, in cases where two fields point to the same next field, it is difficult to determine the previous field.

Related Topics

[Creating Context Menus for Fields](#)

[Creating Tool Tips for Fields](#)

Creating Context Menus for Fields

All form fields except cyclic, frame, and separator fields have a context menu, that is, a pop-up menu, which is displayed when you right-click on the field. Context menus are associated with form fields through the following properties:

- `hiContextMenu`
- `hiShowContextMenuCallback`

To set a context menu for a field,

1. Create a context menu with either `hiCreateSimpleMenu` or `hiCreateMenu`.
2. Set the `hiContextMenu` property on the field.
3. (Optional) If you want to set a callback for the context menu, which starts just before the context menu is displayed, set the `hiShowContextMenuCallback` property on the field.

Setting the `hiContextMenu` Property

To set the `hiContextMenu` property on a field, use the following command:

```
hiFormHandle->hiFieldSymbol->hiContextMenu=r_value
```

where *hiFormHandle* is the handle to the form returned by `hiCreateAppForm`, *hiFieldSymbol* is the symbol of the field (set with the `?name` argument when the field was created), and *r_value* is the handle to the pop-up menu returned by either `hiCreateSimpleMenu` or `hiCreateMenu`.

Once the property is set to a valid value, when you click the field, the context menu is displayed at the cursor's position.

Setting the `hiShowContextMenuCallback` Property

With the `hiShowContextMenuCallback` property, you can specify a callback to be invoked just before the context menu is displayed.

To set the `hiShowContextMenuCallback` property, use the following command:

```
hiFormHandle->hiFieldSymbol->hiShowContextMenuCallback=S_value
```


Cadence User Interface SKILL Reference

Form Functions

hiFormHandle is the handle to the form returned by *hiCreateAppForm*.

hiFieldSymbol is the symbol of the field, set with the *?name* argument when the field was created. *S_value* is a symbol or string specifying the callback function to be invoked. If you specify *S_value* as a string, the string is evaluated and no arguments are passed to the callback function. If you specify *S_value* as a symbol, the callback function is passed certain arguments; these arguments can vary depending on the type of field on which the property is being set. The following table lists the arguments for different fields:

Field	Arguments passed to context menu callback function
Tree Table	<i>(r_menu g_formOrRegion s_field s_item)</i>
Other fields	<i>(r_menu g_formOrRegion s_field)</i>

where

r_menu is the value of the *hiContextMenu* property;

g_formOrRegion is either a form (of type *r*) if the field is placed at the top level of the form, that is, it is not within a scroll region or tab page; a scroll region (of type *o*); or a tab page (of type *o*);

s_field is the symbol of the field;

s_item is the symbol of the tree table item which is right-clicked.

Related Topics

[Creating Tool Tips for Fields](#)

[Setting the hiContextMenu Property](#)

Creating Tool Tips for Fields

All form fields can have tool tips, also sometimes referred to as balloon help. A tool tip is a short description of a field that is displayed when the pointer hovers over the field. The tooltip is displayed when the pointer is over either the prompt or the active part of the field. For fields that display multiple options—specifically, radio fields, toggle fields, and button-box fields—different tool tips can be set for each option.

Set tool tips using the `hiToolTip` property on the fields.

To set a tooltip on a field,

- ➔ Set the `hiToolTip` property on the field with the following command:

```
hiFormHandle->hiFieldSymbol->hiToolTip=toolTip
```

where *hiFormHandle* is the handle to the form returned by `hiCreateAppForm`, *hiFieldSymbol* is the symbol of the field (set with the `?name` argument when the field was created), and *toolTip* is one of the following:

- ❑ A string: "*toolTip*"

The tool tip is displayed when the pointer is over the field prompt, if any, or over the active part of the field. An empty string (" ") or `nil` turns off the tool tip.

For example:

```
myStringField=hiCreateStringField(?name 'myStringField)
hiFormHandle->myStringField->hiToolTip="description for myStringField"
```

- ❑ A list of strings: `list("toolTip1" "toolTip2" "toolTip3" ...)`

Specify a list of tool tips for a radio, toggle, or button-box field. The tool tips are assigned to the items in the field in the order in which the items are listed in the `?choices` argument of the field. The first tool tip is assigned to the first item in `?choices`, the second tool tip to the second item, and so on. Use an empty string (" ") or `nil` for an item for which you do not want to display a tooltip.

If the tool tip list has more strings than the number of items in the `?choices` list, the first extra string is assigned to the field prompt. Any other extra tool tips are ignored.

If the `?choices` list has more items than the tool tips list, the tool tips are assigned to the items in the order in which they are listed and the remaining items do not display any tool tips.

For example, in the following case:

```
myRadioField=(?name 'myRadioField ?choices list("item1" "item2"))
```

Cadence User Interface SKILL Reference

Form Functions

```
hiFormHandle->myRadioField->hiToolTip=(list "tooltip1" "tooltip2"
"tooltip3" "tooltip4"))
```

tooltip1 is assigned to item1, tooltip2 is assigned to item2, tooltip3 is assigned to the field prompt, and tooltip4 is ignored.

In the following case:

```
myRadioField=(?name 'myRadioField ?choices list("item1" "item2" "item3"))
hiFormHandle->myRadioField->hiToolTip=(list "tooltip1" "tooltip2"))
```

tooltip1 is assigned to item1, tooltip2 is assigned to item2, and item3 does not have a tool tip.

To turn off a tooltip,

- ➔ Set the hiToolTip value to an empty string ("") or nil.

For example:

```
myForm->myStringField->hiToolTip=""
```

or

```
myForm->myStringField->hiToolTip=list("tooltip1" tooltip2" "" "tooltip3")
```

Related Topics

[Creating Context Menus for Fields](#)

[Traversing Fields](#)

Form and Field Functions

This topic provides a list of Cadence® SKILL functions associated with customization of forms and fields.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [ddHiCreateCellComboField](#)
- [ddHiCreateLibraryComboField](#)
- [ddHiCreateViewComboField](#)
- [ddoAccepts](#)
- [ddoPreferred](#)
- [ddHiLinkFields](#)
- [hiAddCyclicChoice](#)
- [hiChangeFormCallback](#)
- [hiChangeFormTitle](#)
- [hiCollapseTreeItem](#)
- [hiCreateAppForm](#)
- [hiCreateBBoxField](#)
- [hiCreateBooleanButton](#)
- [hiCreateButton](#)
- [hiCreateButtonBoxField](#)
- [hiTreePrependItem](#)
- [hiTreePrependItems](#)
- [hiTreeRemoveItems](#)
- [hiTreeTableDeselectItem](#)
- [hiTreeTableDeselectAllItems](#)
- [hiTreeTableDeselectAllItems](#)

Cadence User Interface SKILL Reference

Form Functions

- [hiTreeTableDeselectItems](#)
- [hiTreeTableGetExpandedItems](#)
- [hiTreeTableGetExpandedItemCount](#)
- [hiTreeTableGetItems](#)
- [hiTreeTableGetItemCount](#)
- [hiTreeTableGetSelectedItems](#)
- [hiTreeTableGetSelectedItemCount](#)
- [hiTreeTableSelectItem](#)
- [hiTreeTableSelectItems](#)
- [hiItemRemoveTree](#)
- [hiUpdateFormBlock](#)

Cadence User Interface SKILL Reference

Form Functions

ddHiCreateCellComboField

```
ddHiCreateCellComboField(  
    ?name s_fieldName  
    [ ?prompt t_fieldPrompt ]  
    [ ?value t_currentValue ]  
    [ ?defValue t_defaultValue ]  
    [ ?callback g_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
    [ ?toolTip t_toolTip ]  
    [ ?validationType g_validationType ]  
    [ ?highlightType g_highlightType ]  
)  
=> r_comboField
```

Description

Creates a customized Cell combo field, which is an extension of a regular combo field, for selecting cells.

Arguments

?name s_fieldName The handle to this field within a form. To reference it, use `formHandle-><fieldName>`

?prompt t_fieldPrompt
A string specifying the prompt for the field. To reference it, use `formHandle-><fieldName>->prompt`

?value t_currentValue
The initial and current values of this field. To reference it, use `formHandle-><fieldName>->value`

?defValue t_defaultValue
The default value of the field. This value is equal to the initial entry in *t_currentValue*, if *t_defaultValue* is not specified. To reference the default value, use `formHandle-><fieldName>->defValue`

?callback g_callback

Cadence User Interface SKILL Reference

Form Functions

A string, symbol, or function object type callback that is executed if the value of the field changes. If a new value is typed in, the callback is called when the field has “lost focus”. This happens for example, when you move the pointer outside the form, or tab to another field, or when you select any button in the form.

- If a new value is selected from the drop-down list, then the callback is called immediately, not when the field loses focus.
- If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that get executed when the value of the field changes.

`?editable g_editable`

If set to `nil`, the field is read-only; setting `g_editable` to `t` makes the field editable.

`?enabled g_enabled`

Specifies whether the field is enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the field is invisible. The default value is `nil`.

`?nextField g_nextField`

Symbol or string indicating the next field to bring into focus when the *Tab* key is pressed. If you specify `nil` or do not specify this argument, the Tab traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout—left-to-right and top-to-bottom—beginning with the field at the top-left corner of the form.

`?toolTip t_toolTip`

The tool tip that is displayed when the pointer hovers over the field.

`?validationType g_validationType`

Cadence User Interface SKILL Reference

Form Functions

Specifies the criteria that determines if the input is valid or not. Acceptable values (symbol) are:

- `allowAny`: No restriction on input type.
- `allowExisting` (default): Allow existing unfiltered data returned by (DDPI) generator.
- `allowNew`: Allow data other than the return value of (DDPI) generator.

`?highlightType g_highlightType`

Specifies when and how a widget is to be highlighted. Acceptable values (symbol) are:

- `background`
- `highlight`
- `error`
- `highlightInvalidText` (default)

The first three values have the same definitions as given for the `g_highlightType` argument of `hiHighlightField`.

Setting the property to `highlightInvalidText` turns on automatic highlighting, which highlights the widget when an invalid input is entered. The validity of the input is determined based on the setting of the `validationType` property.

Value Returned

`r_comboField` The cell combo field.

Examples

A Cell combo field is created.

```
trCreateBlockCellName = ddHiCreateCellComboField(  
    ?name 'cellName  
    ?prompt "Cells"  
    ?defValue ""  
    ?value ""  
    ?callback "trCreateBlockCellNameCB( trCreateBlockForm 'cellName )"  
)
```


Cadence User Interface SKILL Reference

Form Functions

Related Topic

Form and Field Functions

ddHiLinkFields

ddHiCreateLibraryComboField

```
ddHiCreateLibraryComboField(  
    ?name s_fieldName  
    [ ?prompt t_fieldPrompt ]  
    [ ?value t_currentValue ]  
    [ ?defValue t_defaultValue ]  
    [ ?callback g_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
    [ ?toolTip t_toolTip ]  
    [ ?validationType g_validationType ]  
    [ ?highlightType g_highlightType ]  
)  
=> r_comboField
```

Description

Creates a customized Library combo field, which is an extension of a regular combo field, for selecting libraries.

Arguments

?name s_fieldName The handle to this field within a form. To reference it, use `formHandle-><fieldName>`

?prompt t_fieldPrompt
A string specifying the prompt for the field. To reference it, use `formHandle-><fieldName>->prompt`

?value t_currentValue
The initial and current values of this field. To reference it, use `formHandle-><fieldName>->value`

?defValue t_defaultValue
The default value of the field. This value is equal to the initial entry in *t_currentValue*, if *t_defaultValue* is not specified. To reference the default value, use `formHandle-><fieldName>->defValue`

?callback g_callback

Cadence User Interface SKILL Reference

Form Functions

A string, symbol, or function object type callback that is executed if the value of the field changes. If a new value is typed in, the callback is called when the field has “lost focus”. This happens for example, when you move the pointer outside the form, or the tab to another field, or when you select any button in the form.

- If a new value is selected from the drop-down list, then the callback is called immediately, not when the field loses focus.
- If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that get executed when the value of the field changes.

`?editable g_editable`

Enables to edit the field.

`?enabled g_enabled`

Specifies whether the field is enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the field is invisible. The default value is `nil`.

`?nextField g_nextField`

A symbol or string indicating the next field to bring into focus when the *Tab* key is pressed. If you specify `nil` or do not specify this argument, the Tab traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout —left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

`?toolTip t_toolTip`

The tool tip that is displayed when the pointer hovers over the field.

`?validationType g_validationType`

Cadence User Interface SKILL Reference

Form Functions

Specifies the criteria that determines if the input is valid or not. Acceptable values (symbol) are:

- `allowAny`: No restriction on input type.
- `allowExisting` (default): Allow existing unfiltered data returned by (DDPI) generator.
- `allowNew`: Allow data other than the return value of (DDPI) generator.

`?highlightType g_highlightType`

Specifies when and how a widget is to be highlighted. Acceptable values (symbol) are:

- `background`
- `highlight`
- `error`
- `highlightInvalidText` (default)

The first three values have the same definitions as given for the `g_highlightType` argument of `hiHighlightField`.

Setting the property to `highlightInvalidText` turns on automatic highlighting, which highlights the widget when an invalid input is entered. The validity of the input is determined based on the setting of the `validationType` property.

Values Returned

`r_comboField` Returns the combo field.

Examples

A Library combo field is created.

```
trCreateBlockLibName = ddHiCreateLibraryComboField(  
    ?name 'libraryName  
    ?prompt "Library"  
    ?value "blockLibName"  
    ?callback "trCreateBlockLibNameCB( trCreateBlockForm 'libraryName )"  
)
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[ddHiLinkFields](#)

ddHiCreateViewComboField

```
ddHiCreateViewComboField(  
    ?name s_fieldName  
    [ ?prompt t_fieldPrompt ]  
    [ ?value t_currentValue ]  
    [ ?defValue t_defaultValue ]  
    [ ?callback g_callback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
    [ ?toolTip t_toolTip ]  
    [ ?viewTypes l_viewTypes ]  
    [ ?validationType g_validationType ]  
    [ ?highlightType g_highlightType ]  
)  
=> r_comboField
```

Description

Creates a customized View combo field, which is an extension of a regular combo field, for selecting views.

Arguments

<code>?name <i>s_fieldName</i></code>	A handle to this field within a form. To reference it, use <code>formHandle-><fieldName></code>
<code>?prompt <i>t_fieldPrompt</i></code>	A string specifying the prompt for the field. To reference it, use <code>formHandle-><fieldName>->prompt</code>
<code>?value <i>t_currentValue</i></code>	The initial and current values of this field. To reference it, use <code>formHandle-><fieldName>->value</code>
<code>?defValue <i>t_defaultValue</i></code>	The default value of the field. This value is equal to the initial entry in <i>t_currentValue</i> , if <i>t_defaultValue</i> is not specified. To reference the default value, use <code>formHandle-><fieldName>->defValue</code>
<code>?callback <i>g_callback</i></code>	

Cadence User Interface SKILL Reference

Form Functions

A string, symbol, or function object type callback that is executed if the value of the field changes. If a new value is typed in, the callback is called when the field has “lost focus”. This happens for example, when you move the pointer outside the form, or tab to another field, or when you select any button in the form.

- If a new value is selected from the drop-down list, then the callback is called immediately, not when the field loses focus.
- If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that get executed when the value of the field changes.

`?editable g_editable`

Edits the field.

`?enabled g_enabled`

Specifies whether the field is enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the field is invisible. The default value is `nil`.

`?nextField g_nextField`

A symbol or string indicating the next field to bring into focus when the *Tab* key is pressed. If you specify `nil` or do not specify this argument, the Tab traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout —left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

`?toolTip t_toolTip`

The tool tip that is displayed when the pointer hovers over the field.

`?viewTypes l_viewTypes`

Cadence User Interface SKILL Reference

Form Functions

A list of DFII view types (string). The list is used for view-type based filtering in the view combo field. A view is made available for selection only if the view type associated with its master file is a member of the *l_viewTypes* list.

?validationType *g_validationType*

Specifies the criteria that determines if the input is valid or not. Acceptable values (symbol) are:

- *allowAny*: No restriction on input type.
- *allowExisting* (default): Allow existing unfiltered data returned by (DDPI) generator.
- *allowNew*: Allow data other than the return value of (DDPI) generator.

?highlightType *g_highlightType*

Specifies when and how a widget is to be highlighted. Acceptable values (symbol) are:

- *background*
- *highlight*
- *error*
- *highlightInvalidText* (default)

The first three values have the same definitions as given for the *g_highlightType* argument of *hiHighlightField*.

Setting the property to *highlightInvalidText* turns on automatic highlighting, which highlights the widget when an invalid input is entered. The validity of the input is determined based on the setting of the *validationType* property.

Values Returned

r_comboField Returns the combo field.

Examples

A View combo field is created.

Cadence User Interface SKILL Reference

Form Functions

```
trCreateBlockViewName = ddHiCreateViewComboField(  
    ?name 'viewName  
    ?prompt "View"  
    ?defValue "symbol"  
    ?value "symbol"  
    ?callback "trCreateBlockViewNameCB ( trCreateBlockForm 'viewName )"  
)
```

Additional Information

On the *Library*, *Cell*, and *View* combo boxes, you have a filter property that enables you to customize your search. You need to specify the value of the filter property as a list, where the first member denotes the method for filtering.

The following filtering methods are supported:

- **'wildCard:** Specifies the value of the filter property as a list, where the first member is 'wildCard. For example, (list 'wildCard "Mos*")
- **'regExp:** Specifies the value of the filter property as a list, where the first member is 'regExp. For example, (list 'regExp "^[0-9]_")
- **'viewTypes:** Specifies the value of the filter property as a list, where the first member is 'viewType. This method can only be applied on the *View* combo box. For example, (list 'viewType "schematicSymbol" "schematic")
- **'custom:** Specifies a custom filter by assigning the filter property on *Library*, *Cell*, and *View* combo boxes. For example, cell_combo_box->filter=(list 'custom callable_or_filterInstance, where the first member is 'custom and the second member callable_or_filterInstance is either a SKILL callable or an instance of ddHiObjFilter.

In a custom filter, when the second member is a SKILL function (callable), the function is used as the `accepts` method. It is called with an argument of `ddId` type. A `dd` object becomes available for the selection in the combo box only if the function returns a non-nil value.

If the function returns `nil` on a specified `dd` object, the object is considered as non-existent. If you assign `nil` to the filter property, filtering gets turned off on the LCV combo box.

When a SKILL function (callable) is given as a custom filter on a *View* combo box, a default `preferred` method is supplied. It always returns the first member of a specified `dd` objects list.

If an error occurs in a SKILL callable, the given `dd` object is regarded as accepted. A maximum of three invocations are made if an error occurs in the SKILL callable.

Cadence User Interface SKILL Reference

Form Functions

Alternatively, the second member can be an instance of `ddHiObjFilter`, or an instance of a subclass of `ddHiObjFilter`. The `ddHiObjFilter` class defines two override methods, `ddoAccepts` and `ddoPreferred`.

Both `accepts` and `preferred` methods must be prefixed with `ddo`, where `o` indicates override.

If an error occurs in the `ddoAccepts` method, the specified `dd` object is considered as accepted. Whereas, if an error occurs in the `ddoPreferred` method, the first `dd` object in the specified list is used.

A maximum of three invocations are made if an error occurs in any of these methods.

Related Topics

[Form and Field Functions](#)

[ddoAccepts](#)

[ddoPreferred](#)

[ddHiLinkFields](#)

Cadence User Interface SKILL Reference

Form Functions

ddoAccepts

```
ddoAccepts(  
    g_filter  
    g_obj  
)  
=> t / nil
```

Description

This function is a method of `ddHiObjFilter` that is called with a specified `dd` object argument to get listed in the associated combo box.

Arguments

<i>g_filter</i>	A <code>ddHiObjFilter</code> object.
<i>g_obj</i>	A <code>dd</code> object.

Values Returned

<i>t</i>	The specified object is accepted for selection in the associated combo box.
<i>nil</i>	The specified <code>dd</code> object is accepted and listed in the associated combo box.

Examples

Defines a `ddHiObjFilter` sub-class, and its related methods. The `ddoAccepts` function accepts view names, which are member of `trAcceptedViewNames`.

```
(define trAcceptedViewNames (list "view1" "view2" ...))  
  
(defclass trViewFilter (ddHiObjFilter) ())  
  
(defmethod ddoAccepts ((self trViewFilter) (obj t))  
    (member obj->name trAcceptedViewNames))
```

Related Topics

Form and Field Functions

ddoPreferred

```
ddoPreferred(  
    g_filter  
    l_obj  
)  
  
=> b_obj / nil
```

Description

This function is a method of `ddHiObjFilter` that returns the preferred `dd` view object only for the *View* combo box. This return value is called after a linked cell ID is changed, and the value of the *View* combo box is not valid. The function is called with a list of `dd` object IDs and is expected to return a valid `dd-view-id` or `nil`.

Arguments

<i>g_filter</i>	A <code>ddHiObjFilter</code> object.
<i>l_obj</i>	A list of <code>dd</code> objects.

Values Returned

<i>b_obj</i>	A preferred <code>dd</code> view object.
<code>nil</code>	The first <code>dd</code> object in the specified list is accepted and listed in the <i>View</i> combo box.

Examples

Defines a sub-class of `ddHiObjFilter`. It returns the second object of a given list if available, otherwise returns the first object in the list.

```
(define trAcceptedViewNames (list "view1" "view2" ...))  
  
(defclass trViewFilter (ddHiObjFilter) ())  
  
(defmethod ddoPreferred ((self trViewFilter) (objs list))  
    "preferred 2nd obj over the 1st one."  
    (or (cadr objs) (car objs)))
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

ddHiLinkFields

```
ddHiLinkFields(  
    o_libraryComboBox  
    o_cellCombobox  
)  
=> t / nil  
  
ddHiLinkFields(  
    o_cellCombobox  
    o_viewComboBox  
)  
=> t / nil  
  
ddHiLinkFields(  
    o_libraryComboBox  
    o_cellCombobox  
    o_viewComboBox  
)  
=> t / nil
```

Description

Connects the given combo boxes, where the value changes in one combo box triggers re-populating of the next combo box.

Note: To connect the library, cell, and view combo fields in a form, the form must be instantiated. In addition, to connect the library, cell, and view combo box items in a toolbar, the toolbar must be placed in a window.

Arguments

<i>o_libraryComboBox</i>	The library combo box item or field.
<i>o_cellComboBox</i>	The cell combo box item or field.
<i>o_viewComboBox</i>	The view combo box item or field.

Values Returned

<i>t</i>	The combo box widgets are linked successfully.
<i>nil</i>	The combo box widgets are not linked successfully.

Cadence User Interface SKILL Reference

Form Functions

Examples

The library, cell, and view combo box fields are linked successfully.

```
hiInstantiateForm( trForm )
ddHiLinkFields( trForm->trLibrary trForm->trCell trForm->trView )
=> t
```

This example shows how to open a form and toolbar which uses `ddHiLinkFields`:

```
(toplevel 'ils)
;; Create lcv widget
(defun make_lcv_widget ( widget_type purpose @key name callback viewTypes "ssggl")
  "make a lcv widget of s_widget_type (ComboField or ToolbarComboBox) for
s_purpose (Library, Cell, or View). Return a combo-box defstruct."
  (unless (symbolp name) name=(gensym purpose))
  (when (and (memq widget_type '(ComboField ToolbarComboBox))
            (memq purpose '(Library Cell View)))
    (let ((fun (concat 'ddHiCreate purpose widget_type))
          (args (list '?name name '?prompt (strcat " " (get_pname name) " ")
                      '?callback callback)))
      (when (and (eq purpose 'View) viewTypes)
        args= (append args (list '?viewTypes viewTypes)))
      (apply fun args))))

;; use l_viewTypes to specify a view-type based filter on the view field
(defun make_lcv_form ( name @optional viewTypes "sl")
  (let ((form (hiCreateLayoutForm
                name (get_pname name)
                (hiCreateVerticalBoxLayout
                 'vboxlayout ?items
                 (list (make_lcv_widget 'ComboField 'Library ?name 'library)
                       (make_lcv_widget 'ComboField 'Cell ?name 'cell)
                       (make_lcv_widget 'ComboField 'View ?name 'view
                                       ?viewTypes viewTypes))))))
    (define (link_fields_)
      (ddHiLinkFields form->library form->cell form->view))
    (list form 'link_fields link_fields_)))

;; use l_viewTypes to specify a view-type based filter on the view combo
(defun make_lcv_toolbar ( @key name viewTypes "sl")
  (let ((toolbar
```

Cadence User Interface SKILL Reference

Form Functions

```
(hiCreateToolbar
  ?name name ?toolTip (strcat "window(#)->hiToolbars->" name)
  ?items (list (make_lcv_widget 'ToolbarComboBox 'Library ?name 'library)
               (make_lcv_widget 'ToolbarComboBox 'Cell ?name 'cell)
               (make_lcv_widget 'ToolbarComboBox 'View ?name 'view
                               ?viewTypes viewTypes))))

(define (link_items_)
  (ddHiLinkFields toolbar->library toolbar->cell toolbar->view))

(list toolbar 'link_fields link_items_))

;; Displaying the form
form=(make_lcv_form 'myform)
(hiDisplayForm (car form))
form->link_fields()

;; Placing the toolbar on a window
myToolbar=(make_lcv_toolbar ?name 'test)
myWindow=(hiOpenWindow)
(hiPlaceToolbar myWindow car(myToolbar))
myToolbar->link_fields()
```

Related Topics

Form and Field Functions

hiAddCyclicChoice

```
hiAddCyclicChoice(  
    r_form  
    s_cyclicField  
    g_newChoice  
)  
=> t / nil
```

Description

Adds the new entry to the end of the list of choices in the cyclic field in the form.

To change the entire selection of a cyclic field, use

```
form->cyclicField->choices = newchoicelist
```

where *newchoicelist* is a list of strings and/or icons for this field. (See the description of `hiCreateCyclicField` for information on how to specify the list.)

Arguments

<i>r_form</i>	Form returned from <code>hiCreateAppForm</code> or <code>hiCreateForm</code> .
<i>s_cyclicField</i>	Must be a valid cyclic field in the specified form, and should be specified by its symbol handle within the form. If <i>s_cyclicField</i> is a shared field, <i>g_newChoice</i> is added to all occurrences of that field.
<i>g_newChoice</i>	The new entry to be added to the <i>cyclicField</i> contained in <i>r_form</i> . <i>g_newChoice</i> must either be text or an icon.

Values Returned

<i>t</i>	The new item is added.
<i>nil</i>	The new item is not added. An error message is returned.

Related Topics

[hiCreateCyclicField](#)

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateForm

Form and Field Functions

hiChangeFormCallback

```
hiChangeFormCallback(  
    r_form  
    g_newCallback  
)  
=> t
```

Description

Changes the `doneAction` and/or the `cancelAction` of the specified form to the SKILL callback procedure defined by *newCallback*. *newCallback* can be a symbol, a string, a function object, or a list of two symbols/strings/function objects. The first callback is taken as the `doneAction`. The second callback is taken as the `cancelAction`. After this routine is called, if the *OK* or *Apply* button of the form is pressed, the `doneAction` is executed, and if the *Cancel* button is pressed, the `cancelAction` is executed.

Arguments

<i>r_form</i>	Form whose action is being changed.
<i>g_newCallback</i>	The <code>doneAction</code> and/or <code>cancelAction</code> to be executed. The callback type can be a symbol, a string, a function object, or a list. If a list is given, the items in the list can be of symbol, string, or function object type.

Related Topics

[Form and Field Functions](#)

hiChangeFormTitle

```
hiChangeFormTitle(  
    r_form  
    t_newTitle  
)  
=> t / nil
```

Description

Changes the window manager title of the specified form to *t_newTitle*.

Arguments

<i>r_form</i>	Form whose title is changed.
<i>t_newTitle</i>	New title to appear in the window manager frame. This frame is placed around each window depending on the window manager being used.

Values Returned

<i>t</i>	The window manager title is changed.
<i>nil</i>	The window manager title is not changed.

Related Topics

[Form and Field Functions](#)

hiCollapseTreeItem

```
hiCollapseTreeItem(  
    g_item  
    [ g_notify ]  
)  
=> t / nil
```

Description

Collapses the tree item you specify. This function also executes the collapse callback of the tree table field if you specify *t* as the value of the *g_notify* argument.

This function is the programmatic equivalent of collapsing an item with the mouse or keyboard keys.

Arguments

<i>g_item</i>	The item you want to collapse. <i>g_item</i> can either be the return value of <code>hiCreateTreeItem</code> or a symbol specifying the item name.
<i>g_notify</i>	Runs the collapse callback of the tree table field.

Values Returned

<i>t</i>	The item was collapsed.
<i>nil</i>	The item does not have a tree under it, the item was not expanded, or there was an error.

Related Topics

[Form and Field Functions](#)

[hiExpandTreeItem](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateAppForm

```
hiCreateAppForm(  
    ?name s_name  
    ?fields l_fieldEntries  
    [ ?attachmentList l_fieldAttachments ]  
    [ ?tabOrderIsAddOrder g_tabOrderIsAddOrder ]  
    [ ?formTitle t_formTitle ]  
    [ ?callback g_callback ]  
    [ ?unmapAfterCB g_unmapAfterCB ]  
    [ ?mapCB g_mapCallback ]  
    [ ?formType s_formType ]  
    [ ?dialogStyle s_dialogStyle ]  
    [ ?buttonLayout g_buttonLayout ]  
    [ ?buttonDisabled s_buttonDisabled ]  
    [ ?help g_help ]  
    [ ?initialSize g_initialSize ]  
    [ ?minSize g_minSize ]  
    [ ?maxSize g_maxSize ]  
    [ ?dontBlock g_dontBlock ]  
    [ ?fieldFocus s_fieldFocus ]  
    [ ?plist g_plist ]  
    [ ?baseName s_baseName ]  
)  
=> r_form
```

Description

Returns the SKILL representation of a form with the specified field entries, setting it to the specified name. All standard forms, from simple forms to property lists, are generated using `hiCreateAppForm`, with varying field entries. Use this function instead of `hiCreateForm`.

Note: Do not use any predefined property names (of HI forms, tab fields, tab pages, or scroll regions) as field names on your forms. Using the predefined names results in an error. To avoid the error, it is recommended that you either use a different field name or add a non-Cadence style prefix to the field name. Cadence adds prefixes that begin with an underscore ("_") or a lowercase letter to field names as a standard style.

Arguments

<code>?name <i>s_name</i></code>	Global SKILL symbol used to reference this form. Do not use the same symbol to reference different forms.
<code>?fields <i>l_fieldEntries</i></code>	

Cadence User Interface SKILL Reference

Form Functions

List of field descriptors returned from the field creation routines. You must use at least one field entry. There is no upper limit on the number of field entries you specify.

By default, the fields appear in a column on the form, in the order in which they are listed. To manipulate the order and position of form fields, use two-dimensional attributes. If you use two-dimensional attributes, each field entry for *l_fieldEntries* is a list containing a field descriptor and the two-dimensional attributes for that field. For example, a field entry could be:

```
list (buttonField x:y width:height)
```

For details about two-dimensional attributes, see [Two Dimensional Form Layout](#).

If you are creating a form that supports field attachments (so that fields are automatically resized when you resize the form), you must specify two-dimensional attributes for each field entry in *l_fieldEntries*.

Cadence User Interface SKILL Reference

Form Functions

?attachmentList *l_fieldAttachments*

List of field attachments. Field attachments determine how the two-dimensional attributes for a field are used to position that field in the form. The first field attachment is applied to the first field in *l_fieldEntries*, the second to the second field, and so forth. You must have the same number of attachments in *l_fieldAttachments* as the number of fields in *l_fieldEntries*. If you do not want to specify a field attachment for a field, use `nil` or 0.

Each field attachment in *l_fieldAttachments* consists of a bitwise OR of a list of constants. Each constant is a specification for a side of the field (left, right, top, and bottom). These attachment specifications for the sides of a field can be positional or percentage attachments.

For example, an *l_fieldAttachments* list could be:

```
(constant | constant | constant | constant  
constant | constant nil constant | constant |  
constant)
```

For details about field attachments and how to use positional and percentage constants, see [Field Attachments](#).

If you use the field attachments argument, you must specify the following:

- The *g_initialSize* argument as a list of width and height.
- Two-dimensional attributes for each field entry in the *l_fieldEntries* argument.

Cadence User Interface SKILL Reference

Form Functions

?tabOrderIsAddOrder *g_tabOrderIsAddOrder*

t or *nil*. Determines the order in which fields are traversed when the Tab key is pressed. If you specify *t*, the *nextField* property is automatically set for the fields and the traversal order is the order in which fields are added to the form (that is, the order in which they are listed in the *l_fieldEntries* list). If you specify *nil*, the traversal order is based on the form layout —left-to-right and top-to-bottom— beginning with the field in the top left corner of the form. The default value of *g_tabOrderIsAddOrder* is *nil*.

The *nextField* property set on a field (to any non-*nil* value) overrides the *g_tabOrderIsAddOrder* value.

?formTitle *t_formTitle*

Name that appears in the window manager banner of the form.

Cadence User Interface SKILL Reference

Form Functions

?callback *g_callback*

Used to specify the SKILL functions (callbacks) that execute after you select Apply, OK, or Cancel in the form.

g_callback can be in one of two formats:

g_okAction

or

'(*g_okAction g_cancelAction*)

g_okAction specifies the SKILL function to be executed when the *Apply*, *OK*, *Close*, or *Quit* button is pressed in a form; *g_cancelAction* specifies the SKILL function to be executed when the Cancel button is pressed. Here, if *g_cancelAction* is not specified, no callback gets called when the form is canceled.

Both *g_okAction* and *g_cancelAction* can be in one of the following formats:

t_callback, *s_callback*, or *u_callback*

t_callback is the string representation of the SKILL function to be executed. If you specify *t_callback*, the string is evaluated and the callback function is not passed any argument. *s_callback* is the symbol of the SKILL function to be called. If you specify *s_callback*, the callback function is passed and the *r_form* returned from `hiCreateLayoutForm` as its only parameter. *u_callback* is a callback of function object type. The callback function is passed and *r_form* returned from `hiCreateLayoutForm` as its only parameter.

After the form is instantiated, you can change the callback with the `hiChangeFormCallback` function.

?unmapAfterCB *g_unmapAfterCB*

If set to `t`, the form is removed (unmapped) from the screen after the user's callback is executed. If set to `nil`, the form is removed before the callback is invoked. The default value is `nil`. If you use `hiSetCallbackStatus` in your callback routine, you must set the value of *g_unmapAfterCB* to `t` or the callback status is ignored.

?mapCB *g_mapCallback*

Cadence User Interface SKILL Reference

Form Functions

Specifies the callback to be invoked when the form is mapped, that is, when `hiDisplayForm` is called for a form that is not currently mapped. Specify either the name (string), symbol of the map callback function, or the function object.

?formType *s_formType*

Possible choices include `'nonoptions` (default), or `'options`. Options forms must be modeless.

?dialogStyle *s_dialogStyle*

One of the following symbols, which determines the blocking behavior of the form:

`'modal`

Blocks access to the application while it is displayed. The user must dismiss the form in order to continue using the application. (Access to window manager commands or other applications is not blocked.)

`'modeless`

Does not block access to the application. The user can continue using the application without having to close the form.

The default value of this argument is `'modeless`.

Cadence User Interface SKILL Reference

Form Functions

?buttonLayout *g_buttonLayout*

Used to specify the buttons that appear at the bottom of the form. *g_buttonLayout* can be in one of the following formats:

■ *s_buttonLayout*

Displays a predefined set of buttons, followed by the Help button. You can select from a list of button layouts.

■ '(s_buttonLayout (s_customButtonText
s_customButtonCB) ...)

Displays a predefined set of buttons, followed by your custom buttons and the Help button. Custom buttons are specified as button text/button callback pairs.

■>('Empty (s_customButtonText
s_customButtonCB) ...)

Displays only your custom buttons, followed by the *Help* button. Custom buttons are specified as button text/button callback pairs.

Specify either *OK*, *Close*, or *Quit* as one of the button names in your list of custom buttons so the form can be closed. You can use `hiFormClose()` as the callback for these buttons.

■ 'Empty

Does not display any buttons in the form.

The form can be closed either through the window manager or by calling `hiFormDone()`.

s_buttonLayout specifies a predefined set of buttons. The Help button is automatically added with each of these combinations. Select one of the following button layouts:

Cadence User Interface SKILL Reference

Form Functions

For non-options forms:

- 'OKCancel
- 'OKCancelDef
- 'OKCancelLast
- 'OKCancelDefLast
- 'OKCancelApply
- 'ApplyCancel
- 'ApplyCancelDef
- 'ApplyCancelLast
- 'ApplyCancelDefLast
- 'Close/'Quit
- 'OKCancelDefApply (default)

For options forms:

- 'HideCancel
- 'HideCancelDef
- 'HideCancelLast
- 'HideCancelDefLast (default)
- 'ApplyHideCancel
- 'ApplyHideCancelDef
- 'ApplyHideCancelLast
- 'ApplyHideCancelDefLast

where:

OK denotes the *OK* button (`hiFormDone` is called when this button is pressed, and `hiFormDone` calls the `g_okAction` callback);

Apply denotes the *Apply* button (`hiFormApply` is called when this button is pressed, and `hiFormApply` calls the `g_okAction` callback);

Cadence User Interface SKILL Reference

Form Functions

`Cancel` denotes the *Cancel* button (`hiFormCancel` is called when this button is pressed, and `hiFormCancel` calls the `g_cancelAction` callback);

`Close` denotes the *Close* or *Quit* button (`hiFormClose` is called when this button is pressed, and `hiFormClose` calls the `g_okAction` callback. Here the *Close* and *Quit* buttons are similar to the *OK* button; `hiFormClose` is an alias for `hiFormDone`);

`Last` denotes the *Last* button;

`Def` denotes the *Defaults* button;

`Hide` denotes the *Hide* button.

Note: If there is no *Cancel* button on a form but there is a *Close* or *Quit* button, when you close the form from the window manager or by typing `hiFormCancel` in the CIW, `hiFormDone` is called.

For custom buttons, following `s_buttonLayout` or `'Empty` should be lists consisting of button text/button callback pairs:

```
'(s_customButtonText s_customButtonCB)
```

`s_customButtonText` is the symbol whose printname is the button text. For example, to have a button "Press Me", `s_customButtonText` would be specified as `'Press\ Me`.



Tip

Tip: You can use `concat("string")` to get the symbol of a string.

`s_customButtonCB` is the callback for the button and can be specified as a symbol or string. If it is a symbol, the callback function is passed and `r_form` returns from `hiCreateAppForm` as its only argument. If it is a string, the unchanged string is passed to SKILL to interpret.

Pressing Enter, `<Enter>`, performs the same action as the first button of the form button list. This is usually `hiFormDone()` for non-options forms and `hiToggleEnterForm()` for options forms. If you specify your own button list, you need to be aware that the action of `<Enter>` can change. For example, if your button list is

Cadence User Interface SKILL Reference

Form Functions

'ApplyCancelDef, the action of <Enter> in a type-in field performs the `hiFormApply()` action.

You can also customize this feature by specifying the button that you want to link to Enter. You do this setting the following environment variable in your `.cdsenv` file:

```
ui formDefaultAction string "value"
```

where `value` is the `buttonText` of the button that you want to link to <Enter>.

When this variable is set, the `g_buttonLayout` argument is searched for the button whose text equals the value of the variable. The first button that is found becomes the default button.

If the specified value is not found in `g_buttonLayout`, then <Enter> performs the same action as the first button of the form button list.

```
?buttonDisabled s_buttonDisabled
```

List of buttons that are disabled when the form appears. You can change this value with the `hiSetFormButtonEnabled` function.

Cadence User Interface SKILL Reference

Form Functions

?help *g_help*

Specify *g_help* as a single string, single symbol, list of strings, or list of symbols that is to be used to reference a `.tgf help` file. (If you do not want to use the default `.tgf help` file mechanism, you can specify your own callback for the Help button; see [Specify Your Own Callback for Help](#).)

If you specify a single string or symbol, it is used to set the `hiHelpAppName` property on the symbol that represents the form name. If you specify a list of strings or a list of symbols, the first element in the list is used to set the `hiHelpAppName` property and the second element is used to set the `hiHelpSymName` property.

(If you want to specify the help symbol only, specify a list of two strings or symbols, with `nil` as the first element of the list and the help symbol as the second element. This sets the `hiHelpSymName` property without setting the `hiHelpAppName` property.)

When a user clicks Help on the form, the `hiHelp` function is called with the following arguments:

`appId helpTag`

If the `hiHelpAppName` property has been set, it is used as the `appId`, otherwise the application ID of the current window is used as the `appId`. If the `hiHelpSymName` property has been set, it is used as the `helpTag`, otherwise the form name is used as the `helpTag`.

The `appId` and `helpTag` determine the name of the tag file that is called (`appId.tgf`) and the help tag in the file that is accessed.

You can also use the `hiHelpAppName` and `hiHelpSymName` properties directly to specify the help file and help tag that are used to reference the help.

Cadence User Interface SKILL Reference

Form Functions

?initialSize *g_initialSize*

The size of the form when it first appears. *g_initialSize* accepts the following values:

- A list of width and height. The form is sized to the specified width and height or to the size needed to accommodate all the fields, whichever is less.
- A symbol representing a field in the form. The form is sized to include the field but exclude any other fields to the right of and below the specified field.
- `t`. The form is sized to fit all fields, or to the size of the screen, whichever is smaller.

If you are creating a form that supports field attachments (so that fields are automatically resized when you resize the form), you must specify the *g_initialSize* argument and it must be a list of width and height.

For forms that have field attachments, the initial size value is used only for the field area and excludes the button area. For forms that do not have field attachments, the initial size is interpreted as including the button area. The area required for the scroll bar is never part of the initial size.

?minSize *g_minSize*

The minimum size of the form, specified as a list of width and height or *width:height*. The minimum size excludes the window manager's title and border. The default minimum width is a few pixels wider than the width of the button banner and the default minimum height is a few pixels more than the height of the button banner.

To change the minimum size, use the `hiSetFormMinMaxSize` function.

?maxSize *g_maxSize*

The maximum size of the form, specified as a list of width and height or *width:height*. The maximum size excludes the window manager's title and border. The default maximum size is the size of the screen.

To change the maximum size, use the `hiSetFormMinMaxSize` function.

Cadence User Interface SKILL Reference

Form Functions

`?dontBlock` *g_dontBlock*

If set to `t`, a call to `hiDisplayForm()` on this form immediately returns `t`. The default is `nil`.

If set to `nil`, the blocking form cannot be displayed before the initialization—which includes reading the `.cdsinit` file or other SKILL files—is complete. To delay displaying of the blocking form until initialization is complete, use `hiEnqueueCmd`.

Prior to the 9504 release, `hiDisplayForm` would not return `t` or `nil` immediately. The form used to be displayed until the user selected either the *OK* or *Cancel* button. If the user selected the *OK* button, it would return `t`. If the form was canceled, it would return `nil`.

`?fieldFocus` *s_fieldFocus*

Optional argument used to determine which type-in fields can accept focus through tabbing or clicking the mouse. Possible choices are `'editableTypein` (default) and `'allTypein`.

If set to `'allTypein`, all type-in fields can accept focus, whether they are editable or non-editable. If set to `'editableTypein`, only editable type-in fields can accept focus.

`?plist` *g_plist*

Indicates whether or not this is a property list form.

`?baseName` *s_baseName*

If specified, it indicates that the form name is to be used for help.

Values Returned

r_form

SKILL structure representing the form.

Examples

This example shows how to re-display a form that has already been created. It also shows how to set defaults:

```
;*****  
;Sample form and fields  
;*****
```

Cadence User Interface SKILL Reference

Form Functions

```
; Create string entry for the form.
procedure( myformDefineFields()
  let((strfld boolfld)
    strfld = hiCreateStringField(
      ?name 'strfld
      ?prompt "sample string field"      ;;; FIELD1, a string type
      ?defValue "initial string"
      ?callback "my_strfld_cb(hiGetCurrentForm())"
      ?editable t
    )
    ; Create a Boolean button for the form.
    boolfld = hiCreateBooleanButton(
      ?name 'boolfld
      ?buttonText "sample boolean field"      ;;; FIELD2, a boolean type
      ?defValue t
      ?callback "my_boolfld_cb(hiGetCurrentForm())"
    )
    list( strfld boolfld )
  )
)

; Generate the form
procedure( myformCreateForm()
  if( !boundp(`myformglb) || (myformglb == nil)
    then
      myformglb = hiCreateAppForm(
        ?name `myformglb
        ?fields myformDefineFields()
        ?formTitle "My first Form"
        ?callback 'myformglb_cb
        ?buttonLayout `OKCancel
      )
  )
  myformglb
)

procedure(myformglb_cb(form)
  printf("Field values are: %s and %L\n" form->strfld->value form->boolfld->value)
)

procedure(my_strfld_cb(form)
  printf("String field changed to %s\n" form->strfld->value)
```

Cadence User Interface SKILL Reference

Form Functions

```
)

procedure(my_boolfld_cb(form)
  printf("Boolean field changed to %L\n" form->boolfld->value)
)

; Display the form
procedure( myformDisplayForm()
  myformCreateForm()      ;;; Creation and Display are separate
  hiDisplayForm(`myformglb)
)

myformDisplayForm()
```

- Invoking `myformdisplayform()` displays the above form. If you dismiss the form, the form disappears but the object still remains.
- To redisplay the form use `hiDisplayForm('myformglb)`.
- If you modify some fields on the form, you can restore the default field settings using `hiFormDefaults('myformglb)`.

Additional Information

When you create a form field, if all arguments are valid, the internal data structure for a field is returned; otherwise `nil` is returned. The following routines create fields for a form. Each of these routines returns a field descriptor which should be assigned to a user-specified symbol. These symbols are then used as arguments to the `hiCreateAppForm` routine.

The symbol each field descriptor is *assigned to* may be any arbitrary global symbol. Once these have been passed to the `hiCreateAppForm` function, it is no longer reliable to reference field data through these global symbols. You should then use the `formSymbol->fieldSymbol` notation, where `fieldSymbol` is the symbol passed as the `?name` argument to the field creation routines.

The following form field creation routines are available:

Cadence User Interface SKILL Reference

Form Functions

Fields	SKILL functions
■ Type-in fields:	<u>hiCreateStringField</u> <u>hiCreateIntField</u> <u>hiCreateFloatField</u> <u>hiCreateListField</u> <u>hiCreatePointField</u> <u>hiCreateBBoxField</u> <u>hiCreatePointListField</u>
■ Enumerated choice fields:	<u>hiCreateToggleField</u> <u>hiCreateRadioField</u> <u>hiCreateCyclicField</u> <u>hiCreateButtonBoxField</u> <u>hiCreateSpinBox</u>
■ Combination fields:	<u>hiCreateComboField</u>
■ Display Fields	<u>hiCreateOutputStringField</u> <u>hiCreateHypertextField</u> <u>hiCreateSimpleHypertextField</u> <u>hiCreateReportField</u>
■ Miscellaneous fields:	<u>hiCreateButton</u> <u>hiCreateLabel</u> <u>hiCreateLayerCyclicField</u> <u>hiCreateBooleanButton</u>
■ Scroll region fields:	<u>hiCreateScrollRegion</u>

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormApply

hiFormCancel

hiFormClose

hiFormDefaults

hiCreateBBoxField

```
hiCreateBBoxField(  
    ?name s_fieldName  
    [ prompt t_fieldPrompt ]  
    [ ?value l_currentValue ]  
    [ ?help g_fieldHelp ]  
    [ ?defValue l_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a bounding box field for a form.

A bounding box field accepts only a bounding box as input. A valid bounding box is represented by two points (each point is a list of two integers) separated by blank space. The first point is the lower-left corner of the box, the second point is the upper-right corner.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name s_fieldName Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

prompt t_fieldPrompt

Optional prompt for this field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?value l_currentValue

Cadence User Interface SKILL Reference

Form Functions

Initial and current value of this field in the correct format. This is a required parameter. May be `nil`, in which case the `bBox` field appears blank. To reference the current value, use:

```
formHandle->hiFieldSym->value
```

?help *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?defValue *l_defaultValue*

Default value of the field (in the correct format). This value is equal to the initial entry in *l_currentValue* if *l_defaultValue* is not specified. To reference the default value, use:

```
formHandle->fieldSym->defValue
```

```
formHandle->hiFieldSym->defValue
```

?font *t_font* An optional font argument is accepted but not supported at this time.

?callback *g_callback*

Callback for this field. The callback can be string, symbol, or a function object type. The callback is called when the field has lost focus (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?modifyCallback *g_modifyCallback*

Cadence User Interface SKILL Reference

Form Functions

Do not use the *g_modifyCallback* argument with this field because it may not work as expected. Use this argument only with string fields.

SKILL function to be executed whenever any change is made in the text field, for example, by typing or pasting. The modify callback is called when a change is detected in the field but before the change is displayed.

The modify callback function is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the *value* of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a bounding box. This value replaces the current value of the field.

?focusInCallback *g_focusInCallback*

SKILL function to be executed whenever the field comes into focus. You provide this function.

?editable *g_editable*

Enables the editing of the field.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

?invisible *g_invisible*

Cadence User Interface SKILL Reference

Form Functions

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

You can modify the value of this argument (after the form is created) with the `invisible` property. For example:

```
formHandle->fieldSym->invisible = nil
```

`?nextField` *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the Tab traversal order is determined by the *tabOrderIsAddOrder* argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Values Returned

r_fieldHandle Handle to the bounding box field.

Related Topics

[Form and Field Functions](#)

[hiCreateScrollRegion](#)

[hiCreateAppForm](#)

[Creating Context Menus for Fields](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateBooleanButton

```
hiCreateBooleanButton(  
    ?name s_fieldName  
    ?buttonText t_buttonText  
    [ ?buttonLocation s_buttonLocation ]  
    [ ?callback g_callback ]  
    [ ?value g_booleanValue ]  
    [ ?defValue g_defaultValue ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?partialSelect g_partialSelect ]  
)  
=> r_fieldHandle
```

Description

Creates a single Boolean button for a form. The button text appears as a prompt next to the button; the button is on the left of the button text if the `?buttonLocation` argument is 'left' and on the right of the button text if the `?buttonLocation` argument is 'right'. A Boolean field consists of a single toggle button which may have an associated callback procedure executed when the Boolean state changes.

For efficiency, Boolean buttons should always be used in preference to single toggle buttons.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

`?name s_fieldName` Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

`?buttonText t_buttonText`

Prompt string which appears next to the toggle button.

`?buttonLocation s_buttonLocation`

Cadence User Interface SKILL Reference

Form Functions

`'left` or `'right`

Default value: `'right`

Specify `'left` if you want the button to appear on the left of the button text; specify `'right` if you want the button to appear on the right of the button text.

If the prompt box width is not specified and the button is on the right, the button is right justified in the field. If the prompt box width is not specified and the button is on the left, the button text on the right is not right justified; instead, it is placed as if the prompt box width were set to 20. (The prompt box width is a two-dimensional attribute you can specify when you create the form.)

In one-dimensional forms, the button text is always on the left; the `?buttonLocation` argument is ignored.

`?callback` *g_callback*

A string, function object, or symbol type callback. The callback is called as soon as the button state changes.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

When the callback is either symbol type or string type, it only passes one argument, `t` or `nil`.

In case the callback is string type and contains parentheses, such as `"MyCallback()"` then no argument is passed (other than what is specified in the string, if any)

If the callback is object type, then the syntax used is `<Boolean button callback> (o_field g_scope g_value)`

`?value` *g_booleanValue*

Specifies the value of a boolean button.

The valid values are: `t`, `nil`, or `partial`. The default value is `nil`.

The `partial` value can be specified only when `?partialSelect` is set to `t`.

`?defValue` *g_defaultValue*

Cadence User Interface SKILL Reference

Form Functions

Optional default value of the field. This value is equal to the initial entry in *currentValue* if *defaultValue* is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

`?help g_fieldHelp` A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

`?font t_font` A font argument is accepted but not supported at this time.

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?partialSelect g_partialSelect`

Enables a new visual state of a boolean button by providing indeterminate checks to the checkboxes. The default value is `nil`.

Values Returned

`r_fieldHandle` Handle to the Boolean button.

Examples

```
;*****
;Sample form and fields
;*****

; Create string entry for the form.
procedure( myformDefineFields()
  strfldglb = hiCreateStringField(
    ?name `strfldglb
    ?prompt "sample string field"    ;; FIELD1, a string type
    ?defValue "initial string"
    ?callback "my_strfldglb_cb()"
    ?editable t
```

Cadence User Interface SKILL Reference

Form Functions

```
)

; Create a Boolean button for the form.
boolfldglb = hiCreateBooleanButton(
    ?name `boolfldglb
    ?buttonText "sample boolean field"          ;;; FIELD2, a boolean type
    ?defValue t
    ?callback "my_boolfldglb_cb()"
)
list( strfldglb boolfldglb )
)

; Generate the form

procedure( myformcreateform()
    if( !boundp(`myformglb) || (myformglb == nil)
        then
            myformglb = hiCreateAppForm(
                ?name `myformglb
                ?fields myformDefineFields()
                ?formTitle "My first Form"
                ?callback "myformglb_cb()"
                ?buttonLayout `OKCancel
            )
        )
    myformglb
)

; Display the form
procedure( myformdisplayform()
    myformcreateform()          ;;; Creation and Display are separate
    hiDisplayForm(`myformglb)
)

; Check the current state of the form
hiIsFormDisplayed(myformglb) =>nil

myformdisplayform()
hiIsFormDisplayed(myformglb)
=>t
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateButton

```
hiCreateButton(  
    ?name s_fieldName  
    ?callback g_callback  
    [ ?buttonText t_buttonText ]  
    [ ?buttonIcon g_buttonIcon ]  
    [ ?showTextWithIcon g_showTextWithIcon ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?toolTip l_tooltip ]  
)  
=> r_fieldHandle
```

Description

Creates a standalone push button for a form or two-dimensional menu. The button must have an associated callback procedure to be executed when the button is selected. A button-only field differs from a button box field in that there is no prompt associated with this button.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name *s_fieldName* Handle to this field within a form. To reference it, use:

formHandle->*hiFieldSym*

?callback *g_callback*

A string, function object, or symbol type callback. The callback is called as soon as the button state changes.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?buttonText *t_buttonText*

String that appears on the button.

?buttonIcon *g_buttonIcon*

Cadence User Interface SKILL Reference

Form Functions

The icon for the button.

This argument overrides any *buttonText* specified, unless *g_showTextWithIcon* is set to a non-nil value.

?showTextWithIcon *g_showTextWithIcon*

If you specify a non-nil value for this argument, and if both *g_buttonIcon* and *t_buttonText* are specified, then the button displays both an icon and text. The default value of this argument is *nil*.

?help *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?font *t_font* A font argument is accepted but is not supported at this time.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is *nil*.

?toolTip *l_tooltip*

Specify a list of strings to be used as tooltips for buttons in the form.

Value Returned

r_fieldHandle Handle to the button.

Examples

Creates a new button *User Button* and specifies a tooltip *My Tooltip* for this button.

```
userButton = hiCreateButton(  
    ?name 'userButton  
    ?buttonText "User Button"  
    ?toolTip "My Tooltip"  
)
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateButtonBoxField

```
hiCreateButtonBoxField(  
    ?name s_fieldName  
    ?choices l_buttonText  
    ?callback l_callbackList  
    ?prompt t_fieldPrompt  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
)  
=> r_fieldHandle
```

Description

Creates a button box field for a form. A button box field contains individual buttons to be displayed in one row. Each button in the button box have an associated callback procedure that is run when that button is selected. A prompt is displayed on this row as well.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to this field within a form. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?choices <i>l_buttonText</i></code>	List of text strings representing the text displayed within each individual button in the button box (in left-to-right order).
<code>?callback <i>l_callbackList</i></code>	

Cadence User Interface SKILL Reference

Form Functions

List of callbacks. Each callback can be of string, symbol, or function object type. The number of button text entries must equal the length of the *l_callbackList*. The callback is executed when the corresponding button is pressed.

If function object type callback is specified, use the following syntax:

```
< button box field callback > ( o_field  
g_scope t_button_text)
```

where *t_button_text* is a string member given in the *?choices (l_buttonText)* argument. *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

```
formHandle->hiFieldSym->prompt
```

?prompt t_fieldPrompt

A string specifying the prompt for the field. To reference it, use:

```
formHandle-><fieldName>->prompt
```

?help g_fieldHelp

A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?font t_font

A font argument is accepted but is not supported at this time.

?enabled g_enabled

Specifies whether the fields in the layout are enabled. The default value is *t*.

?invisible g_invisible

Specifies whether the layout and the fields within are invisible. The default value is *nil*.

Values Returned

r_fieldHandle

Handle to the button box field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateComboField

```
hiCreateComboField(  
    ?name s_fieldName  
    ?items g_choices  
    [ ?prompt t_fieldPrompt ]  
    [ ?value t_currentValue ]  
    [ ?help g_fieldHelp ]  
    [ ?defValue t_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?format t_fieldFormat ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a drop-down combination field, which is a type-in field with an attached drop-down list. Users can either select an item from the list or type in a value.

In addition to the callbacks that can be specified through the arguments listed for this field, you can set a callback to run just before the drop-down list is displayed. This callback can be used to alter the contents of the drop-down list. You can set this callback by setting the `hiShowPopupCallback` property on the field with the following command:

```
hiFormHandle-><fieldName>->hiShowPopupCallback=t_value
```

where *hiFormHandle* is the handle to the form returned by `hiCreateAppForm`, *hiFieldSymbol* is the symbol of the field (set with the `?name` argument), and *t_value* is a string which need to be evaluated just before the drop-down list is displayed.

The combination field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Cadence User Interface SKILL Reference

Form Functions

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to this field within a form. To reference it, use : <code>formHandle-><fieldName></code>
<code>?items <i>g_choices</i></code>	A list of strings that represent the choices in the drop-down list. The default value is <code>nil</code> . To reference it, use: <code>formHandle-><fieldName>->items</code> A maximum of 42 items are displayed in the drop-down list; if there are more items, a scroll bar appears.
<code>?prompt <i>t_fieldPrompt</i></code>	A string specifying the prompt for the field. To reference it, use : <code>formHandle-><fieldName>->prompt</code>
<code>?value <i>t_currentValue</i></code>	Initial and current value of this field. To reference it, use : <code>formHandle-><fieldName>->value</code>
<code>?help <i>g_fieldHelp</i></code>	A string or symbol used to reference help. If not specified, <i>s_fieldName</i> is used. This argument is currently not used.
<code>?defValue <i>t_defaultValue</i></code>	Optional, default value of the field. This value is equal to the initial entry in <i>t_currentValue</i> if <i>t_defaultValue</i> is not specified. To reference the default value, use: <code>formHandle-><fieldName>->defValue</code>
<code>?font <i>t_font</i></code>	An optional font argument is accepted but is not supported at this time.
<code>?callback <i>g_callback</i></code>	

Cadence User Interface SKILL Reference

Form Functions

A string, symbol, or function object type callback that is executed if the value of the field changes. If a new value was typed in, the callback is called when the field has “lost focus” (for example, when the the user moves the cursor outside the form or tabs to another field) or when the user selects any button in the form.

If a new value was selected from the drop-down list, then the callback is called immediately, not when the field loses focus.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?modifyCallback *g_modifyCallback*

Cadence User Interface SKILL Reference

Form Functions

SKILL function to be executed whenever any change is made in the text field, for example, by typing a value. The modify callback is called when a change is detected in the field but before the change is displayed. The callback can be of string, symbol, or function object type.

A modify callback of symbol or function object type is passed following arguments:

```
<field modify callback>( o_field g_scope  
t_latestTextValue g_sourceOfChange)
```

, where *o_field* is the field whose modify callback is triggered, *g_scope* is where the field is instantiated, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil*, *t* indicates that the change was made through the program (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback of string type is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a string. This value replaces the current value of the field.

```
?focusInCallback g_focusInCallback
```


Cadence User Interface SKILL Reference

Form Functions

SKILL function to be executed whenever the field comes into focus, specifically when the type-in part of the field comes into focus. This happens either when you click in the type-in part of the field or when you dismiss the drop-down list, which shifts the focus to the type-in part of the field.) The callback can be of string, symbol, or function object type.

`?format t_fieldFormat`

Optional string that can control the display of the value on the form. It is similar to `printf` and its default value is `"%s"`.

Acceptable format characters are those allowed by the SKILL programming language.

`?editable g_editable`

Enables the field for editing.

`?enabled g_enabled`

Specifies whether the field is enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the field is invisible. The default value is `nil`.

`?nextField g_nextField`

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom — beginning with the field at the top-left corner of the form.

Values Returned

`r_fieldHandle`

Handle to the drop-down combo field.

Examples

```
mycombo=(hiCreateComboField ?name 'combo1 ?items '("1" "2" "3" "All")...)
mycombo->insertPolicy = 'InsertAtTop
```

Cadence User Interface SKILL Reference

Form Functions

Additional Information

You can set the insert policy (`fieldHandle->insertPolicy=s_insertPolicy`) by setting the `insertPolicy` property on the field object.

`s_insertPolicy` Optionally sets the insert policy for items in a combo box.

Accepted values are:

- `NoInsert`: new value cannot be inserted into the items list.
- `InsertAtTop`: new value can be inserted as the first item the items list.
- `InsertAtBottom`: new value can be inserted after the last item in the `?items` list.

If the insert policy is not specified, the default is `'NoInsert` (for backward compatability). Additionally, in the event of a combo-field value being changed, if the new value is not a member of the `?items` list, and the `insertPolicy` value is either `'InsertAtTop` or `'InsertAtBottom`, `?items` gets updated with the newly added value.

Setting `form->combo->insertPolicy` to either `'InsertAtTop` or `'InsertAtBottom`, results in *any* new value being added to the `combo->items` list. This also occurs if you enter a new value and select the *Cancel* button on the form. Here, `hiFormCancel` still revert the `combo->value`, but not `combo->items`.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateCyclicField

```
hiCreateCyclicField(  
    ?name s_fieldName  
    ?choices l_enumerations  
    ?prompt t_fieldPrompt  
    [ ?value g_currentValue ]  
    [ ?defValue g_defaultValue ]  
    [ ?callback g_callback ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?keepHistory g_keepHistory ]  
)  
=> r_fieldHandle
```

Description

Creates a cyclic field for a form. A cyclic field contains a list of string or icon enumerations, which are displayed in a drop-down list. Only one item can be selected at a time.

A maximum of 42 items are displayed in the drop-down list; if there are more items, a scroll bar appears.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name s_fieldName Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

?choices l_enumerations

List of strings and/or icons that can be cycled through. The current value and default value (if any) must be contained within this enumerated list.

A maximum of 42 items are displayed in the drop-down list; if there are more items, a scroll bar appears.

?prompt t_fieldPrompt

Cadence User Interface SKILL Reference

Form Functions

String specifying the prompt for the field. To reference it, use:

```
formHandle->hiFieldSym->prompt
```

?value *g_currentValue*

The current value of the cyclic field. Must be a string or icon list specified in *l_enumerations*. If *g_currentValue* is not specified, it defaults to the first choice in the enumerations list. To reference the current value, use:

```
formHandle->hiFieldSym->value
```

?defValue *g_defaultValue*

Default value of the field (in the correct format). This value is equal to the initial entry in *g_currentValue* if *g_defaultValue* is not specified. To reference the default value, use:

```
formHandle->hiFieldSym->defValue
```

?callback *g_callback*

Cadence User Interface SKILL Reference

Form Functions

Callback for this field. If a callback of string type is specified, the item selected (either text or an icon) is appended to this callback before being executed by SKILL.

If the callback is of symbol or function object type, it is called with the following syntax:

```
<cyclic field callback>( o_field g_scope  
g_item)
```

, where *g_item* specifies the item selected, which can be either text or icon.

If the user selects an icon value in this field, then

```
form->field->value
```

is set to the icon list. You may associate a value or values with these icons by creating a multi-element list with `list`. The first element of this list must be the icon list. The second element may be user-defined data. When the value is set to an icon, you can retrieve the user data from this list. For example:

```
layer1Icon = dlDlistToIcon(...)
myIcon = dlDlistToIcon(...)
layer1List = list( layer1Icon getLayerId("layer1"))
myIconList = list( myIcon list("myIcon" 33) )
....
field = hiCreateCyclicField(?name 'icons ?prompt
"Choose one:")
?choices list(layer1List myIconList "red" "yellow")
?value myIconList)
```

In the above example,

`form->field->value` contains something like:

```
((345233 10 10) ("myIcon" 33))
```

where the first element describes the icon.

`?help g_fieldHelp` A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

`?font t_font` A font argument is accepted but is not supported at this time.

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Cadence User Interface SKILL Reference

Form Functions

Specifies whether the layout and the fields within are invisible.
The default value is `nil`.

`?keepHistory` *g_keepHistory*

`t` or `nil`. If you specify `t`, attempts to retain the `?value` when choices are changed. The default value is `nil`.

Values Returned

r_fieldHandle Handle to the cyclic field.

Examples

The following sample displays different types of fields in a form.

```
;;; creating the list of items in the cyclic field
trCyclicList = '(
    "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"
    "11" "12" "13" "14" "15" "16" "17" "18" "19" "20" "21" "22"
)

;;; creating the cyclic field
trCyclicField = hiCreateCyclicField(
    ?name      'trCyclicField
    ?prompt    "Cycle Through: "
    ?choices   trCyclicList
    ?value     "3"
    ?defValue  "7"
    ?callback  "println"
)

;;; creating the boolean button field.
;;; The callback for the trBooleanButton field dynamically retrieves the new value
;;; of the field and embeds it in a message
trBooleanButton = hiCreateBooleanButton(
    ?name      'trBooleanButton
    ?buttonText "Boolean"
    ?value     t
    ?defValue  nil
    ?callback  "println( hiGetCurrentForm()->trBooleanButton-> value )"
)

;;; creating the button box field
trButtonBoxField = hiCreateButtonBoxField(
    ?name      'trButtonBoxField
    ?prompt    "Button Box"
    ?choices   '("Do a" "Do b" "Do c")
    ?callback  '("println( 'a )" "println( 'b )" "println( 'c)" )
)
```

Cadence User Interface SKILL Reference

Form Functions

```
;;; creating the radio field.
;;; The callback for the trConeRadioField field dynamically retrieves the new value
;;; of the field and imbeds it in a message
trConeRadioField = hiCreateRadioField(
    ?name      'trConeRadioField
    ?prompt     "Cone Size: "
    ?choices    list( "small" "medium" "large" )
    ?value      "small"
    ?defValue    "large"
    ?callback   '( "printf( \n\n%s cone chosen \" hiGetCurrentForm()
->trConeRadioField->value )" )
)

;;; creating the scale field
trScaleField = hiCreateScaleField(
    ?name      'trScaleField
    ?prompt     "Slide "
    ?value      500
    ?defValue    250
    ?callback   "println(\"scale changed \")"
    ?range      0:750
)

;;; creating the label field
trLabelField = hiCreateLabel(
    ?name      'trLabelField
    ?labelText  "Label"
    ?justification CDS_JUSTIFY_RIGHT
)

;;; creating the form
hiCreateAppForm (
    ?name      'trSampleForm
    ?formTitle  "Sample Form"
    ?callback   "println( 'FormAction )"
    ?fields
        list(
            trCyclicField
            trBooleanButton
            trButtonBoxField
            trConeRadioField
            trScaleField
            trLabelField
        )
    ?unmapAfterCB  t
) ; hiCreateAppForm

;;; displaying the form
hiDisplayForm( trSampleForm )
```

Related Topics

Form and Field Functions

Creating Context Menus for Fields

Cadence User Interface SKILL Reference

Form Functions

hiCreateFileSelectorField

```
hiCreateFileSelectorField(  
    ?name s_fieldName  
    ?mode s_mode  
    [ ?prompt t_prompt ]  
    [ ?value t_currentValue ]  
    [ ?defValue t_defaultValue ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
    [ ?confirmOverwrite g_confirmOverwrite ]  
)  
=> r_fieldHandle
```

Description

Creates a type-in field similar to a string field, except that it also has a `Browse` button at the right of the type-in box. The `Browse` button, when pressed, brings up a file dialog box that lets users select a file or directory. You can use the `?mode` argument to specify what the user is allowed to select. The file dialog is modal, that is, it blocks access to the application while it is open. When the user makes a selection and closes the file dialog, the selection is entered into the type-in box.

This function uses the `hiDisplayFileDialog` function to create the file dialog.

The file selector field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

`?name s_fieldName` Handle to this field within a form. To reference it, use:
formHandle->hiFieldSym

Cadence User Interface SKILL Reference

Form Functions

?mode *s_mode*

Symbol specifying what the user is allowed to select in the file dialog box. This value is passed to `hiDisplayFileDialog`. Legal values are:

- `'anyFile`: Any file, whether it exists or not. This option is typically used for "Save As" dialog boxes.
- `'existingFile`: An existing file. This option is typically used for *Open File* dialog boxes.
- `'existingFiles`: zero or more existing file. This option is typically used for *Open File* dialog boxes.
- `'directory`: The file dialog displays both files and directories.
- `'directoryOnly`: The file dialog displays only directories.

The default value is `'existingFile`.

?prompt *t_prompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?value *t_currentValue*

Initial and current value of this field. To reference it, use:

`formHandle->hiFieldSym->value`

?defValue *t_defaultValue*

Displays the default value of the field. This value is equal to the initial entry in ?value if ?defValue is not specified. To reference the value, use:

`formHandle->hiFieldSym->defValue`

?callback *g_callback*

Cadence User Interface SKILL Reference

Form Functions

A callback of string, symbol, or function object type. For type-in fields, the callback is called when the field "loses focus" (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form. If you want a callback that is called when the field comes into focus, use the *g_focusInCallback* argument.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

Cadence User Interface SKILL Reference

Form Functions

?modifyCallback *g_modifyCallback*

SKILL function to be executed whenever any change is made in the text field, for example, by typing a value. The modify callback is called when a change is detected in the field but before the change is displayed. The callback can be of string, symbol, or function object type.

A modify callback of symbol or function object type is passed following arguments:

```
<field modify callback>( o_field g_scope  
t_latestTextValue g_sourceOfChange)
```

, where *o_field* is the field whose modify callback is triggered, *g_scope* is where the field is instantiated, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil*, *t* indicates that the change was made through the program (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback of string type is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a string. This value replaces the current value of the field.

Cadence User Interface SKILL Reference

Form Functions

`?focusInCallback` *g_focusInCallback*

SKILL function to be executed whenever the field comes into focus.

`?editable` *g_editable*

Makes the field editable, `nil` to make it read-only.

`?enabled` *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible` *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?nextField` *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the *tabOrderIsAddOrder* argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

`?confirmOverwrite` *g_confirmOverwrite*

Specifies whether confirmation dialog is needed when overwriting an existing file. The default value is `t`.

Values Returned

r_fieldHandle Handle to the file selector field.

Examples

```
(setq workDir (prependInstallPath "test"))
(setq myField (hiCreateFileSelectorField
  ?name 'myField
  ?mode 'existingFile
  ?prompt "File Name"
  ?value (strcat workDir "/*.il")
  ?defValue (strcat workDir "/myCode.il")))
```

Cadence User Interface SKILL Reference

Form Functions

```
(putprop myField "Choose a Script" 'hiFileDialogCaption)
(putprop myField "myCode.il" 'hiFileDialogDefaultValue)
(putprop myField "SKILL Script (*.il);;All Files (*)" 'hiFileDialogFilter)
(putprop myField workDir 'hiFileDialogWorkingDir)
```

Additional Information

You can set the following properties on the file selector field to specify attributes for the file dialog box:

Property Name	Type	Default Value
hiFileDialogCaption	String	"Choose a File" or "Choose a Directory"
hiFileDialogDefaultValue	String	" "
hiFileDialogFilter	String	" "
hiFileDialogWorkingDir	String	". "
hiFileDialogMode	Symbol	'existingFile
hiFileDialogCallback	String	""
hiFileDialogUserData	Symbol	'_hiNoData
hiSimplifyFilename	t nil	nil
hiDontResolveLinks	t nil	nil

The first seven properties correspond to the arguments used for creating a file dialog box with `hiDisplayFileDialog` and are passed to `hiDisplayFileDialog`. If `hiSimplifyFilename` is set to `t`, the path of the file or directory selected in the file dialog box is expanded to an absolute (full) path before it is displayed in the type-in field. If `hiDontResolveLinks` is set to `t`, links in the path are not resolved when the path is being expanded to an absolute path.

For example, to put properties on a file selector field called `myField`, do the following:

```
myField->hiFileDialogCaption="Choose a Script"
myField->hiFileDialogDefaultValue="myCode.il"
myField->hiFileDialogFilter= "SKILL Script (*.il);;All Files (*)"
myField->hiFileDialogWorkingDir="/usr1"
myField->hiFileDialogMode='anyFile
myField->hiSimplifyFilename=t
myField->hiDontResolveLinks=nil
```

Cadence User Interface SKILL Reference

Form Functions

The file selector field does not have special formats like the string field.

Related Topics

[Form and Field Functions](#)

[hiDisplayFileDialog](#)

[hiCreateStringField](#)

hiCreateDisclosureButton

```
hiCreateDisclosureButton(  
    ?name s_name  
    [ ?callback g_callback ]  
    [ ?value g_value ]  
    [ ?defValue g_defValue ]  
    [ ?help S_help ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?onFields l_onFields ]  
    [ ?offFields l_offFields ]  
)  
=> r_fieldHandle
```

Description

A disclosure button is a button without a label used for displaying the additional information or functionality associated with another widget.

By default, a disclosure button is displayed in the closed position, pointing down [v]. When you click a button, it points up [^] and triggers the `?callback`, which is usually used to display additional fields.

This function checks for duplicate fields in the `onFields` and `offFields` lists. In case a field appears in both lists, it gets removed from both lists and a warning message is displayed.

In addition, whenever a field is added to a form after the form is displayed, the form code checks if the field is listed in any disclosure field's `disclosureList`. If it finds the field in any one of the lists, the code sets the visibility of the field appropriately prior to adding the field to the form.

Arguments

<code>?name <i>s_name</i></code>	Handle to this field within a form. To reference it, use: <code>formHandle->fieldSym</code>
<code>?callback <i>g_callback</i></code>	

Cadence User Interface SKILL Reference

Form Functions

A string, function object, or symbol type callback. The callback is called as soon as the button state changes.

If the callback is of String type, you can specify one or more SKILL functions or statements that should be executed when the value of the field changes.

Default value: `nil`

`?value g_value`

The initial and current value of the field. It must be an integer within the range specified by `l_rangeOfItems`.

Default value: `nil`

`?defValue g_defValue`

Optional default value of the field. This value is equal to the initial entry in `g_value` if `defaultValue` is not specified. To reference the default value, use:

`formHandle->fieldSym->defValue`

`?help S_help`

A string or symbol used to reference help. If not specified, `s_name` is used. This argument is currently not used.

Default value: `" "`

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?onfields l_onFields`

If specified, it must be a list of the symbols of fields. All the fields in the `?onFields` list are made visible if the disclosure field value is not `nil`.

Default value: `nil`

`?offFields l_offFields`

If specified, it must be a list of the symbols of fields. All fields in the `?offFields` list are made visible if the disclosure field value is `nil`.

Default value: `nil`

Cadence User Interface SKILL Reference

Form Functions

Values Returned

r_fieldHandle Handle to the disclosure button.

Examples

```
(defun make_font_selection_form ( name use_disclosure_button )
  (let ((attachment (bor hicLeftPositionSet hicTopPositionSet))
        (font_disclosure_args ;; args for hiCreateDisclosure... below
          (list '?name 'trFont '?value nil
                '?onFields '(cmbSt cmbHt cmbJt cmbES)
                '?offFields '(cmbSm)
                '?callback (lambda (field scope value)
                           (hiReattachField
                            scope 'trCo
                            (list (list 5 (if value 170 44)) 20:20 10)
                            (bor hicLeftPositionSet hicTopPositionSet))))))
    (fields
     (list
      (list (hiCreateComboField
             ?name 'cmbSm ?prompt " " ?items '("Stick") ?value "Stick")
             110:10 160:25 10)
       (list (hiCreateComboField
             ?name 'cmbSt ?prompt "          Style" ?items '("Stick")
             ?value "Stick" ?invisible t) 50:40 220:25 70)
       (list (hiCreateComboField
             ?name 'cmbHt ?prompt "          Height" ?items '("0.0625")
             ?value "0.0625" ?invisible t) 50:70 220:25 70)
       (list (hiCreateComboField
             ?name 'cmbJt ?prompt "Justification" ?items '("Lower Center")
             ?value "Lower Center" ?invisible t) 50:100 220:25 70)
       (list (hiCreateComboField
             ?name 'cmbES ?prompt "Entry Style" ?items '("Fixed Offset")
             ?value "Fixed Offset" ?invisible t) 50:130 220:25 70))))
    (unless use_disclosure_button
      font_disclosure_args= (append font_disclosure_args
                                     ' (?labelText "Font (trFont)"))
      fields= (cons (list
                    (apply (if use_disclosure_button 'hiCreateDisclosureButton
                              'hiCreateDisclosureTriangle) font_disclosure_args)
                    5:14 20:20 10)
                  fields))
    (hiCreateAppForm
     ?name name ?formTitle (sprintf nil "Font Selection (%s)" name)
     ?buttonLayout 'OKCancel ?dontBlock t ?initialSize 300:120 ?fields fields
     ?attachmentList (list attachment attachment attachment attachment
                          attachment attachment attachment)))
  (hiDisplayForm (make_font_selection_form (gensym 'form) t) 200:100))
```

Related Topics

Form and Field Functions

hiCreateDisclosureTriangle

```
hiCreateDisclosureTriangle(  
    ?name s_name  
    [ ?labelText t_labelText ]  
    [ ?callback g_callback ]  
    [ ?value g_value ]  
    [ ?defValue g_defValue ]  
    [ ?help S_help ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?onFields l_onFields ]  
    [ ?offFields l_offFields ]  
    [ ?persistent g_state ]  
)  
=> r_fieldHandle
```

Description

A disclosure triangle is a low profile widget for displaying the additional information or functionality embedded in a form.

By default, a disclosure triangle is displayed in the closed position, pointing to the right (>). When you click the triangle, it points down (v) and triggers the `?callback`, if specified. The `?callback` usually displays or hides additional information in the form.

This function checks for duplicate fields in the `onFields` and `offFields` lists. In case a field appears in both lists, it gets removed from both lists and a warning message is displayed.

In addition, whenever a field is added to a form after the form is displayed, the form code checks if the field is listed in any disclosure field's `disclosureList`. If it finds the field in any one of the lists, the code sets the visibility of the field appropriately prior to adding the field to the form.

Arguments

`?name s_name` Handle to this field within a form. To reference it, use:

`formHandle->fieldSym`

`?labelText t_labelText`

The prompt string, which appears as the label.

Default value: " "

`?callback g_callback`

Cadence User Interface SKILL Reference

Form Functions

A string, function object, or symbol type callback. The callback is called as soon as the button state changes.

If the callback is of String type, you can specify one or more SKILL functions or statements that should be run when the value of the field changes.

Default value: `nil`

`?value g_value`

The initial and current value of the field. It must be an integer within the range specified by `l_rangeOfItems`.

Default value: `nil`.

`?defValue g_defValue`

Optional default value of the field. This value is equal to the initial entry in `g_value` if `defaultValue` is not specified. To reference the default value, use:

`formHandle->fieldSym->defValue`

Default value: `None`.

`?help S_help`

A string or symbol used to reference help. If not specified, `s_name` is used. This argument is currently not used.

Default value: " "

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?onFields l_onFields`

If specified, it must be a list of the symbols of fields. All the fields in the `?onFields` list are made visible if the disclosure field value is not `nil`.

Default value: `nil`

`?offFields l_offFields`

Cadence User Interface SKILL Reference

Form Functions

If specified, it must be a list of the symbols of fields. All the fields in the `?offFields` list are made visible if the disclosure field value is `nil`.

Default value: `nil`

`?persistent g_state`

When set to `t`, the initial state of the triangle is restored to the previously saved state and its final state is saved.

When set to `nil`, the initial triangle state is default (close) and its final state is not saved.

Default value: `t`

The fields specified by `?onFields` and `?offFields` must have the same parent of the disclosure field. Therefore, these fields should be in the same scroll region, on the same page of a tab-field, or directly placed in the same form.

In addition, if the form is a `hiLayout` form, then the `?onFields` and `?offFields` list also includes symbols of layouts. These layouts must have the same parent of the disclosure field.

Values Returned

`r_fieldHandle` Handle to the disclosure triangle.

Examples

```
rotateLeftBtn = hiCreateFormButton(  
    ?name 'rotateLeftBtn  
    ?buttonText ""  
    ?buttonIcon list("rotate-90-left.png" 24 24)  
    ?invisible nil  
)  
rotateRightBtn = hiCreateFormButton(  
    ?name 'rotateRightBtn  
    ?buttonText ""  
    ?buttonIcon list("rotate-90-right.png" 24 24)  
    ?invisible nil  
)  
flipVerticalBtn = hiCreateFormButton(  
    ?name 'flipVerticalBtn  
    ?buttonText ""  
    ?buttonIcon list("flip-horizontal-axis.png" 24 24)  
    ?invisible nil  
)  
flipHorizontalBtn = hiCreateFormButton(  
    ?name 'flipHorizontalBtn  
    ?buttonText ""
```

Cadence User Interface SKILL Reference

Form Functions

```
?buttonIcon list("flip-vertical-axis.png" 24 24)
?invisible nil
)
myDiscForm = hiCreateLayoutForm( 'myDiscForm "My Disclosure Form"
    hiCreateVerticalBoxLayout( 'myMainLayout ?items list(
        hiCreateGridLayout( 'myGridLayout
            ?items list(
                list( hiCreateDisclosureTriangle( ?name 'trFont
                    ?labelText "Font (trFont)" ?value nil
                    ?onFields '(cmbSt cmbHt cmbJt cmbES iconBtnLabel
                        cmbStLbl cmbHtLbl cmbJtLbl cmbESLbl
                        hBoxIconBtn)
                    ?offFields '(cmbSm))
                    'horiz_align 'left 'row 0 'col 0)
                list( hiCreateComboField( ?name 'cmbSm ?prompt " "
                    ?items '("Stick") ?value "Stick")
                    'horiz_align 'left 'row 0 'col 1)
                list( hiCreateLabel( ?name 'cmbStLbl ?labelText "Style"
                    ?invisible t ?justification 'right)
                    'horiz_align 'right 'row 1 'col 0)
                list( hiCreateComboField( ?name 'cmbSt ?prompt " "
                    ?items '("Stick") ?value "Stick" ?invisible t)
                    'horiz_align 'left 'row 1 'col 1)
                list( hiCreateLabel( ?name 'cmbHtLbl ?labelText "Height"
                    ?invisible t ?justification 'right)
                    'horiz_align 'right 'row 2 'col 0)
                list( hiCreateComboField( ?name 'cmbHt ?prompt " "
                    ?items '("0.0625") ?value "0.0625" ?invisible t)
                    'horiz_align 'left 'row 2 'col 1)
                list( hiCreateLabel( ?name 'cmbJtLbl ?invisible t
                    ?labelText "Justification"
                    ?justification 'right)
                    'horiz_align 'right 'row 3 'col 0)
                list( hiCreateComboField( ?name 'cmbJt ?prompt " "
                    ?items '("Lower Center") ?value "Lower Center"
                    ?invisible t)
                    'horiz_align 'left 'row 3 'col 1)
                list( hiCreateLabel( ?name 'cmbESLbl ?invisible t
                    ?labelText "Entry Style"
                    ?justification 'right)
                    'horiz_align 'right 'row 4 'col 0)
                list( hiCreateComboField( ?name 'cmbES ?prompt " "
                    ?items '("Fixed Offset") ?value "Fixed Offset"
                    ?invisible t)
                    'horiz_align 'left 'row 4 'col 1)
                list( hiCreateLabel( ?name 'iconBtnLabel
                    ?labelText "Icon Buttons"
                    ?justification 'right ?invisible t)
                    'horiz_align 'right 'row 5 'col 0)
                list( hiCreateHorizontalBoxLayout('hBoxIconBtn
                    ?items list(rotateLeftBtn rotateRightBtn
                        flipVerticalBtn flipHorizontalBtn )
                    ?margins '(nil nil nil nil) ?spacing 0 ?invisible t)
                    'horiz_align 'left 'row 5 'col 1)
                list( 'col_stretch 1 1 )
            )
        )
        list( 'stretch_item 1 )
    ))
    ?buttonLayout 'OKCancel ?minSize 275:70)
hiDisplayForm( myDiscForm)
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

hiCreateFloatField

```
hiCreateFloatField(  
    ?name s_fieldName  
    [ ?value f_currentValue ]  
    [ ?acceptNil g_acceptNil ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?defValue f_defaultValue ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?range l_fieldRange ]  
    [ ?format t_fieldFormat ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a float field for a form. A float field is a field which accepts floating-point input only. A format string can be specified which is automatically imposed on any user input. An optional range can be specified and enforced for this field as well.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

You can also override the default floating point format specification for all `hi` float fields by setting the following variable in your `.cdsenv` file:

```
ui defaultFloatFieldFormat string "format"
```

If this variable is not set, float fields have a default format specification of `"%.6g"`, which displays the value to six significant digits.

Cadence User Interface SKILL Reference

Form Functions

Arguments

?name *s_fieldName* Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

?value *f_currentValue*

Initial and current value of this field. It is a required parameter, and may be referenced as:

`formHandle->hiFieldSym->value`

?acceptNil *g_acceptNil*

If set to `t`, the field accepts `nil` as a value. If there is no value entered for the field on the form, it results in the SKILL value being set to `nil`. Setting a field value to `nil` using the CIW results in the field on the form to be blank.

?prompt *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?defValue *f_defaultValue*

Optional, default value of the field (in the correct format). This value is equal to the initial entry in *currentValue* if *defaultValue* is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

?help *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?font *t_font* A font argument is accepted but not supported at this time.

?callback *g_callback*

Callback of string, symbol, or function object type. The callback is called when the field has “lost focus” (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?modifyCallback *g_modifyCallback*

Cadence User Interface SKILL Reference

Form Functions

Do not use the *g_modifyCallback* argument with this field because it may not work as expected. Use this argument only with string fields.

SKILL function to be executed whenever any change is made in the text field, for example, by typing or pasting. The modify callback is called when a change is detected in the field but before the change is displayed.

The modify callback function is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a floating point. This value replaces the current value of the field.

```
?focusInCallback g_focusInCallback
```

SKILL function to be executed whenever the field comes into focus. You provide this function, which could be of string, symbol, or function object type.

```
?range l_fieldRange
```

Optional list containing the minimum and maximum float values. For example, the list (0.5 10.9), also specified as 0.5:10.9, represents the range of floats from 0.5-10.9 inclusive. To specify an unbounded upper range, use either *nil* or the string "infinity" as the second value. To specify an unbounded lower range, use either *nil* or the string "-infinity" as the first value.

Cadence User Interface SKILL Reference

Form Functions

?format *t_fieldFormat*

Optional string that can control the display of the value on the form. It is similar to `printf`.

Acceptable format characters are those allowed by the C programming language.

?editable *g_editable*

If set to `nil`, the field is read-only; setting *g_editable* to `t` makes the field editable.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is `t`.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

?nextField *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the *tabOrderIsAddOrder* argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Value Returned

r_fieldHandle Handle to the float field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateFrameField

```
hiCreateFrameField(  
    ?name s_fieldName  
    [ ?labelText t_labelText ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
)  
=> r_fieldHandle
```

Description

Creates a frame field for a two-dimensional form. A frame field is a border that surrounds a group of related fields so they are easier to see.

If *t_labelText* is specified and you specify 2D attributes for the frame field, you must also specify the label height in the 2D attributes even though the value is ignored. For example:

```
list( r_framefield x_xpos:x_ypos x_width:x_height x_labelHeight)
```

The frame field created by `hiCreateFrameField` is not supported by the `hiLayout` form. The frame field gets replaced by a frame property on the layout container. In this case, the frames have the advantage to automatically fit to the layout container, even when fields are added or deleted from a container.

Arguments

?name s_fieldName Handle to the frame field.

?labelText t_labelText

Title of the frame field. If the title is specified it appears in the upper left corner of the frame. The title must not extend beyond the width of the frame.

?enabled g_enabled

Specifies whether the fields in the layout are enabled. The default value is `t`.

?invisible g_invisible

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

r_fieldHandle Field handle for the frame field.

Examples

```
f = hiCreateFrameField( ?name 'f ?labelText "New Frame Field" )
g = hiCreateAppForm( ?name 'g ?fields list ( list ( f 10:20 200:200 14 ) ) )
hiDisplayForm ( g )
```

Related Topics

[Form and Field Functions](#)

hiCreateScrollRegion

```
hiCreateScrollRegion(  
    ?name s_name  
    [ ?fields l_fieldEntries ]  
    [ ?attachmentList l_fieldAttachments ]  
    [ ?tabOrderIsAddOrder g_tabOrderIsAddOrder ]  
    [ ?labelText t_labelText ]  
    [ ?borderWidth x_borderWidth ]  
    [ ?scrollBars g_scrollBars ]  
    [ ?rightMargin x_rightMargin ]  
    [ ?bottomMargin x_bottomMargin ]  
    [ ?help g_help ]  
    [ ?invisible g_invisible ]  
    [ ?enabled g_enabled ]  
)  
=> r_fieldHandle
```

Description

Creates a scroll region field, which can be scrolled and can contain any type of field. A scroll region field is like a sub-form within a form.

A scroll region field can be used as a field only on 2D forms. Once the field is instantiated, it is referenced through the form that contains it (see the description for the *s_name* argument below for more information).

A scroll region field can be specified as a form field for all functions that take form field arguments. It can also be specified as the form argument for all functions for adding, moving, resizing and deleting fields from a form.

A scroll region field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

A scroll region field that has an *?attachmentList* specified does not get scroll bars.

Do not place a scroll region field twice in one form or two scroll region fields with the same name in one form.

You cannot place a scroll region field that has field attachments inside a form or scroll region field that does not have field attachments.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Cadence User Interface SKILL Reference

Form Functions

Arguments

`?name s_name`

Handle to the scroll region field within a form. Once the form is instantiated, the field is referenced through the form. For example, to update the fields or access properties in a scroll region field `myRegion` which is in a form `myForm`, you use:

```
myForm->myRegion
```

`?fields l_fieldEntries`

List of field descriptors returned from the field creation routines. These can be fields of any type, including other scroll region fields. You must use at least one field entry. There is no upper limit on the number of fields you can use. You must specify 2D attributes for each field entry.

If the field entries list includes any scroll region fields, the scroll region field is copied. Scroll region fields are never shared.

All fields placed in a scroll region field must be referenced through the form and scroll region field. If the scroll region field is within another scroll region field, then its fields are accessed through the form and both scroll region fields. For example, to access the value of a field `myField`, which is in a scroll region field `myRegion` in a form `myForm`, you use:

```
myForm->myRegion->myField->value
```

In the above example, if you also have a `myRegion2` with a `myField2`, then to access the value of `myField2` you use:

```
myForm->myRegion->myRegion2->myField2->value
```

In case you want to place a scroll region field in an `hiLayout` form, the `?fields` list must contain only an `hiLayout`. Similarly, it applies to the `?field` list for a tab field that needs to be placed in an `hiLayout` form. The `?fields` list passed to `hiCreateTabField` must contain only an `hiLayout` for each page, where each page of a tab field is a scroll region.

The `hiScrollRegionSym` is a property on each field in a form that exists after the form is displayed and that identifies whether or not that field is contained in a scroll region or tab page. If the value of the property is `nil`, the field is at the top level of the form. If it is non-`nil`, it is the symbol name of the scroll region that contains that field.

Cadence User Interface SKILL Reference

Form Functions

?attachmentList *l_fieldAttachments*

List of field attachments. Field attachments determine how the two-dimensional attributes for a field are used to position that field in the scroll region. The first field attachment is applied to the first field in *l_fieldEntries*, the second to the second field, and so forth. You must have the same number of attachments in *l_fieldAttachments* as the number of fields in *l_fieldEntries*. If you do not want to specify a field attachment for a field, use `nil` or `0`.

Each field attachment in *l_fieldAttachments* consists of a bitwise OR of a list of constants. Each constant is a specification for a side of the field (left, right, top, and bottom). These attachment specifications for the sides of a field can be positional or percentage attachments.

For example, an *l_fieldAttachments* list could be:

```
(constant|constant|constant|constant constant|
constant nil constant|constant|constant)
```

For details about field attachments and how to use positional and percentage constants, see [Field Attachments](#).

A scroll region field that has an ?attachmentList specified does not get scroll bars.

You cannot place a scroll region field that has field attachments inside a form or scroll region field that does not have field attachments.

?tabOrderIsAddOrder *g_tabOrderIsAddOrder*

t or `nil`. Determines the order in which fields are traversed when the Tab key is pressed. If you specify *t*, the *nextField* property is automatically set for the fields and the traversal order is the order in which fields are added to the form (that is, the order in which they are listed in the *l_fieldEntries* list). If you specify `nil`, the default traversal order is based on the form layout—left-to-right and top-to-bottom—beginning with the field at the top-left corner of the form. The default value of *g_tabOrderIsAddOrder* is `nil`.

The *nextField* property set on a field (to any non-`nil` value) overrides the *g_tabOrderIsAddOrder* value.

?labelText *t_labelText*

Cadence User Interface SKILL Reference

Form Functions

Title of the scroll region field. It is justified to the upper left corner of the bounding box of the field.

?borderWidth *x_borderWidth*

Thickness of the border shadow in pixels. The default is 2. When scroll bars appear, the border shrinks in size by the amount of space taken up by the scroll bars.

This property can be changed interactively at any time, even after the scroll region field has been instantiated.

?scrollBars *g_scrollBars*

Controls the appearance of scroll bars. Can have the following values:

'dynamic

'static/t

nil

(t|nil t|nil)

scroll bars automatically appear if a field that extends beyond the edge of the scroll region field is added.

scroll bars are always present, whether or not a field extends beyond the boundary of the scroll region field.

scroll bars are never present, even if a field extends beyond the boundary of the scroll region field.

Controls each scroll bar—horizontal and vertical—individually. If the first element in the list is *t*, the horizontal scroll bar is always present. If the second element in the list is *t*, the vertical scroll bar is present.

'dynamic or 'static cannot be used as values in this list.

If you use this list as the value of *g_scrollBars*, the horizontal and vertical scroll bars cannot be made dynamic.

Note: The scroll bars property can only be set when you create the scroll region field. Once the field is instantiated, you cannot change this property.

?rightMargin *x_rightMargin*

Cadence User Interface SKILL Reference

Form Functions

Distance, in pixels, between the right edge of the border shadow and the edge of the field. The default is 0.

The visible part of the scroll region is the width of the field minus the *x_rightMargin*.

This property can be changed interactively at any time, even after the scroll region field is instantiated.

?bottomMargin *x_bottomMargin*

Distance, in pixels, between the bottom edge of the border shadow and the edge of the field. The default is 0.

The visible part of the scroll region is the height of the field minus the *x_bottomMargin*.

This property can be changed interactively at any time, even after the scroll region field is instantiated.

?help *g_help*

A string, symbol, list of strings, or list of symbols used to reference help. If not specified, *s_name* is used. This argument is currently not used.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is *nil*.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

Value Returned

r_fieldHandle Handle to the scroll region field created.

Related Topics

[Form and Field Functions](#)

[Form Layout](#)

[Creating Context Menus for Fields](#)

hiIsScrollRegion

```
hiIsScrollRegion(  
    g_item  
)  
=> t / nil
```

Description

Determines if the item is a scroll region field.

Argument

<i>g_item</i>	Item to be checked.

Values Returned

t	The item is a scroll region field.
nil	The item is not a scroll region field.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateForm

```
hiCreateForm(  
    s_formHandle  
    t_formName  
    g_formAction  
    l_fieldEntries  
    g_formHelp  
    [ g_unmapAfterCB ]  
)  
=> r_form
```

Description

Use [hiCreateAppForm](#) instead of this function.

hiCreateFormButton

```
hiCreateFormButton(  
    ?name s_fieldName  
    ?buttonText t_buttonText  
    ?callback g_callback  
    [ ?buttonIcon g_buttonIcon ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?showTextWithIcon g_showTextWithIcon ]  
)  
=> r_fieldHandle
```

Description

Creates a standalone push button for a form or two-dimensional menu. This function is the same as `hiCreateButton`.

The button must have an associated callback procedure to be executed when the button is selected. A button-only field differs from a button box field in that there is no prompt associated with this button.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

`?name s_fieldName` Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

`?buttonText t_buttonText`

String that appears on the button.

`?callback g_callback`

Callback that is executed if the button is activated.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

`?buttonIcon g_buttonIcon`

Cadence User Interface SKILL Reference

Form Functions

	The icon for the button.
	This argument overrides <i>t_buttonText</i> , if it is specified.
<code>?help g_fieldHelp</code>	A string or symbol used to reference help. If not specified, <i>s_fieldName</i> is used. This argument is currently not used.
<code>?font t_font</code>	A font argument is accepted but is not supported at this time.
<code>?enabled g_enabled</code>	Specifies whether the fields in the layout are enabled. The default value is <i>t</i> .
<code>?invisible g_invisible</code>	Specifies whether the layout and the fields within are invisible. The default value is <i>nil</i> .
<code>?showTextWithIcon g_showTextWithIcon</code>	<i>nil</i> or non- <i>nil</i> . If you specify a non- <i>nil</i> value for this argument, and if both <i>g_buttonIcon</i> and <i>t_buttonText</i> are specified, then the button displays both an icon and text. The default value of this argument is <i>nil</i> .

Values Returned

<i>r_fieldHandle</i>	Handle to the button.

Related Topics

[hiCreateButton](#)

[hiCreateFormButton](#)

[hiDeleteForm](#)

[Creating Context Menus for Fields](#)

[Form and Field Functions](#)

hiCreateFormLabel

```
hiCreateFormLabel(  
    ?name s_name  
    [ ?labelText t_labelText ]  
    [ ?labelIcon g_labelIcon ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?justification s_justification ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
)  
=> r_fieldHandle
```

Description

This function is an alias for `hiCreateLabel`. Use `hiCreateLabel` instead of this function.

Arguments

<i>s_name</i>	Handle to the form label within the form.
<i>?labelText l_labelText</i>	Text for the label.
<i>?labelIcon g_labelIcon</i>	Icon which appears as the label.
<i>?help g_fieldHelp</i>	A string or symbol used to reference help. If this argument is not specified, <i>s_fieldName</i> is used. This argument is currently not used.
<i>?font t_font</i>	This argument is ignored.
<i>?justification s_justification</i>	Must be one of the following symbols: 'left', 'center', or 'right'. 'left' is the default.
<i>?enabled g_enabled</i>	Specifies whether the fields in the layout are enabled. The default value is <code>t</code> .
<i>?invisible g_invisible</i>	Specifies whether the layout and the fields within are invisible. The default value is <code>nil</code> .

Cadence User Interface SKILL Reference

Form Functions

Values Returned

r_fieldHandle Handle to the label.

Related Topics

Form and Field Functions

hiCreateLabel

hiCreateHypertextField

```
hiCreateHypertextField(  
    ?name s_name  
    [ ?title t_title ]  
    [ ?titleAlignment g_titleAlignment ]  
    [ ?value t_value ]  
    [ ?pos t_position ]  
    [ ?defValue t_defaultValue ]  
    [ ?help g_fieldHelp ]  
    [ ?callback g_callback ]  
    [ ?enabled g_enabled ]  
    [ ?shadowed g_shadowed ]  
    [ ?hasVerticalScrollbar g_hasVerticalScrollbar ]  
    [ ?hasHorizontalScrollbar g_hasHorizontalScrollbar ]  
    [ ?invisible g_invisible ]  
)  
=> r_fieldHandle
```

Description

Creates a hypertext field, which displays rich text with hypertext navigation. You can also specify hyperlinks to execute SKILL commands.

Angle brackets (less-than and greater-than characters) are used to enclose text formatting and other HTML commands. Therefore, to display a greater-than (">") or less-than ("<") character, you must use the special strings ">" or "<" respectively.

Links to a Web page is displayed on a Web browser. If a browser is already open, it is used, otherwise, the browser specified in the *Web Browser* field of the CIW User Preferences form is used. The default browser is Netscape. For more information on browser selection, see [hiLaunchBrowser](#).

The hypertext field is based on QTextBrowser (for reference you may want to look over the QTextBrowser class documentation).

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Cadence User Interface SKILL Reference

Form Functions

Arguments

<code>?name <i>s_name</i></code>	Handle to the field. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?title <i>t_title</i></code>	The title of the hypertext field, which is displayed above the field.
<code>?titleAlignment <i>g_titleAlignment</i></code>	Alignment of the hypertext field title. Use one of the following values: <code>'left</code> <code>'center</code> <code>'right</code> The default value is <code>'left</code> .
<code>?value <i>t_value</i></code>	The text, including hyperlinks, that is the initial and current value of the field. Specify the hyperlinks in the same format as an HTML anchor tag. You can specify three attributes in the anchor tag: ■ <code>HREF</code> Specifies the URL ■ <code>NAME</code> Specifies a name that identifies the link; you can jump to this from another link in the document ■ <code>TARGET</code> Specifies the window in which to load the page; for example, <code>TARGET="_blank"</code> loads the page in a new browser window while <code>TARGET="_self"</code> loads the page in the same window

Cadence User Interface SKILL Reference

Form Functions

For example:

```
"<A href=http://www.cadence.com> Cadence Design Systems</A>"
"<A name=cadence href=http://www.cadence.com>
Cadence Design Systems</A>"
"<A href=http://www.cadence.com target=_blank>
Cadence Design Systems </A>"
"<A href=#cadence> Jump to the Cadence link </A>"
```

All the standard formats for the HREF attribute are supported.
For example:

```
href=http://www.myWebsite.com
href=mailto://xyz@mycompany.com
href="#linkName"
href=/localFile
(localfile is loaded into the hypertext field when
the link is clicked.)
href=file:/localFile or href=file:///
localFile(localfile is always sent to the Web
browser.)
```

In addition, a SKILL href can be specified to execute a SKILL command. For example:

```
href=SKILL:println("\Hello world\")
```

You need to use the ampersand encoding for the less-than (" $<$ ") and greater-than (" $>$ ") characters in the SKILL command. For " $>$ " use ">" and for " $<$ " use "<". Otherwise, they are interpreted as HTML enclosures. Space and equal characters can also be misinterpreted and translated into ampersand encoded representations by the HTML interpreter. If this occurs for:

- Spaces separating arguments, replace the spaces with commas.
- An equal symbol, use the appropriate `set`, `setq`, `equal`, `eq`, etc. SKILL command to workaround the issue.

To specify a new line in the text, use `\n`.

Quotation marks must be preceded with a backslash.

Cadence User Interface SKILL Reference

Form Functions

After the form is instantiated, you can use the `hiHypertextReplace()` function or the `value` property to modify the value. Use the following syntax for the value property:

```
formHandle->hiFieldSym->value
```

You can also load a text file as the value of the hypertext field by using the `hiHypertextReadin()` function.

`?pos t_position`

Position of the field that is to be displayed initially, specified by a link name. For example, "myLink" displays the text associated with the tag `</A NAME=myLink>` when the field is first displayed.

After the form is instantiated, you can use the `pos` property to specify the part of the field that you want to display. For example:

```
formHandle->hiFieldSym->pos="linkToDisplay"
```

You can also use the `hiHypertextScroll()` function to specify the part of the field that you want to display.

`?defValue t_defaultValue`

The text, including hyperlinks, that is the default value of the field.

If you do not specify this argument, the default value is equal to the initial entry in `t_value`.

To reference the default value, use:

```
formHandle->hiFieldSym->defValue
```

`?help g_fieldHelp`

A string or symbol used to reference help. If not specified, `s_name` is used. This argument is currently not used.

`?callback g_callback`

Cadence User Interface SKILL Reference

Form Functions

Callback function of string, symbol, or function object type that is executed when a hyperlink is clicked. The callback function is passed from the following arguments:

fieldUT form hyperlinkName hyperlinkRef

where:

fieldUT is the field user type.

A field user type (UT) is the representation for an instantiated field. When a form is initially displayed (`hiDisplayForm`) or instantiated (`hiInstantiateForm`), the fields are converted from defstructs to user types, which are associated with the field widgets.

form is the form defstruct.

hyperlinkName is an empty string.

hyperlinkRef is the hyperlink reference (the `HREF` in the HTML anchor tag).

In the following example:

```
"<A name=cadence href=http://www.cadence.com> Cadence  
Design Systems</A>"
```

the hyperlink name is `cadence` and the hyperlink reference is `http://www.cadence.com`.

Although, in this example, the anchor name is set as `cadence`, the *hyperlinkName* passed to the callback always remains a blank string.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is `t`.

?shadowed *g_shadowed*

Locates a shadow box around the spin box field. The default value is `t`.

?hasVerticalScrollbar *g_hasVerticalScrollbar*

Displays a vertical scroll bar on the field. The default value is `t`.

?hasHorizontalScrollbar *g_hasHorizontalScrollbar*

Cadence User Interface SKILL Reference

Form Functions

Displays a horizontal scroll bar on the field. The default value is `t`.

`?invisible` *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

Values Returned

r_fieldHandle Handle to the hypertext field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

[hiCreateSimpleHypertextField](#)

[hiHypertextReadin](#)

[hiHypertextReplace](#)

[hiHypertextScroll](#)

[hiLaunchBrowser](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateIntField

```
hiCreateIntField(  
    ?name s_fieldName  
    [ ?prompt t_fieldPrompt ]  
    [ ?value x_currentValue ]  
    [ ?acceptNil g_acceptNil ]  
    [ ?help g_fieldHelp ]  
    [ ?defValue x_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?format t_fieldFormat ]  
    [ ?range l_fieldRange ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates an integer field entry for a form. An `int` field is a field that accepts integer input only. A format string can be specified which is imposed automatically on any user input. An optional range can be specified and enforced for this field as well.

An integer field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name *s_fieldName* Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

?prompt *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?value *x_currentValue*

Cadence User Interface SKILL Reference

Form Functions

Initial and current value of this field. To reference it, use:

`formHandle->hiFieldSym->value`

`?acceptNil g_acceptNil`

Accepts `nil` as a value. If there is no value entered for the field on the form, results in the SKILL value being set to `nil`. Setting a field value to `nil` using the CIW results in the field on the form to be blank.

`?help g_fieldHelp` A string or symbol used to reference help. If not specified, `s_fieldName` is used. This argument is currently not used.

`?defValue x_defaultValue`

Default value of the field. This value is equal to the initial entry in `currentValue` if `defaultValue` is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

`?font t_font` A font argument is accepted but not supported at this time.

`?callback g_callback`

Callback of string, symbol, or function object type that is executed when the value of a field changes. The callback is called when the field has lost focus (when the cursor leaves the form, the user tabs to another field, and so forth) or the user selects any button in the form.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

`?modifyCallback g_modifyCallback`

Cadence User Interface SKILL Reference

Form Functions

Do not use the *g_modifyCallback* argument with this field because it may not work as expected. Use this argument only with string fields.

SKILL function to be executed whenever any change is made in the text field, for example, by typing or pasting. The modify callback is called when a change is detected in the field but before the change is displayed.

The modify callback function is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the *value* of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be an integer. This value replaces the current value of the field.

```
?focusInCallback g_focusInCallback
```

SKILL function to be run whenever the field comes into focus. You provide this function.

```
?format t_fieldFormat
```

String that can control the display of the value on the form. It is similar to *printf*.

Acceptable format characters are those allowed by the C programming language.

```
?range l_fieldRange
```


Cadence User Interface SKILL Reference

Form Functions

A list containing the minimum and maximum integer values. For example, the list `(0 10)`, also specified as `0:10`, represents the range of integers from 0-10 inclusive. To specify an unbounded upper range, use either `nil` or the string `"infinity"` as the second value. To specify an unbounded lower range, use either `nil` or the string `"-infinity"` as the first value.

`?editable g_editable`

Enables the field for editing.

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?nextField g_nextField`

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Values Returned

`r_fieldHandle` Handle to the integer field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateLabel

```
hiCreateLabel(  
    ?name s_fieldName  
    [?labelText t_labelText]  
    [?labelIcon g_labelIcon]  
    [?justification s_justification]  
    [?font t_font]  
    [?enabled g_enabled]  
    [?invisible g_invisible]  
    [?help g_fieldHelp]  
)  
=> r_fieldHandle
```

Description

Creates a standalone label entry for a form or two-dimensional menu. This label-only field contains a descriptive label and can provide additional features to a form or two-dimensional menu.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name *s_fieldName* Handle to this field within a form. To reference it, use:
 formHandle->hiFieldSym

?labelText *t_labelText*

Cadence User Interface SKILL Reference

Form Functions

String which appears as the label.

You can change the label text after the form has been instantiated with the `value` property. For example:

```
form->labelField->value="NewLabelText "
```

Before the form is instantiated, however, the `_labelOnly` property contains the value of the label text and you can change it with:

```
form->labelField->_labelOnly="NewLabelText "
```

This means that you should set both properties when you want to modify the label text to ensure that it is correct:

```
form->labelField->value="NewLabelText "  
form->labelField->_labelOnly="NewLabelText "
```

If you specify both the `t_labelText` and `g_labelIcon` arguments, `g_labelIcon` is used.

?labelIcon *g_labelIcon*

Icon which appears as the label.

If you specify both the `t_labelText` and `g_labelIcon` arguments, `g_labelIcon` is used.

?justification *s_justification*

Must be one of the following symbols: 'left, 'center, or 'right. 'left is the default.

?font *t_font*

A font argument is accepted but not supported at this time.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is `t`.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

?help *g_fieldHelp*

A string or symbol used to reference help. If not specified, `s_fieldName` is used. This argument is currently not used.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

r_fieldHandle Handle to the label.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateLayerCyclicField

```
hiCreateLayerCyclicField(  
    d_techFileId  
    t_fieldPrompt  
    t_callback  
    l_layers  
    [ l_LPpair ]  
    [ s_fieldName ]  
)  
=> r_fieldHandle
```

Description

Creates a cyclic field entry containing icon representations of specified layers for a form.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

<i>d_techFileId</i>	Handle representing a techfile object.
<i>t_fieldPrompt</i>	Prompt string that appears in this field. To reference it, use: formHandle->hiLayerField->prompt
<i>t_callback</i>	Callback string for this field. If a callback is specified, the value of the item selected is appended to this callback before being executed by SKILL. The value of this field is always a list representing a layer icon. This value can be referenced by: form->hiLayerField->value The value is always a list containing the icon representation of the layer followed by a string representation. For example: (50331991 106 26 "device (drawing1)") The first three elements in this list represent the cyclic icon; the fourth element is the layer's name followed by its purpose in parentheses.

Cadence User Interface SKILL Reference

Form Functions

l_layers

List containing the layers that should be displayed as possible choices for this field. Each layer is represented as a list of two strings—layer name and layer purpose.

For example:

```
list( list("device" "drawing1")
      list("y0" "drawing")
      list("y1" "drawing")
      list("y2" "drawing") )
```

The layers specified must be defined in *d_techFileId*. If *nil* is specified, all layers in *d_techFileId* is displayed.

l_LPpair

An optional argument specifying the initial value of this layer field. This argument is a list containing the layer name (or number) and purpose. This layer must exist in *d_techFileId*. If not specified, the first layer in the list *l_layers* becomes the initial value.

s_fieldName

Handle to the field. If you do not specify this argument, the default name is 'hiLayerField'.

To reference the field after the form is created, use:

```
formHandle -> hiFieldSym
```

Values Returned

r_fieldHandle

Handle to the field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateListField

```
hiCreateListField(  
    ?name s_fieldName  
    [ ?value l_currentValue ]  
    [ ?help g_fieldHelp ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?defValue l_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a list field entry for a form. A list field accepts a valid SKILL expression as input. `linereadstring` determines if the expression is valid. The list expression ID is not evaluated; it is only checked to see if it is valid.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to this field within a form. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?value <i>l_currentValue</i></code>	Initial and current value of this field in the correct format. May be <code>nil</code> , in which case the list field appears blank. To reference the current value, use: <code>formHandle->hiFieldSym->value</code>
<code>?help <i>g_fieldHelp</i></code>	A string or symbol used to reference help. If not specified, <i>s_fieldName</i> is used. This argument is currently not used.
<code>?prompt <i>t_fieldPrompt</i></code>	

Cadence User Interface SKILL Reference

Form Functions

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?defValue *l_defaultValue*

Default value of the field in the correct format. This value is equal to the initial entry in *currentValue* if *defaultValue* is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

?font *t_font*

A font argument is accepted but not supported at this time.

?callback *g_callback*

A string, function object, or symbol type callback that is executed when the value of the field changes. The callback is called when the field has “lost focus” (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?modifyCallback *g_modifyCallback*

Cadence User Interface SKILL Reference

Form Functions

Do not use the *g_modifyCallback* argument with this field because it may not work as expected. Use this argument only with string fields.

SKILL function to be executed whenever any change is made in the text field, for example, by typing or pasting. The modify callback is called when a change is detected in the field but before the change is displayed.

The modify callback function is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a valid SKILL expression. This value replaces the current value of the field.

?focusInCallback *g_focusInCallback*

SKILL function to be executed whenever the field comes into focus. You provide this function.

?editable *g_editable*

Enables the field for editing.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

?invisible *g_invisible*

Cadence User Interface SKILL Reference

Form Functions

Specifies whether the layout and the fields within are invisible.
The default value is `nil`.

`?nextField g_nextField`

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Value Returned

`r_fieldHandle` Handle to the list field.

Related Topics

[Form and Field Functions](#)

hiCreateListBoxField

```
hiCreateListBoxField(  
    ?name s_fieldName  
    ?choices l_listItems  
    [ ?prompt t_prompt ]  
    [ ?value l_selectedItem ]  
    [ ?defValue l_defSelectedItem ]  
    [ ?valueByPosition g_valueByPosition ]  
    [ ?callback g_callback ]  
    [ ?changeCB g_changeCB ]  
    [ ?numRows x_numRows ]  
    [ ?multipleSelect g_multipleSelect ]  
    [ ?doubleClickCB g_doubleClickCB ]  
    [ ?CBOOnReselect g_CBOOnReselect ]  
    [ ?keepHistory g_keepHistory ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?help g_fieldHelp ]  
    [ ?selectionEnabled g_selectionEnabled ]  
)  
=> r_fieldHandle
```

Description

Creates a list box field for a form.

Any string item selection changes in this field are tracked and are accessible by SKILL.

Arguments

?name s_fieldName Handle to the list box field.

?choices l_listItems

List of items displayed in the list box field. Each item can be either a string, or a pair of (string *hiIcon*).

Each item is normally a null terminated string and must not contain the C newline character `\n`. The maximum limit on any individual string's length is 2047 bytes (2 kilobytes - 1) but strings should be kept short and concise.

Default Value: `nil` (nothing available to select)

?prompt t_prompt Prompt to appear on the left side of the list box field. If no prompt is specified, none appears.

Cadence User Interface SKILL Reference

Form Functions

?value *l_selectedItem*

Items selected in the list box field. The items must be valid values, that is, they must have been previously defined in *l_listItems*. You can specify the items as a list of strings or as a list of integers or ranges representing the position of the items. For example:

```
'( "b" "e" "f" "g" "j")  
'(2 5 6 7 10)  
'(2 '(5 7) 10)
```

The maximum limit on any individual string's length is 2047 bytes (2 kilobytes - 1).

The *g_valueByPosition* argument determines whether the *l_selectedItem* list is managed and reported as a list of strings or a list of integers. See the description of *g_valueByPosition* for more information.

Default Value: `nil` (nothing selected)

You can also set the value with the `value` property. For example:

```
form -> listbox -> value = l_newValue
```

?defValue *l_defSelectedItem*

Cadence User Interface SKILL Reference

Form Functions

Defaults selected in the list box field. The items must be defined in *l_listItems*. You can specify the items as a list of strings or as a list of integers or ranges representing the position of the items. For example:

```
'( "b" "e" "f" "g" "j")  
'(2 5 6 7 10)  
'(2 '(5 7) 10)
```

This argument is similar to *l_selectedItem* and is assigned to *l_defSelectedItem* when the user presses the *Defaults* button on the form.

The *g_valueByPosition* argument determines whether the *l_defSelectedItem* list is managed and reported as a list of strings or a list of integers. See the description of *g_valueByPosition* for more information.

You can also set the default value with the *defValue* property. For example:

```
form -> listbox -> defValue = l_newValue
```

Cadence User Interface SKILL Reference

Form Functions

`?valueByPosition` *g_valueByPosition*

Lists *l_selectedItem* and *defSelectedItem* through the items in the value and default value. These are managed and reported as integers representing their position. The position is determined by the order of the items in the list box field—the first item is 1, the second 2, and so forth. If *g_valueByPosition* is non-nil, when you retrieve the value or default value of the field (for example, with `form -> listBox -> defValue`), you get a list of integers.

To get the field's value and default value as a list of strings even when *g_valueByPosition* is non-nil, use the [hiGetListBoxValue](#) function.

You can also set the `?valueByPosition` argument with the `valueByPosition` property. For example:

```
form -> listBox -> valueByPosition = nil
```

Default value: nil

If you set *g_valueByPosition* to nil, the items in the value and default value lists—*l_selectedItem* and *defSelectedItem*—are managed and reported as strings. When you retrieve the value or default value of the field (for example, with `form -> listBox -> value`), you get a list of strings.

`?callback` *g_callback*

Callback executed when the user has exited the list box field and the value has changed. The callback can be of string, symbol, or function object type.

`?changeCB` *g_changeCB*

Callback for this field. It can be of string, symbol, or function object type. The callback is executed when the value (selected items) of the list box has changed. Programmatic SKILL changes can also trigger this callback, for example:

```
form->listBox->value->list( "newValue")
```

Cadence User Interface SKILL Reference

Form Functions

`?numRows` *x_numRows*

Number of rows to be displayed in the list box field in a 1D form or minimum number of rows to be displayed in the list box field in a 2D form.

Whenever a horizontal scroll bar appears in the field, the number of visible rows is reduced by one.

`?multipleSelect` *g_multipleSelect*

If you specify `t`, multiple items can be selected in the list box. If you specify `nil`, only a single selection is allowed.

Default Value: `nil`

After the form is mapped, you can set this value with the `multipleSelect` property as follows:

```
form->listboxField->multipleSelect=t/nil
```

If `?multipleSelect` is set to `t` and there are duplicate items in the `?choices` list then `?valueByPosition` should be set to `t`.

`?doubleClickCB` *g_doubleClickCB*

Callback executed when the user double clicks an item in the list box. The double-clicked item is selected and all previously selected values are deselected. If the selected item has changed as a result of this double click, *t_changeCB* is also called.

Default Value: `nil` (no double click occurred)

The callback can be of string, symbol, or function object type.

`?CBOOnReselect` *g_CBOOnReselect*

Calls the change callback (*t_changeCB*) when a value is selected, even if it is the same value as before; otherwise, the change callback is called only when the value changes.

`?keepHistory` *g_keepHistory*

Cadence User Interface SKILL Reference

Form Functions

`t` or `nil`. If you specify `t`, attempts to retain the `?value` when choices are changed. The default value is `nil`.

If `g_valueByPosition` is non-`nil`, the position integers are retained instead of strings. For example, if the current value is 3 and the item list is (`"a" "b" "c" "d"`), when you change the item list to (`"c" "d" "e" "f"`), the current value remains 3 even though its corresponding string has changed from `"c"` to `"e"`.

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?help g_fieldHelp` A string or symbol used to reference help. If not specified, `s_fieldName` is used. This argument is currently not used.

`?selectionEnabled g_selectionEnabled`

Specifies whether the selection in the list box field can be changed using mouse and keyboard.

By default it is set to `t`, which means the selection can be changed.

If you set the value to `nil`, the current selection cannot be changed interactively and gets displayed as read-only mode.

Values Returned

`r_fieldHandle` Handle to the list box field.

Examples

The following sample shows you how to include a list box in a form. The user can rearrange the items of the list box by selecting the Move Up or Move Down buttons.

```
;;; The following procedures returns a copy of aList
;;; with elem swapped one position closer to the top
;;; of the list. == aList if elem is already on top
;;; of the list.
procedure( trSwapElementTowardTop( elem aList )
```


Cadence User Interface SKILL Reference

Form Functions

```
cond(
  ( !aList nil )
  ( elem == car( aList ) aList )    ;;; already the first item
  ( elem == cadr( aList )
    `( ,elem ,car( aList ) ,@caddr( aList ) )
  )
  ( t
    cons(
      car( aList )
      trSwapElementTowardTop( elem cdr( aList ) )
    )
  ) ; t
) ; cond
) ; procedure

/*
trSwapElementTowardTop( 1 nil )
trSwapElementTowardTop( 1 '( 1 ) )
trSwapElementTowardTop( 1 '( 1 2 ) )
trSwapElementTowardTop( 1 '( 2 1 ) )
trSwapElementTowardTop( 1 '( 2 1 3 4 ) )
trSwapElementTowardTop( 1 '( 2 3 4 1 ) )
trSwapElementTowardTop( 1 '( 2 1 3 4 ) )
*/

;;; The following procedure returns a copy of aList with
;;; elem swapped one position closer to the bottom of
;;; the list.==aList if elem is already on the bottom of
;;; the list.
procedure( trSwapElementTowardBottom( elem aList )
  cond(
    ( !aList nil )
    ( length( aList ) == 1 aList )
    ( elem == car( aList )
      ;
      `( ,cadr( aList ) ,elem ,@caddr( aList ) )
    )
    ( t
      cons(
        car( aList )
        trSwapElementTowardBottom( elem cdr( aList ) )
      )
    ) ; t
  ) ; cond
) ; procedure

/*
trSwapElementTowardBottom( 1 nil )
trSwapElementTowardBottom( 1 '( 1 ) )
trSwapElementTowardBottom( 1 '( 1 2 ) )
trSwapElementTowardBottom( 1 '( 2 1 ) )
trSwapElementTowardBottom( 1 '( 2 1 3 4 ) )
trSwapElementTowardBottom( 1 '( 2 3 4 1 ) )
*/

;;; The following procedure returns a list build using the
;;; elements of aList as indices into anAssocList. It looks
;;; up each element of aList in anAssocList to determine the
;;; element to include in the result.
procedure( trReorderList( anAssocList aList )
  foreach( mapcan item aList
```

Cadence User Interface SKILL Reference

Form Functions

```
list( cadr( assoc( item anAssocList ) ) )
) ; foreach
) ; procedure

;;; A list box field can only display text strings.
;;; Since trListEditor should work on any list, the
;;; code should translate back and forth from a data
;;; item X to its print representation returned by
;;; sprintf( nil "%L"X).
procedure( trListEditor_Choices( aList )
  foreach( mapcar item aList
    sprintf( nil "%L" item )
  ) ; foreach
) ; procedure

;;; The following procedure recovers the underlying items
;;; from their print representations.
procedure( trListEditor_AssocList( aList )
  foreach( mapcar item aList
    list( sprintf( nil "%L" item ) item )
  ) ; foreach
) ; foreach

procedure( trListEditor( aList )
  let( ( listBoxField buttonBoxField aListChoices theForm moveUpButton
    moveDownButton )

    listBoxField = hiCreateListBoxField(
      ?name 'trListBoxField
      ?prompt "Items"
      ?choices trListEditor_Choices( aList )
      ?numRows min( length( aList ) 10 )
    )

    ;;; creating the Move Up button
    moveUpButton = hiCreateButton(
      ?name 'trMoveUpButton
      ?buttonText "Move Up"
      ?callback "trListEditorCB( hiGetCurrentForm()
'MoveUp )"
    )

    ;;; creating the Move Down button
    moveDownButton = hiCreateButton(
      ?name 'trMoveDownButton
      ?buttonText "Move Down"
      ?callback "trListEditorCB( hiGetCurrentForm() 'MoveDown)"
    )

    ;;; creating the Delete button
    deleteButton = hiCreateButton(
      ?name 'trDeleteButton
      ?buttonText "Delete"
      ?callback "trListEditorCB( hiGetCurrentForm() 'Delete )"
    )

    ;;; creating the form
    theForm = hiCreateAppForm(
      ?name gensym( 'trListEditor )
      ?formTitle "List Editor"
```

Cadence User Interface SKILL Reference

Form Functions

```
?fields list( listBoxField moveUpButton moveDownButton deleteButton )
?callback "trListEditorCB( hiGetCurrentForm()
'Apply )"
)

theForm->trListEditor_AssocList = trListEditor_AssocList( aList )
;;; displaying the form
hiDisplayForm( theForm )
) ; let
) ; procedure

procedure( trListEditorCB( theForm theAction )
  prog( ( selectedItem choices )
    selectedItem = car( theForm->trListBoxField->value )
    when( !selectedItem
      printf( "You haven't selected an item yet\n" )
      return( nil )
    ) ; when
    choices = theForm->trListBoxField->choices
    case( theAction
      ( MoveUp
        theForm->trListBoxField->choices =
trSwapElementTowardTop(
          selectedItem      ;;; selected element
          choices           ;;; within list of items
        )
        theForm->trListBoxField->value = list(
selectedItem )
      ) ; MoveUp
      ( MoveDown
        theForm->trListBoxField->choices =
trSwapElementTowardBottom(
          selectedItem      ;;; selected element
          choices           ;;; within list of items
        )
        theForm->trListBoxField->value = list(
selectedItem )
      ) ; MoveDown
      ( Delete
        theForm->trListBoxField->choices = remove(
          selectedItem      ;;; selected element
          choices           ;;; from list of items
        ) ; remove
        ;;; no selection
      ) ; Delete
      ( Apply
        println( theForm->trListEditor_result =
trListEditor_ReorderList( theForm ) )
      ) ; Apply
      ( t
        error( "Unsupported action: %L" theAction )
      ) ; t
    ) ; case
  ) ; let
) ; procedure

procedure( trListEditor_ReorderList( theForm )
  let( ( assocList choices )
    assocList = theForm->trListEditor_AssocList
    choices = theForm->trListBoxField->choices
    trReorderList( assocList choices )
  )
```

Cadence User Interface SKILL Reference

Form Functions

```
    ) ; let  
  ) ; procedure
```

To test your list editor:

```
Z = trListEditor( '( 1 2 3 ) )
```

After the form is displayed, rearrange the list box by moving the items up or down.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateMLTextField

```
hiCreateMLTextField(  
    ?name s_fieldName  
    [ ?prompt t_prompt ]  
    [ ?value t_value ]  
    [ ?help g_fieldHelp ]  
    [ ?defValue t_defaultValue ]  
    [ ?font t_font ]  
    [ ?hasVerticalScrollbar g_hasVerticalScrollbar ]  
    [ ?hasHorizontalScrollbar g_hasHorizontalScrollbar ]  
    [ ?enableWordWrap g_enableWordWrap ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?callback g_callback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?changeCB g_changeCB ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a multiline text field entry for a form. Any input typed into this field gets converted to a single string accessible through SKILL.

To select the proper size and location of the multiline text field in a form, first select a specific size range for the field, and then use the `hiGetTextFieldFit` function to find out the pixel size it needs for the specified range.

To create a new line in a multiline text field, the user has to press either Control-Return or Shift-Return. Pressing just the Return key applies the changes indicated in the form. To move to the next tab stop, the user has to press Control-Tab. Pressing just the Tab key moves the cursor to the next field in the form.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name *s_fieldName* Handle to this field within a form.

Cadence User Interface SKILL Reference

Form Functions

<code>?prompt t_prompt</code>	String indicating the prompt for the multiline text field. The prompt is displayed on the left side of the text field.
<code>?value t_value</code>	Indicates the content of the multiline text field. The string can contain the C newline character ('\n') to indicate an advance to the next logical line. The last logical line may or may not have an ending newline. The maximum length of the string is 8191 bytes. The default value is <code>nil</code> .
<code>?help g_fieldHelp</code>	A string or symbol used to reference help. If not specified, <code>s_fieldName</code> is used. This argument is currently not used.
<code>?defValue t_defaultValue</code>	Indicates the default content of the multiline text field. The value of the text field is set to <code>defValue</code> if the user clicks the Defaults button on a form.
<code>?font t_font</code>	The font to use in the multiline text field. It must be <code>nil</code> , causing the font to be derived from the defaults. The default is determined by the user's <code>.cdsinit</code> file. If there is no font specified in the user <code>.cdsinit</code> , the fixed font is used. Using a specified font name is not supported.
<code>?hasVerticalScrollbar g_hasVerticalScrollbar</code>	Indicates if there should be a vertical scroll bar displayed for the multiline text field. The default is to have a vertical scroll bar (<code>t</code>).
<code>?hasHorizontalScrollbar g_hasHorizontalScrollbar</code>	Indicates if there should be a horizontal scroll bar displayed for this field. The default is to have a horizontal scroll bar (<code>t</code>). If set to <code>t</code> , the word wrapping feature gets disabled.
<code>?enableWordWrap g_enableWordWrap</code>	If this argument is set to <code>t</code>, and there is no horizontal scroll bar for the field, word wrapping occurs. The default is <code>nil</code>. If <code>g_hasHorizontalScrollbar</code> is set to <code>t</code>, the word wrapping feature gets disabled even if you set <code>g_enableWordWrap</code> to <code>t</code>.
<code>?editable g_editable</code>	Enables the field for editing. The default is <code>t</code> .
<code>?enabled g_enabled</code>	

Cadence User Interface SKILL Reference

Form Functions

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?callback g_callback`

SKILL callback to be run when the user exits the text field. You provide this function.

`?focusInCallback g_focusInCallback`

SKILL function to be run whenever the field comes into focus. You provide this function.

`?changeCB g_changeCB`

SKILL callback to be run when a significant change occurs, for example if the user clicks `Enter`, pastes a newline, or exits the field after making changes to the text. Programmatic SKILL changes can also trigger this callback. The default is `nil`.

`?nextField g_nextField`

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Value Returned

`r_fieldHandle` Handle to the multiple-line text field.

Examples

The following sample uses a multiline text field to display the contents of a given file.

```
procedure( trMoreFile( )
  let( ( fileNameField fileContentsField theForm )
  ;;; creating the File Name string field
```

Cadence User Interface SKILL Reference

Form Functions

```
fileNameField = hiCreateStringField(
    ?name 'trFileNameField
    ?prompt "File Name"
    ?defValue ".cshrc"
    ?callback "trMoreFileCB( hiGetCurrentForm() 'trFileNameField )"
)

;;; creating the Contents multiline text field for
;;; viewing the file
fileContentsField = hiCreateMLTextField(
    ?name 'trMLTextField
    ?prompt "Contents"
    ?defValue ""
    ?editable t ;; so users can edit the file
)

;;; creating the form
theForm = hiCreateAppForm(
    ?name gensym( 'trMoreFile )
    ?formTitle "File Browser"
    ?callback "trMoreFileCB(hiGetCurrentForm() 'apply)"
    ?buttonLayout 'ApplyCancelDef
    ?fields
        list( fileNameField fileContentsField )
)
hiDisplayForm( theForm )
) ; let
) ; procedure

procedure( trMoreFileCB( theForm actionOrField )
    case( actionOrField
        ;;; checking if the file exists
        ( trFileNameField
            if( isFile( theForm->trFileNameField->value )
                then
                    println("File exists")
                t
            else
                println("File Does Not Exist--Try Again")
                nil
            ) ;if
        )
        ( apply
            when( trMoreFileCB( theForm 'trFileNameField )
                theForm->trMLTextField->value =
                    trReadFileIntoString(
theForm->trFileNameField->value )
            ) ; when
        )
        ( t
            error( "Illegal action or unknown field: %L" actionOrField )
        )
    ) ; case
) ; procedure

;;; The following procedure returns a list of the lines
;;; in the file in the correct order
procedure( trReadFile( pathname )
    let( ( inPort nextLine allLines )
        inPort = infile( pathname )
        when( inPort = infile( pathname )
```


Cadence User Interface SKILL Reference

Form Functions

```
        while( gets( nextLine inPort )
              allLines = cons( nextLine allLines )
              ); while
        close( inPort )
      ) ; when
    reverse( allLines )
  ) ; let
) ; procedure

;;; reading the entire file into a single string
procedure( trReadFileIntoString( pathname )
  let( ( allLines )
    allLines = trReadFile( pathname )
    when( allLines apply( 'strcat allLines ) )
  ) ; let
  ) ; procedure
```

To test your file browser:

```
trMoreFile
```

After the file browser is displayed, type the name of an existing file and press Apply to view the file.

Related Topics

[Form and Field Functions](#)

[hiGetTextFieldFit](#)

[Creating Context Menus for Fields](#)

hiCreateOptionsForm

```
hiCreateOptionsForm(  
    s_formHandle  
    t_formName  
    l_fieldEntries  
    g_help  
)  
=> r_form
```

Description

Generates a form with the specified field entries, setting it to *formHandle*.

If the form is created successfully, an internal data structure for the form is returned. Otherwise, if the arguments to *hiCreateOptionsForm* are invalid, an error message is issued and *nil* is returned. Options forms may be created using the same fields as those used by standard forms.

This function is only used for enterfunctions and cannot be used independently.

Arguments

<i>s_formHandle</i>	Global SKILL symbol used to reference this form. An options form is to be used with enterFunctions only.
<i>t_formName</i>	Name that appears in the banner of the form.
<i>l_fieldEntries</i>	List of field descriptors returned from the field creation routines. You can specify any number of field entries (but there must be at least one). The fields on the form appear in a column, in the order they are specified.

Cadence User Interface SKILL Reference

Form Functions

g_help

Specify *g_help* as a single string, single symbol, list of strings, or list of symbols that need to be used to reference a *.tgf* help file.

If you specify a single string or symbol, it is used to set the `hiHelpAppName` property on the symbol that represents the form name. If you specify a list of strings or a list of symbols, the first element in the list is used to set the `hiHelpAppName` property and the second element is used to set the `hiHelpSymName` property.

(If you want to specify the help symbol only, specify a list of two strings or symbols, with `nil` as the first element of the list and the help symbol as the second element. This sets the `hiHelpSymName` property without setting the `hiHelpAppName` property.)

When a user clicks *Help* on the form, the `hiHelp` function is called with the following arguments:

appId helpTag

If the `hiHelpAppName` property has been set, it is used as the *appId*, otherwise the application ID of the current window is used as the *appId*. If the `hiHelpSymName` property has been set, it is used as the *helpTag*, otherwise the form name is used as the *helpTag*.

The *appId* and *helpTag* determine the name of the tag file that is called (*appId.tgf*) and the help tag in the file that is accessed.

You can also use the `hiHelpAppName` and `hiHelpSymName` properties directly to specify the help file and help tag that are used to reference the help. See [Online Help Mechanism](#) for more information.

Cadence User Interface SKILL Reference

Form Functions

Value Returned

r_form Handle to the form.

Additional Information

Options forms are used only when entering graphical data. Use an options form with `enterFunctions` only. The form is created with a standard banner containing the name of the form and a Help button, and selectable button at the bottom of the form. The buttons perform these functions:

- *Hide* applies all changes indicated by the form and then removes the form from the screen.
- *Cancel* calls `cancelEnterFun` and removes the form from the screen.
- *Defaults* sets all values in the form to their default values.
- *Last* cancels any changes or selections made to the form. The selections are returned to the state that existed before the form was brought up.
- *Help* opens Doc Assistant which contains related documents and additional information.

Do not use the following functions for options forms. They apply only to standard forms:

`hiSetCallbackStatus`

`hiDisplayForm`

`hiChangeFormCallback`

`hiFormDone`

`hiFormClose`

`hiFormCancel`

`hiFormUnmap`

`hiInstantiateForm`

Related Topics

Form and Field Functions

hiCreateOutputStringField

```
hiCreateOutputStringField(  
    ?name s_fieldName  
    [ ?value t_currentValue ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?help g_fieldHelp ]  
    [ ?defValue t_defaultValue ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
)  
=> r_fieldHandle
```

Description

Creates a field that displays text. You cannot type into this field.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name s_fieldName Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

?value t_currentValue

Text to be displayed in the field. This is the initial and current value of the field. To reference it, use:

`formHandle->hiFieldSym->value`

The text is displayed in the text font, which you can set in the User Preferences form or with the `hiSetFont` function.

?prompt t_fieldPrompt

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

The prompt, or label, is displayed in the label font, which you can set with the `hiSetFont` function.

Cadence User Interface SKILL Reference

Form Functions

`?help g_fieldHelp` A string or symbol used to reference help. If this argument is not specified, *s_fieldName* is used. This argument is currently not used.

`?defValue t_defaultValue`

Default value of the field. This value is equal to the initial entry in *t_currentValue* if *t_defaultValue* is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

Value Returned

`r_fieldHandle` Handle to the string field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

[hiCreateStringField](#)

hiCreatePointField

```
hiCreatePointField(  
    ?name s_fieldName  
    [ ?value l_currentValue ]  
    [ ?help g_fieldHelp ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?defValue l_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a point field entry for a form.

A point field accepts only a point as input. A valid point is represented by a SKILL list of two numbers (the x and y coordinates). These numbers can be floating-point numbers or integers.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name s_fieldName Handle to this field within a form. To reference it, use:

formHandle->hiFieldSym

?value l_currentValue

Initial and current value of this field in the correct format. May be *nil*, in which case the point field appears blank. To reference the current value, use:

formHandle->hiFieldSym->value

?help g_fieldHelp A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

Cadence User Interface SKILL Reference

Form Functions

?prompt *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?defValue *l_defaultValue*

Default value of the field (in the correct format). This value is equal to the initial entry in *currentValue* if *defaultValue* is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

?font *t_font*

Arguments may be accepted for some form fields but are not supported at this time.

?callback *g_callback*

A string, function object, or symbol type callback that is executed when the value of the field changes. The callback is called when the field has lost focus (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?modifyCallback *g_modifyCallback*

Cadence User Interface SKILL Reference

Form Functions

Do not use the *g_modifyCallback* argument with this field because it may not work as expected. Use this argument only with string fields.

SKILL function to be executed whenever any change is made in the text field, for example, by typing or pasting. The modify callback is called when a change is detected in the field but before the change is displayed.

The modify callback function is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a point. This value replaces the current value of the field.

?focusInCallback *g_focusInCallback*

SKILL function to be run whenever the field comes into focus. You provide this function.

?editable *g_editable*

Enables the field for editing.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is *nil*.

Cadence User Interface SKILL Reference

Form Functions

?nextField *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the *tabOrderIsAddOrder* argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout—left-to-right and top-to-bottom—beginning with the field at the top-left corner of the form.

Value Returned

r_fieldHandle Handle to the point field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreatePointListField

```
hiCreatePointListField(  
    ?name s_fieldName  
    [ ?value l_currentValue ]  
    [ ?help g_fieldHelp ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?defValue l_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a point list field entry for a form.

A point list field accepts only a point list as input. A valid point list is represented by one or more points (a list of two integers or floats), separated by a blank space.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to this field within a form. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?value <i>l_currentValue</i></code>	Initial and current value of this field in the correct format. May be <code>nil</code> , in which case the field appears blank. To reference the default value, use: <code>formHandle->hiFieldSym->value</code>
<code>?help <i>g_fieldHelp</i></code>	A string or symbol used to reference help. If not specified, <i>s_fieldName</i> is used. This argument is currently not used.

Cadence User Interface SKILL Reference

Form Functions

?prompt *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?defValue *l_defaultValue*

Default value of the field in the correct format. This value is equal to the initial entry in *currentValue* if *defaultValue* is not specified. To reference the default value, use:

`formHandle->hiFieldSym->defValue`

?font *t_font*

An optional font argument is accepted but not supported at this time.

?callback *g_callback*

A string, function object, or symbol type callback that is executed when the value of the field changes. The callback is called when the field has lost focus (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?modifyCallback *g_modifyCallback*

Cadence User Interface SKILL Reference

Form Functions

Do not use the *t_modifyCallback* argument with this field because it may not work as expected. Use this argument only with string fields.

SKILL function to be executed whenever any change is made in the text field, for example, by typing or pasting. The modify callback is called when a change is detected in the field but before the change is displayed.

The modify callback function is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a point list. This value replaces the current value of the field.

```
?focusInCallback g_focusInCallback
```

SKILL function to be executed whenever the field comes into focus. You provide this function.

```
?editable g_editable
```

Enables the field for editing.

```
?enabled g_enabled
```

Specifies whether the fields in the layout are enabled. The default value is *t*.

```
?invisible g_invisible
```

Specifies whether the layout and the fields within are invisible. The default value is *nil*.

Cadence User Interface SKILL Reference

Form Functions

?nextField *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the *tabOrderIsAddOrder* argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Value Returned

r_fieldHandle Handle to the point list field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateRadioField

```
hiCreateRadioField(  
    ?name s_fieldName  
    ?choices l_radioList  
    [ ?help g_fieldHelp ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?value t_currentSelection ]  
    [ ?defValue t_defaultSelection ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?callback l_radioListCallbacks ]  
)  
=> r_fieldHandle
```

Description

Creates a radio field for a form.

A radio field contains several radio items which default to being displayed in one row. Each radio item conveys on or off status or yes or no meaning. The items in a radio field are mutually exclusive—no more than one may be set at any time. Each radio item within the radio field may specify its own callback.

If the radio field is instantiated in a one-dimensional form whose width is less than the width of the radio field, the `?itemsPerRow` argument is set automatically and is based on the width of the longest `?choices` string.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

`?name s_fieldName` Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

`?choices l_radioList`

Cadence User Interface SKILL Reference

Form Functions

List of either strings or string-icon pairs or both representing the selections that can be made.

These strings or icons appear as the prompts for radio buttons.

A string-icon pair is a list of two items where the first item is a string and the second item is either a string with a valid icon name or an icon list.

?help *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?prompt *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?value *t_currentSelection*

String of the radio button representing the *on* value (defaulting to the first, if not specified). Must be a string in *l_radioList*.

?defValue *t_defaultSelection*

Optional default value of the field. This value is equal to the initial entry in *currentSelection* if *defaultSelection* is not specified. To reference the default value, use

`formHandle->hiFieldSym->defValue`

?itemsPerRow *x_itemsPerRow*

Optional integer argument specifying the number of toggle items to lay out in one row. The items are divided into as many rows as necessary, in row-major order, to accommodate this value. If *itemsPerRow* is 1, the buttons are arranged in a single column.

The toolkit imposes restrictions on the geometry allocated to these rows. Although an *itemsPerRow* value is specified, the toolkit may ignore this layout if it does not make optimum use of screen geometry.

?font *t_font*

An optional font argument is accepted but is not supported at this time.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

Cadence User Interface SKILL Reference

Form Functions

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible.
The default value is `nil`.

`?callback l_radioListCallbacks`

Optional list of user-defined SKILL procedures that are executed when the corresponding item in the *radioList* is triggered. If only one callback is specified, it is called for each radio item when it is selected. If a list is given, the items in the list can be of symbol, string, or function object type.

`?showTextWithIcon g_showTextWithIcon`

Displays both icon and text for the radio button, if *l_radioList* contains a string-icon pair. The default value of this argument is `nil`, which means that only an icon is displayed for a radio button for which string-icon pair is specified.

Value Returned

r_fieldHandle Handle to the radio field.

Examples

This example creates a form with a radio option field, `myradio`, that displays four options: item 1, item 2, item 3, item 4. Option item 2 is grayed out by using the `enabledItems` property after the form has been created.

```
(let ((myradiofield (hiCreateRadioField
    ?name 'myradio ?prompt "myradio" ?value "item 2"
    ?choices '("item 1" "item 2" "item 3" "item 4"))))
;; disable "item 2"
myradiofield->enabledItems = '(t nil t)
(hiDisplayForm (hiCreateAppForm ?name 'myform ?fields (list myradiofield)))
```

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateReportField

```
hiCreateReportField(  
    ?name s_fieldName  
    [ ?title t_title ]  
    [ ?titleAlignment g_titleAlignment ]  
    [ ?headers g_headers ]  
    [ ?choices l_choices ]  
    [ ?value g_value ]  
    [ ?defValue g_defaultValue ]  
    [ ?returnPackedValue g_returnPackedValue ]  
    [ ?help g_fieldHelp ]  
    [ ?callback g_callback ]  
    [ ?enableDeselectCB S_enableDeselectCB ]  
    [ ?doubleClickCB g_doubleClickCB ]  
    [ ?sortCallback g_sortCallback ]  
    [ ?sort l_sort ]  
    [ ?selectMode s_selectMode ]  
    [ ?invisible g_invisible ]  
    [ ?enabled g_enabled ]  
    [ ?nextField g_nextField ]  
    [ ?altRowHilight g_rowHighlight ]  
)  
=> r_fieldHandle
```

Description

Creates a report field, which you can use to display information in a complex format. For example, you can display a table with sortable columns. The items in a report field can be selected but they cannot be edited.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

<code>?name <i>s_name</i></code>	Handle to the field. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?title <i>t_title</i></code>	Report title, which is displayed above the report.
<code>?titleAlignment <i>g_titleAlignment</i></code>	

Cadence User Interface SKILL Reference

Form Functions

Alignment of the report title. Use one of the following values:

`'left`

`'center`

`'right`

The default value is `'left`.

Cadence User Interface SKILL Reference

Form Functions

?headers *g_headers*

The list of columns in the report. Specify each column as a list that contains the following elements:

```
(columnName columnWidth colNameAlignment  
type sortable comparisonFunction userData)
```

- *columnName* is the heading of the column
- *columnWidth* is the initial width of the column. The width can be changed interactively after the form is created.

If the column width is set to 0, the column gets hidden in the report.

- *colNameAlignment* is the alignment of the column heading and the data in the column: 'left, 'right, or 'center
- *type* is the type of data in the column: 'int, 'float[n], or 'string. You can specify float in the following ways:
 - 'floatn (where *n* is the number of decimal points to be displayed in the float field). For example, 'float4.
 - 'float (which defaults to 'float3)
 - 'scientificn (where *n* is an optional integer specifying the precision). For example, 'scientific4.
 - 'scientific (which defaults to 'scientific3)
- *sortable* indicates if the column is sortable. Specify *t* if you want the column to be sortable; *nil*, otherwise. The report field can sort integer, float, and string data. Clicking on the column header sorts the data in the column.
- *comparisonFunction* (optional) is the name (string) of the comparison function that is called by the sorting algorithm. The comparison function is called with two arguments that point to the elements being compared. The return value of this sorting function must be -1, 0, or 1.

Cadence User Interface SKILL Reference

Form Functions

```
comparisonFunction(row1 row2 userData)
```

where *row1* and *row2* are members of the *l_choices* list (that is, they are entire rows, not just the specific elements in the columns).

If *userData* is *nil*, *comparisonFunction* is called as described in *comparisonFunction* above.

You need to specify *type*, *sortable*, *compareFunction*, and *userData* only if you want the column to be sortable.

Once the report field is instantiated, you can modify the headers (specifically, the name, width, and alignment of the columns) with the *headers* property by using the following commands:

```
form->reportField->headers=newHeaders
```

```
or (putprop form->reportField newHeaders  
'headers)
```

where *newHeaders* is a list in the same format as the *g_headers* argument described above except that only *columnName*, *columnWidth*, and *columnNameAlignment* values are used (the remaining values, if specified, are ignored). Also, any *nil* elements in the *newHeaders* list are ignored and do not replace the corresponding elements in the existing headers list.

For example, if the old headers list is:

```
'(("one" 100 left) ("two" 200 right float)  
("three" 300 center))
```

and the new headers list is:

```
'(nil ("TWO" nil left int))
```

then the new headers:

```
'(("one" 100 left) ("TWO" 200 left float)  
("three" 300 center))
```

Cadence User Interface SKILL Reference

Form Functions

?choices *l_choices*

List containing the items in the report. Each item forms a row of the report. For example:

```
?choices list(  
list(1 "ABC" "buy" 62.0 "Buy & Hold")  
list(2 "DEF" "sell" 17.0 "Sell")  
list(3 "XYZ" "hold" 9.5 "Hold")  
)
```

To retrieve or change the value of *l_choices*, use:

```
formHandle->hiFieldSym->choices
```

Cadence User Interface SKILL Reference

Form Functions

?value *g_value*

Initial value of the field, indicating which items in the report are selected when the form first appears.

Specify *g_value* as a list of integers in the ascending order, in which each integer represents an item in ?choices. The first item in ?choices is 0, the second item is 1, the third item is 2, and so on. You can specify a range of items as a list within the list.

For example:

?value '(1 3 6);selects items 1, 3, 6

?value '(2 4); selects items 2, 3, 4

?value '(1 3) 9 11); selects items 1, 2, 3, 9, 11

After the form is instantiated, you can use the `value` property to retrieve or change the list of currently selected items. To retrieve the list of selected items, use the following syntax:

```
formHandle->hiFieldSym->value
```

The `value` property returns a list of integers. This list contains packed values if the *g_returnPackedValue* argument is `t`.

To change the list of selected items with the `value` property, use the following syntax:

```
formHandle->hiFieldSym->value = listOfItems
```

Setting the value through the `value` property always triggers the callback of the report field. If you want to set the value without triggering the callback, use the functions

`hiReportSelectItem`, `hiReportSelectItems`, or `hiReportSelectAllItems`, with the *g_notify* argument set to `nil`.

Cadence User Interface SKILL Reference

Form Functions

`?defValue` *g_defaultValue*

Default value of the field, indicating which items in the report are selected when the *Defaults* button is clicked. After the form is instantiated, you can use the `defValue` property to retrieve or change the default value.

Specify *g_defaultValue* as a list of integers in the ascending order, in which each integer represents an item in `?choices`. The first item in `?choices` is 0, the second item is 1, the third item is 2, and so on. You can specify a range of items as a list within the list.

For example:

```
?defValue '(1 3 6)      ; selects items 1, 3, 6
?defValue '(2 4)        ; selects items 2, 3, 4
?defValue '(1 3) 9 11)  ; selects items 1, 2, 3, 9, 11
```

Use the following syntax for the `defValue` property

```
formHandle->hiFieldSym->defValue
```

The `defValue` property returns a list of integers. This list contains packed values if the *g_returnPackedValue* argument is `t`.

`?returnPackedValue` *g_returnPackedValue*

Return packed values for `value`, `defValue`, and `lastValue` properties to return packed values. *g_returnPackedValue* also determines whether the second argument passed to *S_callback* contains packed values.

A packed value is a range of items specified as a list within the list. For example, if items 1, 2, 3, 4, 5, 9, and 11 are selected in the report, the `value` property returns the following:

```
'(1 5) 9 11)  ; if g_returnPackedValue is t
'(1 2 3 4 5 9 11) ; if g_returnPackedValue is
nil
```

`?help` *g_fieldHelp*

A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

Cadence User Interface SKILL Reference

Form Functions

?callback *g_callback*

A string, a symbol, or a function object that specifies the name of the callback function to be called when items are selected. If you want the callback to also be called when items are deselected, specify *t* as the value of the ?enableDeselectCB argument.

If you specify a symbol for this argument, or a non-empty string without space or parentheses, it is assumed to be the callback function name. The callback function is passed two arguments. The first argument is the symbol representing the field name. The second argument is a list of integers representing the selected items of the report (the first item is 0, the second item is 1, and so forth). The second argument contains packed values if the value of the *g_returnPackedValue* argument is *t*.

If you specify a string with space or parentheses for this argument, it is assumed to be a SKILL expression and is passed to SKILL to be evaluated.

For example:

```
procedure( reportCB(fieldName selectedItems "sl")
    printf( "Report field callback, fieldName = %L,
selectedItems = %L"                fieldName
selectedItems )
reportField = hiCreateReportField(
?name      'reportField
?title     "Report Field"
?selectMode 'single
?headers   ' (("Property"    120 'left)
("Operator"    70 'left)
("Value"       120 'left))
?choices   ' (("test"  "==" "argh")
("test2"  ">" "argh")
("test3"  "<" "test4"))
?callback   'reportCB
)
```

Cadence User Interface SKILL Reference

Form Functions

`reportField` can also take a single click `?callback` called `hiClickedCallback`, which enables you to retrieve selected column, row or symbol information for a mouse click selection. For example:

```
(defun myClickedCallback (fieldSymbol row column
"xxx") ...)
rptProps->hiClickedCallback='myClickedCallback
```

If specified, clicking on a `reportField` item cause `hiClickedCallback` to be applied. The callback then is passed three arguments; `fieldSymbol`, row number and column number, where both row and column numbers are zero-based. Here `hiClickedCallback` is called before `hiReportSelectItems` is executed.

`?enableDeselectCB` *`S_enableDeselectCB`*

`t` or `nil`. Specify `t` if you want the callback procedure (specified with the `?callback` argument) to be called when items are deselected, in addition to being called when items are selected. If you specify `nil`, the callback procedure is called only when items are selected.

The default value of *`S_enableDeselectCB`* is `nil`.

Cadence User Interface SKILL Reference

Form Functions

?doubleClickCB *g_doubleClickCB*

A string, symbol or function object specifying the callback procedure to be called when an item is double-clicked.

If specified as a symbol, the procedure gets passed two arguments. The first argument is a string representing the report field name; the second argument is an integer representing the item that was double-clicked (the first item is 0, the second item is 1, and so forth).

If specified as a string, the string is evaluated and the callback function is not passed any arguments.

The default value of this argument is an empty string (" "), which means that a double-click callback has not been specified.

If specified as a function object, it is passed following arguments:

```
< double click callback > ( o_field g_scope  
x_item_index )
```

where *o_field* is the report field, *g_scope* specifies where the report is instantiated, and *x_item_index* is the index of the item being double-clicked.

Cadence User Interface SKILL Reference

Form Functions

?sortCallback *g_sortCallback*

A string, symbol, or function object type callback that is used to notify when the field is sorted. The sort callback procedure is passed two arguments. The first argument is the field name symbol; the second argument is `list(columnToSort sortDirection)`, where *columnToSort* is an integer indicating the column to sort (the first column is 0, the second is 1, and so on) and *sortDirection* is `t` or `nil` (`t` indicates descending order, `nil` indicates ascending order). The sorting is then done by the `comparisonFunction`, which is passed as part of the column description in the `?headers` argument.

If callback is of function object type, it is passed following arguments:

```
< sort callback > ( o_field g_scope  
l_sort_property )
```

, where *l_sort_property* is a list of `(columnToSort sortDirection)`. Same as that used to sort callback of string or symbol type.

?sort *l_sort*

Indicates which column is sorted when the form first appears. Specify *l_sort* as a list of two elements:

```
'(columnToSort sortDirection)
```

where *columnToSort* is an integer indicating the column to sort (the first column is 0, the second is 1, and so on) and *sortDirection* is `t` or `nil` (`t` indicates descending order, `nil` indicates ascending order).

You can also use the `sort` property to sort a column after the form is instantiated.

Use the following format for the `sort` property:

```
formHandle->hiFieldSym->sort
```

Clicking on the column header sorts the column. Clicking again on the column header changes the sorting direction, from ascending to descending or vice versa.

?selectMode *s_selectMode*

Cadence User Interface SKILL Reference

Form Functions

Specifies the selection mode for the report field. Use one of the following symbols:

`'single`: Only one item can be selected at a time. Click to select the item. To deselect an item, click it again or select another item.

`'multiple`: Multiple items can be selected. Click to select the items. To deselect an item, click it again.

`'browse`: One item always remains selected. Click or drag to select the item. To deselect an item, select another item.

`'extended`: One or more items always remain selected. Multiple contiguous ranges of items can be selected. Click to select a single item. Click and drag to select a range of items. Press the `CTRL` key and click to select an additional item or a range of additional items. To deselect an item, press the `CTRL` key and click the item. `CTRL+click` also deselects the last selected item. You can also use `CTRL+SHIFT` to select or deselect a range of items.

`'none`: Disables the selection, which means no item can be selected in the report field. It displays the report field in a read-only mode.

Cadence User Interface SKILL Reference

Form Functions

The default mode is 'extended.

You can modify the value of this argument with the `selectMode` property. For example:

```
formHandle->fieldSym->selectMode = 'single
```

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?enabled g_enabled`

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?nextField g_nextField`

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the `tabOrderIsAddOrder` argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the `tabOrderIsAddOrder` argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

`?altRowHilight g_rowHighlight`

Specifies whether background of alternate rows in the table should be highlighted. The default is `nil`, which means that highlight is not enabled.

Values Returned

`r_fieldHandle` Handle to the report field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateScrollRegion

hiReportSelectItem

hiReportSelectItems

hiReportSelectAllItems

hiReportDeselectItem

hiReportDeselectItems

hiReportDeselectAllItems

hiReportGetSelectedItems

hiCreateScaleField

```
hiCreateScaleField(  
    ?name s_fieldName  
    [ ?prompt t_prompt ]  
    [ ?value x_currentValue ]  
    [ ?callback g_callback ]  
    [ ?isContinuous g_isContinuous ]  
    [ ?range l_range ]  
    [ ?defValue x_defaultValue ]  
    [ ?help g_fieldHelp ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?precision x_precision ]  
)  
=> r_fieldHandle
```

Description

Creates a scale field for a form.

A scale field indicates a value from within a range of numerical values and allows the user to input or modify a value from the same range. The field consists of a rectangular region containing a slider indicating the current value. It also includes a label indicating the scale value at any position of the slider.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to this field within a form. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?prompt <i>t_prompt</i></code>	String specifying the prompt for the field. To reference it, use: <code>formHandle->hiFieldSym->prompt</code>
<code>?value <i>x_currentValue</i></code>	

Cadence User Interface SKILL Reference

Form Functions

Initial and current value of the scale field which must be within the given range:

```
formHandle->hiFieldSym->value
```

?callback *g_callback*

A string, function object, or symbol type callback. Called when the slider is released in a new position or when it moves incrementally along the bar, depending on the value of the *g_isContinuous* argument. If *g_isContinuous* is not specified, the callback is executed when the slider is released in the new position.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

?isContinuous *g_isContinuous*

t or *nil*. When *g_isContinuous* is *t*, the callback is continuous while the slider is being moved; when *g_isContinuous* is *nil*, the callback is executed when the slider is released in the new position. The default value is *nil*. You can modify *g_isContinuous* at any time with the following property:

```
formHandle->hiFieldSym->isContinuous = t|nil
```

?range *l_range*

Indicates the upper and lower values the scale field may obtain. The default range is from 0 to 1000, inclusive. To change the range the slider displays, the user may type

```
form->scalefield->range = newrange
```

where *newrange* is a list of two numbers. These numbers can include negative integers.

If the maximum value for the range is not an integer, it is set to 1000. If the minimum value for the range is not an integer or is equal to or greater than the maximum value, it is set to the maximum value minus 1000 (*maxValue* - 1000).

If the range is changed, *currentValue* and *defaultValue* are modified, if necessary, to fall within the new range.

?defValue *x_defaultValue*

Cadence User Interface SKILL Reference

Form Functions

Default value of the field (in the correct format). This value is equal to the initial entry in *currentValue* if *defaultValue* is not specified. To reference it, use:

`formHandle->hiFieldSym->defValue`

?help *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?font *t_font* An optional font argument is accepted but is not supported at this time.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is `t`.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

?precision *x_precision*

Integer specifying the number of decimal points to display. The default value of this argument is 2.

Value Returned

r_fieldHandle Handle to the scale field.

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

hiCreateSeparatorField

```
hiCreateSeparatorField(  
    ?name s_fieldName  
    [ ?invisible g_invisible ]  
)  
=> r_fieldHandle
```

Description

Creates a separator field for a form.

A separator is a line used to separate fields for easier visual recognition. It is always two pixels thick. Only horizontal separators can be created for one-dimensional forms. Two-dimensional forms can have vertical or horizontal separators. A horizontal separator in a 1D form always spans the width of the form.

When specifying 2D attributes for a separator field use the following format:

```
list( r_separatorField x_xpos:x_ypos x_length:x_orient )
```

x_xpos:*x_ypos* represents the origin of the separator. The length can be controlled by *x_length* but the thickness is always two pixels. *x_orient* is 0 if the separator is horizontal, 1 if vertical.

Arguments

?name *s_fieldName* Handle to this field within a form.

?invisible *g_invisible*

t or *nil*. Specify *t* if you want the field to be invisible; specify *nil* if you want the field to be visible. The default value is *nil*.

You can modify the value of this argument (after the form is created) with the *invisible* property. For example:

```
formHandle->fieldSym->invisible = nil
```

Cadence User Interface SKILL Reference

Form Functions

Value Returned

r_fieldHandle Handle for the separator field.

Examples

An example of specifying a horizontal separator 40 pixels long starting at 10:30 is:

```
list( hiCreateSeparatorField progId ( ?name 'sep ) 10:30 40:0 )
```

Related Topics

[Form and Field Functions](#)

hiCreateSimpleHypertextField

```
hiCreateSimpleHypertextField(  
    s_name  
    t_text  
    t_link  
    [ g_newWindow ]  
)  
=> r_fieldHandle
```

Description

Creates a simple one-line hypertext field, which displays a hyperlink to a Web page. See `hiCreateHypertextField()` for information on how to create a more complex hypertext field.

Links to a Web page is displayed in a Web browser. If a browser is already open, it is used, otherwise, the browser specified in the *Web Browser* field of the CIW User Preferences form is used. The default browser is Netscape.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

<i>s_name</i>	Handle to the field. To reference it, use <code>formHandle->hiFieldSym</code>
<i>t_text</i>	The text that you want to display.
<i>t_link</i>	The URL associated with <i>t_text</i> , specified as a string.
<i>g_newWindow</i>	<code>t</code> or <code>nil</code> . Specify <code>t</code> if you want the Web page to be displayed in a new browser window when a user clicks the hyperlink. The default value is <code>nil</code> .

Values Returned

<i>r_fieldHandle</i>	Handle to the simple hypertext field.

Cadence User Interface SKILL Reference

Form Functions

Example

```
hiCreateSimpleHypertextField( 'mySimpleField "Cadence Design Systems" "http://  
www.cadence.com" )
```

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

[hiCreateHypertextField](#)

[hiHypertextReadin](#)

[hiHypertextReplace](#)

[hiHypertextScroll](#)

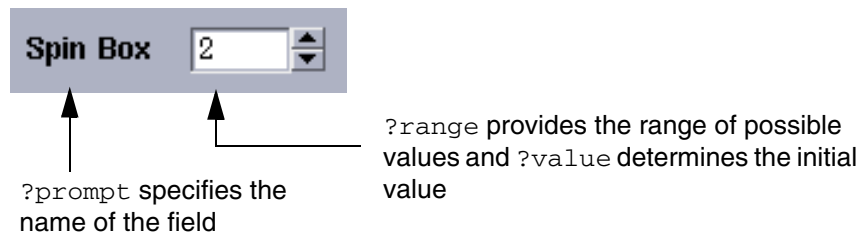
[hiLaunchBrowser](#)

hiCreateSpinBox

```
hiCreateSpinBox(  
    ?name s_fieldName  
    [ ?prompt t_prompt ]  
    [ ?help g_fieldHelp ]  
    [ ?value g_value ]  
    [ ?defValue g_defaultValue ]  
    [ ?font t_fontName ]  
    [ ?callback g_callback ]  
    [ ?range l_rangeOfItems ]  
    [ ?arrowOrientation g_arrowOrientation ]  
    [ ?shadowed g_shadowed ]  
    [ ?enabled g_enabled ]  
    [ ?editable g_editable ]  
    [ ?obeyWidth g_obeyWidth ]  
    [ ?invisible g_invisible ]  
    [ ?wrapAround g_wraparound ]  
    [ ?justification s_justification ]  
)  
=> r_fieldHandle
```

Description

Creates a spin box field, which contains a range of items that users can cycle through. The spin box field can only contain integers. For floating point numbers, use `hiCreateFloatSpinBox`.



This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Cadence User Interface SKILL Reference

Form Functions

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to the spin box field. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?prompt <i>t_prompt</i></code>	The name that appears next to the spin box field, specified as a string.
<code>?help <i>g_fieldHelp</i></code>	A string, symbol, list of strings, or list of symbols used to reference help. If this argument is not specified, <i>s_name</i> is used. This argument is not currently used.
<code>value <i>g_value</i></code>	The initial and current value of the field. <i>g_value</i> must be an integer within the range specified by <i>l_rangeOfItems</i> .
<code>?defValue <i>g_defaultValue</i></code>	The default value of the field. <i>g_defaultValue</i> must be an integer within the range specified by <i>l_rangeOfItems</i> .
<code>?font <i>t_fontName</i></code>	This optional argument is accepted but not supported currently.
<code>?callback <i>g_callback</i></code>	A string, function object, or symbol type callback. If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.
<code>?range <i>l_rangeOfItems</i></code>	A list of two integers defining the range of values in the spin box field. The first integer must be less than the second integer. For example: <code>?range list(-12 12)</code>
<code>?arrowOrientation <i>g_arrowOrientation</i></code>	This argument is now ignored. Arrows that are used to cycle through the spin box field are always vertical and are always displayed on the right side of the field.
<code>?shadowed <i>g_shadowed</i></code>	This argument is now ignored. There is always a shadow box around the arrows in the spin box field.
<code>?enabled <i>g_enabled</i></code>	

Cadence User Interface SKILL Reference

Form Functions

Specifies whether the fields in the layout are enabled. The default value is `t`.

`?editable g_editable`

`t` or `nil`. Specify `t` if you want the field to be editable; specify `nil` if you want it to be read-only.

`?obeyWidth g_obeyWidth`

`t` or `nil`. Specify `t` if you want the spinbox field to follow the width specified in the 2D attributes when the form is created. The default value is `nil`, which means the width specified in the 2D attributes is ignored.

For information on 2D attributes, see [Two Dimensional Form Layout](#).

`?invisible g_invisible`

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

`?wrapAround g_wraparound`

Increments or decrements the spin box field as normal, but stop at the max/min values respectively. The default value is `t`.

`?justification s_justification`

Specifies one of the following symbols for the value added to the spinbox field: `'left`, `'center`, or `'right`. The default value is `'left`.

Values Returned

`r_fieldHandle` Handle to the spin box field.

Examples

```
spinField = hiCreateSpinBox(  
    ?name 'spinField  
    ?prompt "Spin box: "  
    ?arrowOrientation 'vertical  
    ?range '(1 100)  
    ?value 3  
    ?defValue 100  
    ?callback "callbackOne"  
    ?wrapAround nil  
)
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[Creating Context Menus for Fields](#)

[hiCreateFloatSpinBox](#)

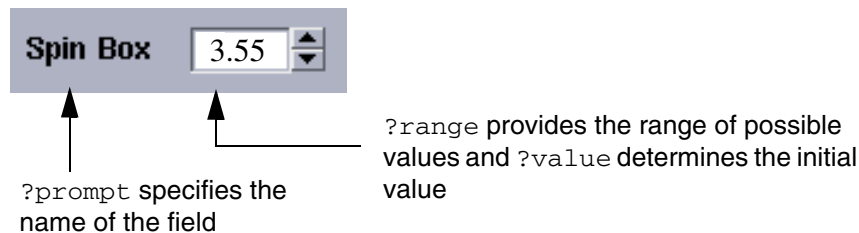
hiCreateFloatSpinBox

```
hiCreateFloatSpinBox(  
    ?name s_fieldName  
    [ ?prompt t_prompt ]  
    [ ?help g_fieldHelp ]  
    [ ?value g_value ]  
    [ ?defValue g_defaultValue ]  
    [ ?font t_fontName ]  
    [ ?callback g_callback ]  
    [ ?range l_rangeOfItems ]  
    [ ?shadowed g_shadowed ]  
    [ ?enabled g_enabled ]  
    [ ?editable g_editable ]  
    [ ?precision x_precision ]  
    [ ?step f_step ]  
    [ ?obeyWidth g_obeyWidth ]  
    [ ?invisible g_invisible ]  
    [ ?wrapAround g_wraparound ]  
    [ ?arrowOrientation g_arrowOrientation ]  
)  
=> r_fieldHandle
```

Description

Creates a float spin box field, which contains a range of items that users can cycle through. The spin box field can only contain floating point numbers. For integers, use

`hiCreateSpinBox`.



This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Cadence User Interface SKILL Reference

Form Functions

Arguments

<code>?name <i>s_fieldName</i></code>	Handle to the spin box field. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?prompt <i>t_prompt</i></code>	The name that appears next to the spin box field, specified as a string.
<code>?help <i>g_fieldHelp</i></code>	A string, symbol, list of strings, or list of symbols used to reference help. If this argument is not specified, <i>s_name</i> is used. This argument is not currently used.
<code>?value <i>g_value</i></code>	The initial and current value of the field. <i>g_value</i> must be a floating point number within the range specified by <i>l_rangeOfItems</i> .
<code>?defValue <i>g_defaultValue</i></code>	The default value of the field. <i>g_defaultValue</i> must be a floating point number within the range specified by <i>l_rangeOfItems</i> .
<code>?font <i>t_fontName</i></code>	This optional argument is accepted but not supported currently.
<code>?callback <i>g_callback</i></code>	A string, symbol, or function object type callback. If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.
<code>?range <i>l_rangeOfItems</i></code>	A list of two floating point numbers defining the range of values in the spin box field. The first number must be less than the second number. For example: <code>?range list(-12.0 12.5)</code>
<code>?shadowed <i>g_shadowed</i></code>	This argument is ignored. There is always a shadow box around the arrows in the spin box field.
<code>?enabled <i>g_enabled</i></code>	Specifies whether the fields in the layout are enabled. The default value is <code>t</code> .

Cadence User Interface SKILL Reference

Form Functions

?editable *g_editable*

Enables the field for editing.

?precision *x_precision*

Integer specifying the number of decimal points to display. The default value of this argument is 2.

?step *f_step*

The floating point number by which to increment or decrement when the up or down arrow is pressed. The default value of this argument is 1.0.

?obeyWidth *g_obeyWidth*

Enables the spinbox field to follow the width specified in the 2D attributes when the form is created. The default value is `nil`, which means the width specified in the 2D attributes is ignored.

For information on 2D attributes, see [Two Dimensional Form Layout](#).

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

?wrapAround *g_wraparound*

Increments or decrements the spin box field as normal, but stop at the max/min values respectively. The default value is `t`.

?arrowOrientation *g_arrowOrientation*

This argument is now ignored. Arrows that are used to cycle through the spin box field are always vertical and are always displayed on the right side of the field.

Value Returned

r_fieldHandle

Handle to the float spin box field.

Related Topics

[Creating Context Menus for Fields](#)

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateSpinBox

hiCreateStringField

```
hiCreateStringField(  
    ?name s_fieldName  
    [ ?prompt t_fieldPrompt ]  
    [ ?value t_currentValue ]  
    [ ?help g_fieldHelp ]  
    [ ?defValue t_defaultValue ]  
    [ ?font t_font ]  
    [ ?callback g_callback ]  
    [ ?modifyCallback g_modifyCallback ]  
    [ ?focusInCallback g_focusInCallback ]  
    [ ?format t_fieldFormat ]  
    [ ?editable g_editable ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?nextField g_nextField ]  
)  
=> r_fieldHandle
```

Description

Creates a string field entry for a form. Any input typed into this field is surrounded by double quotation marks before being passed to SKILL to interpret.

A string field is a field allowing for typed input. Up to three methods of encouraging the desired format of the typed input may be specified: An initial string can indicate the format of the string to be entered, the prompt can convey the type of response desired, or a format string can be specified which is automatically imposed on any user input.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Cadence User Interface SKILL Reference

Form Functions

Arguments

?name *s_fieldName* Handle to this field within a form. To reference it, use:

`formHandle->hiFieldSym`

?prompt *t_fieldPrompt*

String specifying the prompt for the field. To reference it, use:

`formHandle->hiFieldSym->prompt`

?value *t_currentValue*

Initial and current value of this field. To reference it, use:

`formHandle->hiFieldSym->value`

?help *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

?defValue *t_defaultValue*

Optional, default value of the field. This value is set to the initial entry in ?value if this argument is not specified. To reference the value, use:

`formHandle->hiFieldSym->defValue`

?font *t_font* An optional font argument is accepted but is not supported at this time.

?callback *g_callback*

A string, function object, or symbol type callback. For type-in fields, the callback is called when the field has “lost focus” (when the cursor leaves the form, the user tabs to another field, and so forth) or when the user selects any button in the form. If you want a callback that is called when the field comes into focus, use the *g_focusInCallback* argument.

If the callback is of string type, you can specify one or more SKILL functions or even multiple SKILL statements that should be executed when the value of the field changes.

Cadence User Interface SKILL Reference

Form Functions

?modifyCallback *g_modifyCallback*

SKILL function to be executed whenever any change is made in the text field, for example, by typing a value. The modify callback is called when a change is detected in the field but before the change is displayed. The callback can be of string, symbol, or function object type.

A modify callback of symbol or function object type is passed following arguments:

```
<field modify callback>( o_field g_scope  
t_latestTextValue g_sourceOfChange)
```

, where *o_field* is the field whose modify callback is triggered, *g_scope* is where the field is instantiated, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil*, *t* indicates that the change was made through the program (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback of string type is passed the following arguments:

```
s_fieldName t_latestTextValue  
g_sourceOfChange
```

where *s_fieldName* is the symbol of the field in which the change was made, *t_latestTextValue* is the latest text value of the field, and *g_sourceOfChange* is *t* or *nil* where *t* indicates that the change was made programmatically (by changing the value of the field) and *nil* indicates that the change was made by a user action.

The modify callback function should return the following:

```
t | nil | value
```

If the modify callback function returns *t*, the changes made to the field are allowed and are displayed as they are entered. If the function returns *nil*, the changes made to the field are not allowed and are not displayed—the original value of the field is retained. If the function returns some other value, it must be a string. This value replaces the current value of the field.

Cadence User Interface SKILL Reference

Form Functions

?focusInCallback *g_focusInCallback*

SKILL function to be executed whenever the field comes into focus.

?format *t_fieldFormat*

Optional string that can control the display of the value on the form. It is similar to `printf`.

Acceptable format characters are those allowed by the SKILL programming language.

?editable *g_editable*

Enables the field for editing.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is `t`.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

?nextField *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the *tabOrderIsAddOrder* argument of `hiCreateAppForm` or `hiCreateScrollRegion`. If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout — left-to-right and top-to-bottom— beginning with the field at the top-left corner of the form.

Value Returned

r_fieldHandle Handle to the string field.

Examples

Creates string entry for the form.

```
;*****
```

Cadence User Interface SKILL Reference

Form Functions

```
;Sample form and fields
;*****
; Create string entry for the form.
procedure( myformDefineFields()
  strfldglb = hiCreateStringField(
    ?name `strfldglb
    ?prompt "sample string field"      ;;; FIELD1, a string type
    ?defValue "initial string"
    ?callback "my_strfldglb_cb()"
    ?editable t
  )

  ; Create a Boolean button for the form.
  boolfldglb = hiCreateBooleanButton(
    ?name `boolfldglb
    ?buttonText "sample boolean field"      ;;; FIELD2, a boolean type
    ?defValue t
    ?callback "my_boolfldglb_cb()"
  )
  list( strfldglb boolfldglb )
)

; Generate the form

procedure( myformcreateForm()
  if( !boundp(`myformglb) || (myformglb == nil)
    then
      myformglb = hiCreateAppForm(
        ?name `myformglb
        ?fields myformDefineFields()
        ?formTitle "My first Form"
        ?callback "myformglb_cb()"
        ?buttonLayout `OKCancel
      )
  )
  myformglb
)

; Display the form
procedure( myformdisplayform()
  myformcreateForm()      ;;; Creation and Display are separate
  hiDisplayForm(`myformglb)
```

Cadence User Interface SKILL Reference

Form Functions

```
)  
; Check the current state of the form  
hiIsFormDisplayed(myformglb) =>nil  
  
myformdisplayform()  
hiIsFormDisplayed(myformglb)  
=>t
```

The following sample displays different types of fields in a form.

```
;;; creating the File Name field  
trFileNameField = hiCreateStringField(  
    ?name      'trFileNameField  
    ?prompt    "File Name"  
    ?defValue  ".cshrc"  
    ?callback  "trDoFieldCheckCB( hiGetCurrentForm() )"  
    ?editable  t  
)  
  
;;; checking if the file exists  
procedure( trDoFieldCheckCB( theForm )  
    if( isFile( theForm->trFileNameField->value )  
        then  
            println("File exists")  
            t  
        else  
            println("File Does Not Exist--Try Again")  
            nil  
        ) ;if  
    ) ; procedure  
  
;;; creating the form  
trFileForm = hiCreateAppForm(  
    ?name      'trFileForm  
    ?formTitle "File Form"  
    ?callback  'trFileFormCB  
    ?fields    list( trFileNameField )  
    ?unmapAfterCB t  
)  
  
procedure( trFileFormCB( theForm )  
    if( trDoFieldCheckCB( theForm )  
        then  
            hiSetCallbackStatus( theForm t )  
            hiHighlightField( theForm 'trFileNameField 'background )  
            view( theForm->trFileNameField->value )  
        else  
            hiSetCallbackStatus( theForm nil )  
            hiHighlightField( theForm 'trFileNameField 'highlight )  
        ) ; if  
    ) ; procedure  
  
;;; displaying the form  
hiDisplayForm( trFileForm )
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[hiCreateOutputStringField](#)

[Form and Field Functions](#)

hiCreateTabField

```
hiCreateTabField(  
    ?name s_name  
    ?fields l_fieldEntries  
    [ ?attachmentList l_fieldAttachments ]  
    [ ?tabs g_tabs ]  
    [ ?pageBase g_pageBase ]  
    [ ?value g_currentValue ]  
    [ ?defValue g_defaultValue ]  
    [ ?tabPlacement g_tabPlacement ]  
    [ ?tabOrderIsAddOrder g_tabOrderIsAddOrder ]  
    [ ?callback g_callback ]  
    [ ?pageScroller g_pageScroller ]  
    [ ?notebookStyle g_notebookStyle ]  
    [ ?help g_fieldHelp ]  
    [ ?invisible g_invisible ]  
    [ ?tabToolTips l_tooltip ]  
    [ ?tabLabels l_tabLabel ]  
    [ ?scrollPage l_scrollPage ]  
)  
=> r_fieldHandle
```

Description

Creates a tab field, which can have multiple pages. You can change the pages interactively by selecting the tab attached to each page, or programmatically by using the `form->field->value` property.

Tab fields do not have scroll bars. If you want scroll bars on a tab page, place a scroll region on that page. You can do so only if the fields in the tab field do not have attachments and if the tab field itself is not an attached field, that is, a field attachment is not specified for it when the form is created. For information about scroll regions, see [hiCreateScrollRegion](#).

You can enable or disable (gray out) either a specific page of the tab field or the entire tab field with the `enabled` property.

To enable or disable the entire field, use the following:

```
form->tabField->enabled = t | nil
```

To enable or disable a specific page of the tab field, use the following:

```
form->tabField->page->enabled = t | nil
```

The status of the fields within the tab field or tab page does not get updated with these settings. If you query a field within a tab field with `form->tabfield->field->enabled`,

Cadence User Interface SKILL Reference

Form Functions

the field reports its own enabled/disabled status, regardless of whether it is in an enabled or disabled tab field.

Also, you cannot enable a field that is within a disabled tab field or disabled tab page. If you specify *form->tabField->field->enabled=t*, the field does not get enabled until the tab field itself becomes enabled.

A tab field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

You can create a tooltip for this field; see [Creating Tool Tips for Fields](#).

Arguments

?name *s_name* Handle to the tab field within a form. To reference it, use:

formHandle->hiFieldSym

?fields *l_fieldEntries*

Cadence User Interface SKILL Reference

Form Functions

A list of 2-D lists. Each 2-D list forms a page of the tab field. The number and order of elements in the *l_fieldEntries* argument determine the number and order of pages in the tab field. The default value of this argument is *nil*.

For more information about 2-D lists, see [Two Dimensional Form Layout](#).

You can reference a tab page by its base name (which lets you reference the page by position) or with a combination of the base name and tab label (which lets you reference the page by name). Use the following command:

```
formHandle->hiFieldSym->tabPage
```

where *tabPage* is either *pageBaseName#* or *pageBaseName_tabLabel*. (See the *g_pageBase* argument for more information about the base name and the *g_tabs* argument for more information about the tab label.)

For example, if you have the following tab field in a form *myForm*:

```
hiCreateTabField(?name 'myTabField ... ?tabs list("abc" "def" "ghi")...)
```

and the default page base name, *page*, is used, then you can reference the second tab in one of the following ways:

```
myForm->myTabField->page2  
myForm->myTabField->page_def
```

Referencing the tab page by tab name (the second option listed above) allows you to insert additional tab pages easily without affecting existing code.

Any whitespace in the *tabLabel* name is converted to underscore and leading or trailing whitespace is removed.

```
?attachmentList l_fieldAttachments
```

Determines how each page of the tab field attaches to the tab pane. For more information about field attachments, see [Field Attachments](#).

Cadence User Interface SKILL Reference

Form Functions

?tabs *g_tabs*

List of strings representing the tab labels. *nil* and " " (empty string) are also legal values. If you specify " ", the page has an empty tab. If you specify *nil*, the page has a tab but whether the page is enabled or disabled depends on the value of the *g_pageScroller* argument—if *g_pageScroller* is *t*, the page is enabled and its tab displays the page number; if *g_pageScroller* is *nil*, the page is disabled and its tab is blank.

The default value of this argument is *nil*.

?pageBase *g_pageBase*

A string or symbol that defines the base name for each page of the tab field. The default base name for each page is "page".

The first page of the tab field is named *pageBaseName1*, the second page is named *pageBaseName2*, and so on, up to the last page of the field. If you do not provide a base name, the default names of the pages are *page1*, *page2*, *page3*, and so on.

This argument is necessary if you create a tab field within a tab field as you need to distinguish the pages of one from the other.

?value *g_currentValue*

Initial and current value of the field. Determines which page is displayed when the tab field first appears. The legal values for this argument are 1 to *number_of_pages*, where 1 indicates the first page. To change the value, use:

```
form->field->value = x
```

where *x* is any number between 1 and *number_of_pages*.

?defValue *g_defaultValue*

Default value of the field. Determines which page of the tab field is displayed as the default. The legal values for this argument are 1 to *number_of_pages*, where 1 indicates the first page. To change the value, use:

```
form->field->defValue = x
```

where *x* is any number between 1 and *number_of_pages*.

?tabPlacement *g_tabPlacement*

Cadence User Interface SKILL Reference

Form Functions

Location of the tabs on the field. Specify one of the following:

`'top`

`'bottom`

The default value of this argument is `'bottom`.

`'left` and `'right` are no longer supported. For any existing `'left` or `'right` settings, tabs are placed at the top of the field.

`?tabOrderIsAddOrder` *g_tabOrderIsAddOrder*

Determines the order in which pages within the tab field are traversed when the `Tab` key is pressed. If you specify `t`, the *nextField* property is automatically set for the fields and the traversal order is the order in which fields are added to the form (that is, the order in which they are listed in the *l_fieldEntries* list). If you specify `nil`, the default traversal order is based on the form layout—left-to-right and top-to-bottom—beginning with the field at the top-left corner of the form.

The default value of *g_tabOrderIsAddOrder* is `nil`.

The *nextField* property set on a field (to any non-`nil` value) overrides the *g_tabOrderIsAddOrder* value.

`?callback` *g_callback*

Triggers callback function when the pages are changed. The default value of this argument is an empty string (`" "`).

`?pageScroller` *g_pageScroller*

Enables a page that has a tab label of `nil` and the page number gets displayed for its tab. If this argument is `nil`, any page that has a tab label of `nil` is disabled and displays a blank tab.

The default value of this argument is `nil`.

`?notebookStyle` *g_notebookStyle*

This argument is now ignored. The Motif Notebook style is not supported any longer.

`?help` *g_fieldHelp* A string or symbol used to reference help. If not specified, *s_name* is used. This argument is currently not used.

Cadence User Interface SKILL Reference

Form Functions

`?invisible` *g_invisible*

Controls if you want the field to be invisible or visible. The default value is `nil`.

You can modify the value of this argument (after the form is created) with the `invisible` property. For example:

```
formHandle->fieldSym->invisible = nil
```

`?tabToolTips` *l_tooltips*

A list specifying strings to be used as tab tooltips.

`?tabLabels` *l_tabLabel*

A list specifying strings to be used as tab labels. You can specify multiple lines as labels. The label strings can be specified in plain text or rich text. For new line, use `<br>` for rich text and `/n` for plain text.

When this option is specified, the value for the `?tab` argument is ignored.

Specifies a list of strings as tab labels. Each string can be specified in plain text or in rich text.

The argument supports multiple-line label. For newline, use `\n` for plain text and `<br>` for rich text.

When specified, the option replaces `?tag` for tab labels.

Cadence User Interface SKILL Reference

Form Functions

`?scrollPage l_scrollPage`

A list specifying whether scrollbar needs to be enabled for different tabs in a form:

```
?scrollPage list(t/nil)
```

The default is `t`, which means that if you do not specify any value or specify the following, scrollbars are added to all tabs of the form:

```
?scrollPage list()
```

The following statement also adds scrollbar to all tabs:

```
?scrollPage list(t)
```

You can specify values for only some of the tabs. In this case, all of the next tabs have the same value as the last value defined in the list. For example, consider you have five tabs and you specify:

```
?scrollPage list(t nil)
```

It is same as specifying:

```
?scrollPage list(t nil nil nil nil)
```

In this example, scrollbars are added only to the first tab.

If you specify the following:

```
?scrollPage list(t nil t)
```

It is same as specifying:

```
?scrollPage list(t nil t t t)
```

This means that all tabs except the second tab have scrollbars.

Value Returned

`r_fieldHandle` Handle to the tab field.

Examples

A form named *Tab Field Form* with the specified tabs is created.

```
CycField = hiCreateCyclicField(  
    ?name      'CycField  
    ?prompt    "Cyclic"  
    ?choiceslist( "hello" "bonjour" "guten tag" "buenos tardes" )  
    ?value     "hello"  
)
```

Cadence User Interface SKILL Reference

Form Functions

```
StringField = hiCreateStringField(
    ?name      'StringField
    ?prompt    "String"
    ?format    "%s"
    ?defValue  "String"
    ?value     "valString"
)

FrameField = hiCreateFrameField(
    ?name'FrameField
    ?labelText"Frame"
)

ListBField = hiCreateListBoxField(
    ?name'ListBField
    ?prompt"List Box"
    ?choiceslist( "bowling" "volleyball" "softball" "baseball"
                  "duckpins" "basketball" "tennis" "football" )
    ?valuelist( 1 3 4 "football" )
    ?numRows3
    ?multipleSelect t
    ?keepHistory t
)

MLTextField = hiCreateMLTextField(
    ?name'MLTextField
    ?prompt"ML Text"
    ?value"The quick brown fox jumped\nover the lazy yellow dog."
    ?defValue"The quick brown fox jumped\nover the lazy yellow dog."
    ?enableWordWrap t
)

IntField = hiCreateIntField(
    ?name'IntField
    ?prompt"Integer"
    ?value4192
    ?defValue4192
)

ListField = hiCreateListField(
    ?name'ListField
    ?prompt"List"
    ?valuelist( 1 2 "three" "four" 4.999764 )
    ?defValuelist( 1 2 "three" "four" 4.999764 )
)

PointField = hiCreatePointField(
    ?name'PointField
```

Cadence User Interface SKILL Reference

Form Functions

```
?prompt"Point"
?value124:255
?defValue124:255
)
BboxField = hiCreateBBoxField(
    ?name'BboxField
    ?prompt"BBox"
    ?valuelist( 12:25 233:165 )
    ?defValuelist( 12:25 233:165 )
)
TabField = hiCreateTabField(
    ?name'TabField
    ?pageScroller nil
    ?fieldslist(
        list(
            list( FrameField 5:17 235:52 15 )
            list( CycField 10:33 130:30 55 )
            list( StringField 5:70 230:10 55 )
        )
        list(
            list( ListBField 5:10 230:80 55 )
            list( MlTextField 5:90 250:80 55 )
            list( IntField 5:170 230:10 55 )
        )
        list(
            list( ListField 5:10 230:10 55 )
            list( PointField 5:40 230:10 55 )
            list( BboxField 5:70 230:10 55 )
        )
    )
    ?tabPlacement 'top
    ?callback'SampleTabFieldCB
    ?tabslist( "Fred" "Wilma" "Barney" )
)
procedure( SampleTabFieldCB( )
    let((MyForm hiGetCurrentForm()))
    MyForm->TabField->defValue = MyForm->tabField->value
    printf("tab field value = %L\n", MyForm->tabField->value)
)
)
TabFieldForm = hiCreateAppForm(
```

Cadence User Interface SKILL Reference

Form Functions

```
?name      'TabFieldForm
?formTitle "Tab Field Form"
?fields list( list( TabField 10:10 300:235 ))
?buttonLayout 'OKCancelApply
?formType 'nonOptions
?unmapAfterCBt
?initialSize t
)
```

```
hiDisplayForm(TabFieldForm)
```

A form named *2D Form* is created. Each tab in the form has tooltips added using the `?tabToolTips` argument. The form has multiple tabs and each tab has a tooltip. The example also shows how to add different types of labels using the `?tabLabels` argument.

```
listField = hiCreateListField(
    ?name      'listField
    ?prompt     "List"
    ?value      list( 1 2 "three" "four" 4.999764 )
    ?defValue    list( 1 2 "three" "four" 4.999764 )
    ;?editable   nil
)
```

```
pointField = hiCreatePointField(
    ?name      'pointField
    ?prompt     "Point"
    ?value      124:255
    ?defValue    124:255
    ?editable   nil
    ?invisible  nil
)
```

```
bboxField = hiCreateBBoxField(
    ?name      'bboxField
    ?prompt     "BBox"
    ?value      list( 12:25 233:165 )
    ?defValue    list( 12:25 233:165 )
    ?editable   nil
)
```

```
ptListField = hiCreatePointListField(
    ?name      'ptListField
    ?prompt     "Point List"
```

Cadence User Interface SKILL Reference

Form Functions

```

    ?value      list( 50:50 60:60 45:95 95:45 50:60 60:50 )
    ?defValue   list( 50:50 60:60 45:95 95:45 50:60 60:50 )
    ?editable   nil
)

tabField1 = hiCreateTabField(
    ?name        'tabField1
    ?fields      list(
        list(
            list( pointField 5:15 230:10 55 )
            ;list( bboxField 5:45 230:10 55 )
            ;list( ptListField 5:75 230:10 55 )
        )
        list(
            list( pointField 5:15 230:10 55 )
            list( bboxField 5:45 230:10 55 )
            ;list( ptListField 5:75 230:10 55 )
        )
        list(
            list( pointField 5:15 230:10 55 )
            list( bboxField 5:45 230:10 55 )
            list( ptListField 5:75 230:10 55 )
        )
    )
    ?tabPlacement 'top
    ?tabs          list( "Tab1" "Tab2" "Tab3")
    ;?tabLabels list( "&Line 1\nLine 2" "&Short\nVery Long Long" "Very Long
Long\nShort")
    ?tabLabels list( "&Line 1\nLine 2" "<b>Short<br>&Very Long Long</b>"
"<h2><b><font color='green'+&Big+</font><<b>"
    ?tabToolTips list("tooltip 1" "tooltip 2" "tooltip 3")
)

tabField2 = hiCreateTabField(
    ?name        'tabField2
    ?fields      list(
        list(
            list( pointField 5:15 230:10 55 )
            ;list( bboxField 5:45 230:10 55 )
            ;list( ptListField 5:75 230:10 55 )
        )
    )

```


Cadence User Interface SKILL Reference

Form Functions

```
list(
    ;list( pointField 5:15 230:10 55 )
    list( bboxField 5:45 230:10 55 )
    ;list( ptListField 5:75 230:10 55 )
)
list(
    ;list( pointField 5:15 230:10 55 )
    ;list( bboxField 5:45 230:10 55 )
    list( ptListField 5:75 230:10 55 )
)
)
;?tabPlacement 'bottom
?tabPlacement 'top
?tabs      list( "Tab1" "Tab2" "&Tab3")
?tabLabels list( "&One Line" "&Short\nVery  Long Long" "<b><font
color='red'>&Red Tab</font></b>")
?tabToolTips list("tooltip a" "tooltip b" "tooltip c")
)

tabField3 = hiCreateTabField(
    ?name      'tabField3
    ?fields    list(
        list(
            list( pointField 5:15 230:10 55 )
            ;list( bboxField 5:45 230:10 55 )
            list( ptListField 5:75 230:10 55 )
        )
        list(
            list( pointField 5:15 230:10 55 )
            list( bboxField 5:45 230:10 55 )
            ;list( ptListField 5:75 230:10 55 )
        )
        list(
            ;list( pointField 5:15 230:10 55 )
            list( bboxField 5:45 230:10 55 )
            list( ptListField 5:75 230:10 55 )
        )
    )
    ?tabPlacement 'top
    ?tabs      list( "Tab&1" "Tab&2" "Tab&3")
    ?tabToolTips list("tooltip x" "tooltip y" "tooltip z")
)
```

Cadence User Interface SKILL Reference

Form Functions

```
)

my2DFields = list(
    list( tabField1 15:15 300:150 15 )
    list( tabField2 15:175 300:150 15 )
    list( tabField3 15:340 300:150 15 )
)

my2DForm = hiCreateAppForm(
    ?name           'my2DForm
    ?formTitle      "2D Form"
    ?fields         my2DFields
    ?buttonLayout   'Close
    ?formType       'nonOptions
    ?unmapAfterCB   t
    ?dontBlock      t
    ?initialSize    1000:800
    ;?maxSize       2000:2100
    ?minSize        330:300
    ?fieldFocus     'allFields
)

my2DForm->tabField3->hiToolTip = "Body 3 ToolTip"
my2DForm->tabField2->hiToolTip = "Body 2 ToolTip"
my2DForm->tabField1->hiToolTip = "Body 1 ToolTip"

hiDisplayForm my2DForm
```

Related Topics

Form and Field Functions

hiCreateTree

```
hiCreateTree(  
    s_treeName  
)  
=> r_tree / nil
```

Description

Creates a new tree or sub-tree, to which you then add items you create with the `hiCreateTreeItem` function. You add the tree to a tree table field created with the `hiCreateTreeTable` function.

Note: This function should not be built into a context. If it is in a context file, it should be within a function, which runs at a run-time, instead of being inline, which runs when the context is built.

For an overview of the tree table field, see [hiCreateTreeTable](#).

Arguments

<i>s_treeName</i>	The name of the tree, specified as a symbol. This symbol is the handle to the tree and can be evaluated to return the tree structure.

Values Returned

<i>r_tree</i>	Handle to the tree.
<i>nil</i>	The tree was not created.

Related Topics

[Form and Field Functions](#)

[hiltemInsertTree](#)

[hiCreateTreeItem](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

Cadence User Interface SKILL Reference

Form Functions

hiltemRemoveTree

hiCreateTreeTable

hiCreateTreeItem

```
hiCreateTreeItem(  
    s_itemName  
    [ g_description ]  
    [ g_itemIcon ]  
    [ g_selectionIcon ]  
    [ g_expandedIcon ]  
    [ g_expandedSelectionIcon ]  
)  
=> r_treeItem / nil
```

Description

Creates a new item for a tree. Use `hiTreeAppendItem` or `hiTreePrependItem` to attach the item to a tree.

For an overview of the tree table field, see [hiCreateTreeTable](#).

Arguments

<i>s_itemName</i>	The name of the item, specified as a symbol. This symbol is the handle to the item and can be evaluated to return the item structure.
<i>g_description</i>	Description of the item that is displayed in the tree table. Specify the description as a list of one or more strings, symbols, integers, or floating-point numbers.
<i>g_itemIcon</i>	Icon that is displayed on the left of the item description. You can also set the icon with the <code>hiSetTreeItemIcons</code> function.
<i>g_selectionIcon</i>	Icon that <i>g_itemIcon</i> changes to when the item is selected. This argument is currently ignored.
<i>g_expandedIcon</i>	Icon that is displayed when the item is expanded. You can also set the expanded icon with the <code>hiSetTreeItemIcons</code> function.
<i>g_expandedSelectionIcon</i>	Icon that is displayed when the item is expanded and selected. This argument is currently ignored.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

<code>r_treeItem</code>	Handle to the tree item.
<code>nil</code>	The item was not created.

Additional Information

You can specify a tooltip for a tree item by setting the `hiToolTip` property on `r_treeItem` (the return value of `hiCreateTreeItem`). The value of the `hiToolTip` property can either be a string or a list of strings. If it is one string, that string is the tooltip for all columns of the tree item. If it is a list of strings, the first string is the tooltip for the first item in `g_description`, the second is the tooltip for the second item in `g_description`, and so on. If there are fewer tooltip items than the number of columns, the last tooltip string is applied to the remaining columns. For example:

```
putprop _i908259_0 (list "one" "two" "three" "four") 'hiToolTip
```

specifies tooltips for all four columns of a tree item.

```
putprop _i908259_1 "tooltip for all cols except those w/o description" 'hiToolTip
```

specifies one tooltip that applies to all columns of the tree item.

```
putprop _i908259_3 (list nil "two" "tooltip for three and four") 'hiToolTip
```

specifies that the first column has no tooltip, the second column has the tooltip `"two"`, and the last two columns have the last string as the tooltip.

You can specify a callback, to be called when a tree item is double-clicked, by setting a property on the tree table field.

Note: This function should not be built into a context. If it is in a context file, it should be within a function, which runs at run-time, instead of being inline, which runs when the context is built.

Related Topics

[Form and Field Functions](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

[hiTreeRemoveItem](#)

[hiCreateTree](#)

Cadence User Interface SKILL Reference

Form Functions

hiSetTreeItemIcons

hiItemInsertTree

hiCreateTreeTable

hiGetTreeItems

hiCreateTreeTable

```
hiCreateTreeTable(  
    ?name s_name  
    [ ?title t_title ]  
    [ ?choice g_choice ]  
    [ ?value g_value ]  
    [ ?defValue g_defaultValue ]  
    [ ?help g_fieldHelp ]  
    [ ?headers g_headers ]  
    [ ?sort l_sort ]  
    [ ?callback g_callback ]  
    [ ?enableDeselectCB g_enableDeselectCB ]  
    [ ?expandCallback g_expandCallback ]  
    [ ?collapseCallback g_collapseCallback ]  
    [ ?sortCallback g_sortCallback ]  
    [ ?nextField g_nextField ]  
    [ ?titleAlignment g_titleAlignment ]  
    [ ?selectMode s_selectMode ]  
    [ ?invisible g_invisible ]  
    [ ?enabled g_enabled ]  
    [ ?altRowHighlight g_rowHighlight ]  
    [ ?expandCollapseOnDoubleClick g_expandCollapseOnDoubleClick ]  
)  
=> r_fieldHandle
```

Description

Creates a tree table field. If you have already created a tree, you can specify it when you create the field; otherwise you can add a tree later to the field. To create a tree, use `hiCreateTree` and other tree table field functions.

Cadence User Interface SKILL Reference

Form Functions

Arguments

<code>?name <i>s_name</i></code>	Handle to the tree table field. To reference it, use: <code>formHandle->hiFieldSym</code>
<code>?title <i>t_title</i></code>	Title of the tree table field, which is displayed above the tree.
<code>?choice <i>g_choice</i></code>	The tree that the tree table field contains. This is the return value of hiCreateTree. To retrieve or change the value of <i>g_choice</i>, use: <code>formHandle->hiFieldSym->choice</code>
<code>?value <i>g_value</i></code>	The initial value of the field, indicating which items in the tree table field are selected when the form first appears. Specify <i>g_value</i> as a list of item symbols. After the form is instantiated, you can use the <code>value</code> property to retrieve or change the list of currently selected items. To retrieve the list of selected items, use the following syntax: <code>formHandle->hiFieldSym->value</code> To change the list of selected items with the <code>value</code> property, use the following syntax: <code>formHandle->hiFieldSym->value = <i>listOfItems</i></code> Setting the value through the <code>value</code> property always triggers the callback of the tree table field. If you want to set the value without triggering the callback, use the functions hiTreeTableSelectItem, hiTreeTableSelectItems, or hiTreeTableSelectAllItems, with the <i>g_notify</i> argument set to <code>nil</code>.
<code>?defValue <i>g_defaultValue</i></code>	

Cadence User Interface SKILL Reference

Form Functions

Default value of the field, indicating which items in the tree table are selected when the *Defaults* button is clicked.

Specify *g_defaultValue* as a list of item symbols.

After the form is instantiated, you can use the `defValue` property to retrieve or change the default value. Use the following syntax for the `defValue` property:

```
formHandle->hiFieldSym->defValue
```

?help *g_fieldHelp*

A string, symbol, list of strings, or list of symbols used to reference help. If this argument is not specified, *s_name* is used. This argument is currently not used.

?headers *g_headers*

Cadence User Interface SKILL Reference

Form Functions

The list of columns in the tree table field. Specify each column as a list that contains the following elements:

```
'(columnName columnWidth colNameAlignment  
type sortable compareFunction)
```

- *columnName* is the heading of the column
- *columnWidth* is the initial width of the column. The width can be changed interactively after the form is created.
- *colNameAlignment* is the alignment of the column heading and the data in the column: 'left, 'right, or 'center

This should be of type symbol.

- *type* is the type of data in the column: 'int, 'float[n], or 'string. You can specify float in the following ways:
- 'floatn (where *n* is the number of decimal points to be displayed in the float field). For example, 'float4.
- 'float (which defaults to 'float3)

This should be of type symbol.

sortable indicates if the column is sortable. Specify *t* if you want the column to be sortable; *nil* otherwise. The report field can sort integer, float, and string data. Clicking on the column header sorts the data in the column.

compareFunction is the name (string) of the optional sorting function you write if you want to sort the data yourself. The return value of this sorting function must be -1, 0, or 1.

You need to specify *type*, *sortable*, and *compareFunction* only if you want the column to be sortable.

After the tree table is displayed, you can choose which columns you want to display. To do so, right-click on the tree table title, and in the pop-up menu that appears, select the columns you want to display and deselect the columns that you want to hide. By default, all the columns are displayed.

Cadence User Interface SKILL Reference

Form Functions

After the form is instantiated, you can use the `headers` property to retrieve or change the header. Use the following syntax to set the header property:

```
formHandle->hiFieldSym->headers = newHeader
```

?sort *l_sort*

Indicates which column is sorted when the tree table first appears. Specify *l_sort* as a list of two elements:

```
'(columnToSort sortDirection)
```

where *columnToSort* is an integer indicating the column to sort (the first column is 0, the second is 1, and so on) and *sortDirection* is `t` or `nil` (`t` indicates descending order, `nil` indicates ascending order).

You can also use the `sort` property to sort a column after the form is instantiated.

Use the following format for the `sort` property:

```
formHandle->hiFieldSym->sort
```

Clicking on the column header sorts the column. Clicking again on the column header changes the sorting direction, from ascending to descending or vice versa.

This argument is valid only if `t` is defined for the column in `?headers g_headers`.

?callback *g_callback*

A string or symbol that specifies the name of the SKILL function to be called when items are selected in the tree table field. If you want the callback function to also be called when items are deselected, specify `t` as the value of the `?enableDeselectCB` argument.

The callback function is passed two arguments:

- the field name (symbol)
- the list of selected items (list of symbols)

The default value of *g_callback* is an empty string (`" "`).

Cadence User Interface SKILL Reference

Form Functions

?enableDeselectCB *g_enableDeselectCB*

t or *nil*. Specify *t* if you want the callback function (specified with the ?callback argument) to be called when items are deselected, in addition to being called when items are selected. If you specify *nil*, the callback function is called only when items are selected.

The default value of *g_enableDeselectCB* is *nil*.

?expandCallback *g_expandCallback*

The SKILL function that is called when any tree node is expanded.

The expand callback function is passed two arguments:

- the field name (symbol)
- the name of the expanded item (symbol)

The default value of *g_expandCallback* is an empty string ("").

?collapseCallback *g_collapseCallback*

The SKILL function that is called when any tree node is collapsed.

The collapse callback function is passed two arguments:

- the field name (symbol)
- the name of the collapsed item (symbol)

The default value of *g_collapseCallback* is an empty string ("").

Cadence User Interface SKILL Reference

Form Functions

?sortCallback *g_sortCallback*

The SKILL function that is called when a column is sorted.

The sort callback function is passed two arguments:

- the field name (symbol)
- `list(columnToSort sortDirection)`

where *columnToSort* is an integer indicating the column to sort (the first column is 0, the second is 1, and so on) and *sortDirection* is `t` or `nil` (`t` indicates descending order, `nil` indicates ascending order).

The default value of *g_sortCallback* is an empty string (`" "`).

?nextField *g_nextField*

Symbol or string indicating the next field to bring into focus when the `Tab` key is pressed. If you specify `nil` or do not specify this argument, the `Tab` traversal order is determined by the *tabOrderIsAddOrder* argument of [hiCreateAppForm](#) or [hiCreateScrollRegion](#). If the *tabOrderIsAddOrder* argument is not set or if its value is `nil`, the default traversal order is based on the form layout—left-to-right and top-to-bottom—beginning with the field at the top-left corner of the form.

?titleAlignment *g_titleAlignment*

Alignment of the title of the tree table field. Use one of the following values:

- `'left`
- `'right`
- `'center`

The default value is `'left`.

?selectMode *s_selectMode*

Cadence User Interface SKILL Reference

Form Functions

Specifies the selection mode for the tree table field. Use one of the following symbols:

- `'single`: Only one item can be selected at a time. Click to select the item. To deselect an item, click it again or select another item.
- `'browse`: One item always remains selected. Click to select the item. To deselect an item, select another item.
- `'multiple`: Multiple items can be selected at a time. When you select an item, its selection state (selected or unselected) is toggled and the other items are unchanged.
- `'extended`: When you select an item, any previously-selected item is unselected. However, if you press `CTRL` and click an item, its selection state (selected or unselected) is toggled and the other items are not changed. If you press `SHIFT` and click an item, all the items between the current item and the clicked item are selected. Multiple items can also be selected by dragging the mouse over them.

The default mode is `'extended`.

You can modify the value of this argument with the `selectMode` property. For example:

```
formHandle->fieldSym->selectMode = 'single
```

```
?invisible g_invisible
```

Specifies whether the layout and the fields within are invisible. The default value is `nil`.

```
?enabled g_enabled
```

Specifies whether the fields in the layout are enabled. The default value is `t`.

```
?altRowHilight g_rowHighlight
```

Specifies whether background of alternate rows in the table should be highlighted. The default is `nil`, which means that highlight is not enabled.

```
?expandCollapseOnDoubleClick g_expandCollapseOnDoubleClick
```

Cadence User Interface SKILL Reference

Form Functions

Specifies whether tree nodes should expand or collapse when double clicked. The default is `t`, which means that it is possible to collapse or expand tree nodes by double clicking.

You can modify the value of this argument with the `mechanism` property. For example:

```
form -> field -> expandCollapseOnDoubleClick ->
mechanism
```

This setting does not affect the call on the registered double-click callback.

Values Returned

r_fieldHandle Handle to the tree table field.

Examples

The following sample creates a form with a tree table field.

```
; create a root tree
indexTree=hiCreateTree('index)

; create two tree items and append them to the root tree
dow=hiCreateTreeItem('dow list("Dow Jones"))
nas=hiCreateTreeItem('nas list("NASDAQ"))
hiTreeAppendItem(indexTree dow)
hiTreeAppendItem(indexTree nas)

; create two sub-trees
dowTree=hiCreateTree('dows)
nasTree=hiCreateTree('nass)

; put the two sub-trees into the two items created earlier
hiItemInsertTree(dow dowTree)
hiItemInsertTree(nas nasTree)

; add a few leaves for the dowTree and nasTree
hiTreeAppendItem(dowTree hiCreateTreeItem('cdn list("Cadence" 80.5 1 "Strong
Buy"))))
hiTreeAppendItem(dowTree hiCreateTreeItem('ibm list("I.B.M." 118 3 "Hold"))))
hiTreeAppendItem(nasTree hiCreateTreeItem('sunw list("Sun Micro" 27.750 4
"Sell"))))
hiTreeAppendItem(nasTree hiCreateTreeItem('orcl list("Oracle" 28.925 5 "Strong
Sell"))))
```


Cadence User Interface SKILL Reference

Form Functions

```
; create callback, which is called whenever any tree items are selected and which
; prints the selected item's description
procedure( myTreeCB(name itemSymList)
    while(itemSymList
        itemSym=car(itemSymList)
        println(hiGetTreeItemDescription(eval(itemSym)))
        itemSymList=cdr(itemSymList)
    )
)
```

```
; create tree table and use indexTree as the value for ?choice
treeField = hiCreateTreeTable(
    ?name 'treeField
    ?title "Stocks"
    ?titleAlignment 'center
    ?headers list(list("Name" 125 'left 'string)
                  list("Price" 60 'left 'float3)
                  list("Rating" 45 'center 'int)
                  list("Recommendation" 125 'right))
    ?choice indexTree
    ?callback "myTreeCB"
)
```

```
; create a form
form = hiCreateAppForm(
    ?name 'form
    ?formTitle "A Tree Sample"
    ?fields list(
        list(treeField 5:5 400:200 55)
    )
    ?initialSize 500:250)

```

```
hiDisplayForm(form)
```

The following sample creates a form with two tree table fields. The two fields are different views of the same tree. View 1 displays the tree structure while View 2 displays more details for each element of the tree. When you make a change in one view, the other view is updated simultaneously. For example, if you expand an item in View 2, the item is also expanded in View 1.

```
procedure( myTreeCB(name items)
    println(name)
    println(items)
)
```

```
procedure( expandCB(name item)
    println("expanded")
    println(name)
    println(item)
)
```

Cadence User Interface SKILL Reference

Form Functions

```
procedure( collapseCB(name item)
    println("collapsed")
    println(name)
    println(item)
)

; create a root node for the tree
rootTree=hiCreateTree('root)

; create four items for the tree and add them to the root
item1=hiCreateTreeItem('it1 '("Folder1" "Folder1 Description" 12.120105))
item2=hiCreateTreeItem('it2 '("Folder2" "Folder2 Description" 1))
item3=hiCreateTreeItem('it3 '("Folder3" "Folder3 Description" -0.5))
item0=hiCreateTreeItem('it0 '("Leaf" "Has no children" 3.22))
hiTreeAppendItem(rootTree item1)
hiTreeAppendItem(rootTree item0)
hiTreeAppendItem(rootTree item2)
hiTreeAppendItem(rootTree item3)

; create a sub-tree for each item
Folder1=hiCreateTree('folder1)
Folder2=hiCreateTree('folder2)
Folder3=hiCreateTree('folder3)
hiItemInsertTree(item1 Folder1)
hiItemInsertTree(item2 Folder2)
hiItemInsertTree(item3 Folder3)

; add items for the sub-trees
item11=hiCreateTreeItem('i11 '("Item11" "Item1 For Folder1" 123))
item12=hiCreateTreeItem('i12 '("Item12" "Item2 For Folder1" 5.7))
item13=hiCreateTreeItem('i13 '("Item13" "Item3 For Folder1" 0.999))
hiTreeAppendItem(Folder1 item11)
hiTreeAppendItem(Folder1 item12)
hiTreeAppendItem(Folder1 item13)

item21=hiCreateTreeItem('i21 '("Item21" "Item1 For Folder2" 321.0) )
item22=hiCreateTreeItem('i22 '("Item22" "Item2 For Folder2" 33))
item23=hiCreateTreeItem('i23 '("Item23" "Item3 For Folder2" 34.0))
hiTreeAppendItem(Folder2 item21)
hiTreeAppendItem(Folder2 item22)
hiTreeAppendItem(Folder2 item23)

item31=hiCreateTreeItem('i31 '("Item31" "Item1 For Folder3" 54.67))
item32=hiCreateTreeItem('i32 '("Item32" "Item2 For Folder3" 12.4) )
item33=hiCreateTreeItem('i33 '("Item33" "Item3 For Folder3" 3.1415) )
hiTreeAppendItem(Folder3 item31)
hiTreeAppendItem(Folder3 item32)
hiTreeAppendItem(Folder3 item33)

; create tree tables
treeF=hiCreateTreeTable(
    ?name          'treeField
    ?title         "Tree Table -- View 1"
    ?selectMode    'extended
```

Cadence User Interface SKILL Reference

Form Functions

```

        ?titleAlignment      'center
        ?callback            "myTreeCB"
        ?expandCallback      "expandCB"
        ?collapseCallback    "collapseCB"
        ?headers             '(("Folder" 120 left) ("Comment" 150 left)
                                ("Misc" 80 right float))
        ?choice              rootTree
    )

treeF2=hiCreateTreeTable(
    ?name                    'treeField2
    ?title                   "Tree Table -- View 2"
    ?selectMode              'extended
    ?titleAlignment          'center
    ?callback                "myTreeCB"
    ?expandCallback          "expandCB"
    ?collapseCallback        "collapseCB"
    ?headers                 '(("Folder" 120 left) )
    ?choice                  rootTree
)

myRep=hiCreateAppForm(
    ?name                    'myRepForm
    ?formTitle               "Multiple-View Tree "
    ?fields                  list(
                                list(treeF2 5:5 350:200 55)
                                list(treeF 5:220 400:250 55)
                            )
    ?initialSize             500:680
)

hiDisplayForm(myRep)

```

Additional Information

Cadence User Interface SKILL Reference

Form Functions

A tree table field displays a tree that can be presented in a table format. Each element of the tree can have several sortable columns of descriptions attached to it. For example, the following tree has two columns:

Name	Price
⊕ Dow Jones	
⊖ Nasdaq	
⋮ Sun Micro	27.750
⋮ Oracle	28.925

The tree table field API provides a set of SKILL functions. You create a tree table field with `hiCreateTreeTable` and specify a tree as the value of the field.

To create a tree,

1. Create a tree object with `hiCreateTree`.
2. Create items for the tree with `hiCreateTreeItem`.
3. Add the items to the tree with `hiTreeAppendItem`, `hiTreeAppendItems`, `hiTreePrependItem`, or `hiTreePrependItems`.

A tree can have several levels. You can create a sub-tree by inserting a new tree under an item with `hiItemInsertTree`.

After the field is created, you can do the following at any time:

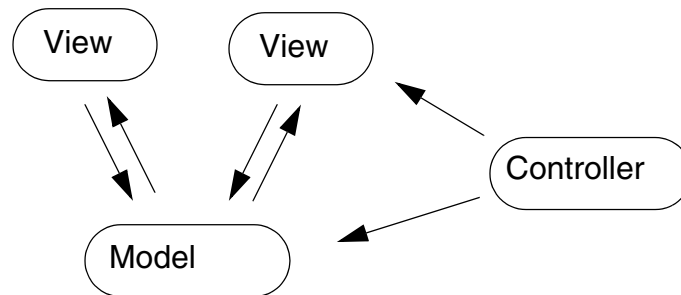
- Add and remove items and sub-trees
- Change the description of the items
- Expand or collapse items in a tree
- Select or deselect items
- Sort columns by clicking on the column header

The tree table API also lets you create multiple views of the same data. The API is based on the model-view-controller paradigm. In the model-view-controller paradigm, a program is implemented in three parts—the model maintains the data; the view displays the data to the

Cadence User Interface SKILL Reference

Form Functions

user; and the controller responds to user actions such as mouse clicks and keyboard input by sending messages to the model or view. This paradigm allows multiple views of the model.



With the tree table API, you can create multiple views of the same data in such a way that when one view makes changes in the data (model), the other views are updated automatically to reflect the changes. For instance, if you expand or collapse a tree in one view, the tree is expanded or collapsed in the other views as well.

Trees and tree items are part of the data model, while the tree-table field is the view. Therefore, actions on the view such as expanding a sub-tree, selecting items, and sorting a column do not have any effect until a view (tree-table field) is instantiated. If you want some tree items to be expanded, for example, when a form is displayed, you need to instantiate the form, then call `hiExpandTreeItem`, and then display the form. For example:

```
hiCreateAppForm( ?name 'myForm, ?fields (list (list myTreeTable ...
(hiInstantiateForm myForm)
hiExpandTreeItem( 'myTreeItem t)
hiDisplayForm( myForm)
```

The following table describes the keyboard interface to a tree table field.

Action	Keys	Select
Ascend or descend the tree	the $\uparrow$ or $\downarrow$ keys	

Cadence User Interface SKILL Reference

Form Functions

Action	Keys	Select
Select an item		■ the item - In extended mode only: ■ CTRL-click to select additional items ■ SHIFT-click to select a range of items between the current item and the clicked item ■ Click and drag to select a range of items
Deselect an item		■ the selected item (in single or multiple mode only) ■ another item (in browse mode only) ■ CTRL-click the selected item (in extended mode only)
Expand a tree	the → key	the  icon -or- double-click the item
Collapse a tree	the ← key	the  icon -or- double-click the item

You can customize the selection mode with the `?selectMode` argument of `hiCreateTreeTable`.

You can also select, deselect, expand, or collapse items in a tree programmatically.

When a tree table field is displayed, you can interactively select the columns that you want to display. To do so, right-click the tree table title and, in the pop-up menu that appears, select the columns you want to display and deselect the columns you want to hide. Columns that are currently displayed have a check-mark next to them.

Cadence User Interface SKILL Reference

Form Functions

A tree table field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field. See [Creating Context Menus for Fields](#) for more information.

You can specify a callback that is invoked when a tree item is double-clicked.

To specify the double-click callback,

- ➔ Set the `hiDoubleClickCallback` property on the tree table field with the following command:

```
hiFormHandle->hiFieldSymbol->hiDoubleClickCallback=g_callback
```

where *hiFormHandle* is the handle to the form returned by `hiCreateAppForm`; *hiFieldSymbol* is the symbol of the tree table field (set with the `?name` argument); and *g_callback* is the callback function, specified as a symbol, a string, or a function object. If *g_callback* is specified as a string, the string is evaluated and the callback function is not passed any arguments. If *g_callback* is specified as a symbol, the callback function is passed the following arguments:

```
(g_formOrRegion s_field s_item)
```

g_formOrRegion is either a form (of type `r`), if the field is placed at the top level of the form, that is, it is not within a scroll region or tab page; a scroll region (of type `o`); or a tab page (of type `o`). *s_field* is the symbol of the tree table field. *s_item* is the item that was double-clicked.

Related Topics

[Form and Field Functions](#)

[hiCreateTreeItem](#)

[hiCreateTree](#)

[hiltemInsertTree](#)

[hiltemRemoveTree](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

[hiTreeRemoveItem](#)

[hiTreeRemoveAllItems](#)

hiCreateToggleField

```
hiCreateToggleField(  
    ?name s_fieldName  
    ?choices l_toggleList  
    [ ?help g_fieldHelp ]  
    [ ?numSelect x_numSelectable ]  
    [ ?prompt t_fieldPrompt ]  
    [ ?value l_currentValues ]  
    [ ?defValue l_defaultValues ]  
    [ ?itemsPerRow x_itemsPerRow ]  
    [ ?font t_font ]  
    [ ?enabled g_enabled ]  
    [ ?invisible g_invisible ]  
    [ ?callback l_toggleListCallbacks ]  
)  
=> r_fieldHandle
```

Description

Creates a toggle field entry for a form.

A toggle field may contain several toggle items which default to being displayed in one row. Each toggle item conveys on or off status, or yes or no meaning. A *numSelectable* argument is optional with toggle fields. This is the maximum number of toggle items that can be simultaneously “turned on.” If *numSelectable* is set to 1 (implying mutual exclusion), a radio type should be used instead. Each toggle item within the toggle field may specify its own callback.

You can enable/disable toggle items, created using `hiCreateToggleField`, individually using the following: `form - toggle - item - enabled -> t/nil`.

This field can also have a context menu, that is, a pop-up menu that is displayed when you right-click on the field.

For information on creating tooltips for fields, see [Creating Tool Tips for Fields](#).

Arguments

?name s_fieldName Handle to this field within a form. To reference it, use:
`formHandle->hiFieldSym`

?choices l_toggleList

Cadence User Interface SKILL Reference

Form Functions

List of symbol, string, and icon, representing handles to individual toggle items within the field and their prompts. You can specify choice for the toggle field in the following ways:

```
list(symbol)
list(symbol string)
list(symbol nil icon)
list(symbol string icon)
```

When you specify a list containing only the symbol for a toggle item, the prompt for the toggle item defaults to the print string of this symbol.

`?help g_fieldHelp` A string or symbol used to reference help. If not specified, *s_fieldName* is used. This argument is currently not used.

`?numSelect x_numSelectable`

Total number of choices (in *toggleList*) that may be selected at one time. This argument defaults to allowing all items to be selected.

`?prompt t_fieldPrompt`

String specifying the prompt for the field. To reference it, use:

```
formHandle->hiFieldSym->prompt
```

`?value l_currentValues`

List where each entry is either `t` or `nil`, representing the on or off value of each choice in *toggleList*. If *currentValues* is not specified, this argument defaults to a list in which every entry is `nil`. To reference the current value, use:

```
formHandle->hiFieldSym->value
```

For toggle fields, an additional level of values may be obtained. The user may either inquire about all toggle values at once (a list) as shown above, or the value of an individual toggle item:

```
formHandle->hiFieldSym->toggleItemSym->value
```

(returns `t` or `nil` for the specified toggle item)

`?defValue l_defaultValues`

Cadence User Interface SKILL Reference

Form Functions

Optional default value of the field. This value is equal to the initial entry in *currentValues* if *defaultValues* is not specified. To reference the default value, use:

```
formHandle->hiFieldSym->defValue
```

?itemsPerRow *x_itemsPerRow*

Row arrangement. This argument is useful for customizing the layout of toggle fields. The items are divided into as many rows as necessary, in row-major order, to accommodate this value. If *itemsPerRow* is 1, the buttons are arranged in a column.

The toolkit imposes restrictions on the geometry allocated to these rows. Although an *itemsPerRow* value is specified, the toolkit may ignore this layout if it does not make optimum use of screen geometry.

?font *t_font*

An optional font argument is accepted but is not supported at this time.

?enabled *g_enabled*

Specifies whether the fields in the layout are enabled. The default value is *t*.

?invisible *g_invisible*

Specifies whether the layout and the fields within are invisible. The default value is *nil*.

?callback *l_toggleListCallbacks*

Optional list of callback strings. If only one callback is specified, it is called for each toggle item, when it changes. If a list is given, the items in the list can be of symbol, string, or function object type.

For more information on callback, see [Cadence User Interface Callback Routines](#).

Cadence User Interface SKILL Reference

Form Functions

Value Returned

r_fieldHandle Handle to the toggle field.

Related Topics

Form and Field Functions

Cadence User Interface SKILL Reference

Form Functions

hiDeleteForm

```
hiDeleteForm(  
    r_form  
)  
=> t / nil
```

Description

Deletes the form specified and any fields the form contains.

The global SKILL handle representing this form is reset to `nil`. Users should not programmatically reset a form SKILL handle to `nil`. This function should not be called within the callback routine of form, or if the form is currently being displayed. Users should take care to delete any unnecessary forms. This routine returns `t` if the form was deleted successfully; otherwise, if this function was called from within the callback procedure, or the form is still displayed on the screen, `nil` is returned.

Argument

<i>r_form</i>	Form to be deleted. It must be a valid form description returned from <code>hiCreateAppForm</code> , <code>hiCreateForm</code> , or <code>hiCreateOptionsForm</code> .

Value Returned

<code>t</code>	The form is deleted.
<code>nil</code>	An error occurred.

Related Topics

[Form and Field Functions](#)

hiDisplayForm

```
hiDisplayForm(  
    g_form  
    [ l_location ]  
)  
=> t / nil
```

Description

Displays a form. Do not use `hiDisplayForm` for displaying options forms. You can also use this function to bring a form to the top of the screen if the form is covered by other windows.

Arguments

g_form

Either a valid form handle returned from `hiCreateAppForm` or `hiCreateForm`, or a symbol whose value is a valid form handle.

l_location

This argument is ignored if the form placement style is anything other than the default. If specified, the form is displayed with its upper left corner at the given location. If no location is specified, the form first appears at a default position set by `hiSetFormPosition`, if previously specified (again, only if the form placement style is the default). Once a form is displayed, it subsequently appears in the position it last appeared on the screen.

The default value can be set in your `.Xdefaults` file with the line:

```
Opus.formPlacement: default
```

or by setting the `formPlacement` property of the Command Interpreter Window to "default" with the line:

```
hiGetCIWindow()->formPlacement="default"
```

If you specify `-1:-1` for the `l_location` argument, this causes the form to be displayed centered under the cursor.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

<code>t</code>	The form is displayed.
<code>nil</code>	You selected <i>Cancel</i> to bring the form down.

If you set the `g_dontBlock` argument of the `hiCreateAppForm` function set to `t`, a call to `hiDisplayForm()` immediately return `t`.

Examples

```
;*****
;Sample form and fields
;*****

; Create string entry for the form.
procedure( myformDefineFields()
  strfldglb = hiCreateStringField(
    ?name `strfldglb
    ?prompt "sample string field"      ;;; FIELD1, a string type
    ?defValue "initial string"
    ?callback "my_strfldglb_cb()"
    ?editable t
  )

  ; Create a Boolean button for the form.
  boolfldglb = hiCreateBooleanButton(
    ?name `boolfldglb
    ?buttonText "sample boolean field"      ;;; FIELD2, a boolean type
    ?defValue t
    ?callback "my_boolfldglb_cb()"
  )
  list( strfldglb boolfldglb )
)

; Generate the form

procedure( myformcreateform()
  if( !boundp(`myformglb) || (myformglb == nil)
    then
      myformglb = hiCreateAppForm(
        ?name `myformglb
```

Cadence User Interface SKILL Reference

Form Functions

```
?fields myformDefineFields()
?formTitle "My first Form"
?callback "myformglb_cb()"
?buttonLayout `OKCancel
)
)
myformglb
)

; Display the form
procedure( myformdisplayform()
  myformcreateForm()    ;;; Creation and Display are separate
  hiDisplayForm(`myformglb)
)

; Check the current state of the form
hiIsFormDisplayed(myformglb) =>nil

myformdisplayform()
hiIsFormDisplayed(myformglb)
=>t
```

Related Topics

[hiCreateAppForm](#)

[Form and Field Functions](#)

hiEditPropList

```
hiEditPropList(  
    g_object  
    [ t_title ]  
    [ g_modifiable ]  
)  
=> t / nil
```

Description

Opens the property list editor for the given *g_object*.

The object must be either a database object, a design manager object, a probe object, or a list. This function is compatible with the SKILL function `makeForm`, which was used in earlier versions of the software, where *g_object* is a list.

This function can be used to create a form by specifying a list as the *g_object*. This list must contain one or more lists of a specified format. This format is the same format specified for the function `makeForm`. Each sublist contains a variable that corresponds to a field on the form. When the form is applied, the new values are assigned to their respective variables, to use them later in a SKILL procedure.

Each of the sublists that make up the main list must be in the following format:

```
'(s_sym t_prompt t_type [g_value] [l_range] [t_help] [l_constraints])
```

The elements of each sublist are:

<i>s_sym</i>	Variable (symbol) representing the field on the form.
<i>t_prompt</i>	Prompt string describing the variable function. This prompt is displayed to the left of the corresponding field on the form.
<i>t_type</i>	String describing the type of the variable. Permitted values are: ■ "integer" or "int" ■ "float" ■ "string" ■ "boolean" or "Boolean" or "yesOrNo" ■ "filename", "fileName" or "file"

Cadence User Interface SKILL Reference

Form Functions

<i>g_value</i>	Optional initial value for the variable (which must be compatible with the type specified). This value is the field's initial value when the form is first displayed. Valid values for a Boolean type are <code>t</code> , "TRUE", "true", "yes", "FALSE", "false", "no", or <code>nil</code> . It is not possible to show an empty value for integer or float types, but strings can have a value of "". If an initial value is not specified, the symbol <i>s_sym</i> is evaluated for the initial value. If <i>s_sym</i> is unbound, a warning is issued and this function returns <code>nil</code> .
<i>l_range</i>	Optional list defining the range of possible values that the variable can assume. This is valid for all types except "boolean". If <i>t_type</i> is "int" or "float", <i>l_range</i> must be a list of two integers or floating point numbers that represent the minimum and maximum values of the variable. If <i>t_type</i> is "string" or "filename", <i>l_range</i> must be a list of possible strings.
<i>t_help</i>	Optional help string for this variable. This is currently not supported.
<i>l_constraints</i>	Optional string or list of strings describing possible constraints on the variable. Currently, the only constraint supported is "read-only." If this constraint is specified, the associated field associated appears grayed-out and is non-editable. This constraint is ignored if <i>t_type</i> is "boolean" or if <i>t_type</i> is "filename" or "string" and an <i>l_range</i> is specified.

Arguments

<i>g_object</i>	Database object, design manager object, probe object, or list.
<i>t_title</i>	Name to be displayed in the window manager frame of the Property List Editor form. If it is not specified, an appropriate title is used.
<i>g_modifiable</i>	Determines if properties can be modified. If <i>g_modifiable</i> is set to <code>t</code> , you are allowed to add, delete, or modify properties by selecting the appropriate button on the form. If set to <code>nil</code> , you are not allowed to add, delete, or modify the properties in any way, except to change their values. The default for <i>g_modifiable</i> is <code>t</code> .

Cadence User Interface SKILL Reference

Form Functions

Values Returned

<code>t</code>	The form has been brought down by pressing the <i>Done</i> button.
<code>nil</code>	The <code>Cancel</code> button was pressed, <code>s_sym</code> is unbound, or <code>g_object</code> is not a valid database object, design manager object, or list.

Examples

The following is an example of `hiEditPropList` using the `list` object:

```
age = 3
mylist = list( ('myname "Name" "string" "Dumbo"
  list("Dumbo" "Mickey Mouse" "Donald Duck") )
  list('age "Age" "int")
  list('weight "Weight" "float" 3600.0
    list(0 9000.0))
  list('type "Type" "string" "ANIMAL" nil
    "" "read-only")
  ('cartoon "Disney character?" "boolean" t))
hiEditPropList( mylist "Cartoon Character" nil )
```

The symbol variables `myname`, `age`, `weight`, `type`, and `cartoon` can now be manipulated in subsequent lines. They contain the values you specify on the form.

Related Topics

[Form and Field Functions](#)

hiEscapeStringChars

```
hiEscapeStringChars(  
    S_stringOrSymbol  
)  
=> s_stringResult
```

Description

Escapes certain characters (backslash and double quote characters) in a string by preceding them with a backslash so that they do not cause errors when the string is evaluated in SKILL. You can use this function to prepare strings that needs to be evaluated in callbacks.

Argument

S_stringOrSymbol String or symbol.

Value Returned

s_stringResult String that preserves backslash and double quote characters when it is evaluated in SKILL.

Related Topics

Form and Field Functions

hiEscapeHTMLTags

```
hiEscapeHTMLTags (
    t_string
)
=> t_escString
```

Description

Returns a string in which HTML tags such as greater-than and less-than are replaced with or converted to character directives such as ampersand-g-t-semicolons and ampersand-l-t-semicolons. This is done to preserve the display of the HTML tags in places such as field prompts.

Argument

<i>t_string</i>	Specifies the string that has HTML tags

Values Returned

<i>t_escString</i>	String that has the converted character directives

Examples

```
hiEscapeHTMLTags ( "&between-ampersands& <in-angle-brackets>" )
=> "&amp; &amp;between-ampersands&amp; &amp; &amp;lt;in-angle-brackets&amp;gt;"
```

Related Topics

[Form and Field Functions](#)

hiExpandTreeItem

```
hiExpandTreeItem(  
    g_item  
    [ g_notify ]  
)  
=> t / nil
```

Description

Expands the tree item you specify. This function also executes the expand callback of the tree table field if you specify *t* as the value of the *g_notify* argument.

This function is the programmatic equivalent of expanding an item with the mouse or keyboard keys.

Arguments

<i>g_item</i>	The item you want to expand. <i>g_item</i> can either be the return value of <code>hiCreateTreeItem</code> or a symbol specifying the item name.
<i>g_notify</i>	Runs the expand callback of the tree table field. Specify <code>nil</code> if you do not want to execute the expand callback.

Values Returned

<i>t</i>	The item was expanded.
<code>nil</code>	The item does not have a tree under it, the item is already expanded, or there was an error.

Related Topics

[hiCollapseTreeItem](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormApply

```
hiFormApply(  
    r_form  
)  
=> t
```

Description

Equivalent to clicking the *Apply* button on a form.

This function takes the form as an argument and executes the Done callback associated with the form without unmapping it. This can either be a symbol or a string. If it is a symbol, it is the name of the function that should be called with the form as an argument. You can call

```
hiInFormApply( form ) => t | nil
```

in the Done callback to find out if the Done callback is called due to an *OK* or an *Apply* of the form.

Argument

<i>r_form</i>	Specifies the form.

Values Returned

t	Operation successful.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiFormCancel](#)

[hiFormClose](#)

[hiFormDefaults](#)

[hiFormDone](#)

hiFormCancel

```
hiFormCancel (  
    r_form  
)  
=> t
```

Description

Equivalent to clicking the *Cancel* button on a form.

If the form is mapped, it gets unmapped. The values of the form fields get restored to the last values saved, and any field callbacks are called if this results in a change of field values.

If `hiFormCancel` is called on a form that has a *Close* button but does not have a *Cancel* button, `hiFormCancel` calls `hiFormClose`.

Argument

<i>r_form</i>	Form to be canceled.

Values Returned

t	The form is canceled.

Related Topics

[hiCreateAppForm](#)

[hiFormApply](#)

[hiFormClose](#)

[hiFormDefaults](#)

[hiFormDone](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormClose

```
hiFormClose(  
    r_form  
)  
=> t
```

Description

Equivalent to clicking the *Close* button on a form. This function is an alias for `hiFormDone`. The *Close* button is identical to the *OK* button, except that it triggers `hiFormClose` instead of `hiFormDone`.

Argument

<i>r_form</i>	Form handle.

Value Returned

t	The form was closed.

Related Topics

[hiFormApply](#)

[hiFormCancel](#)

[hiFormDefaults](#)

[hiFormDone](#)

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormDefaults

```
hiFormDefaults(  
    r_form  
)  
= t / nil
```

Description

Equivalent to clicking the *Defaults* button on a form. The value of each field on the form is compared to its default value and, if it is different, it is set to the default value and the value-change callback is invoked.

Argument

<i>r_form</i>	Form handle.

Values Returned

t	The function was logged.
nil	The <i>Defaults</i> button is disabled.

Related Topics

[Form and Field Functions](#)

[hiFormApply](#)

[hiFormCancel](#)

[hiFormClose](#)

[hiFormDone](#)

[hiCreateAppForm](#)

hiFormDone

```
hiFormDone (  
    r_form  
)  
=> t
```

Description

Equivalent to clicking the *OK* button on a form. This function is identical to the `hiFormFinish` function.

If the form is currently displayed, it gets unmapped according to the value of the `?unmapAfterCB` argument of `hiCreateAppForm` specified when the form was created. If `?unmapAfterCB` is `nil` (the default), the form is unmapped and the values of the form fields are saved before the form's callback function is called. If `?unmapAfterCB` is `t`, the form is unmapped and the field values are saved after the callback function is called, but only if the callback has called `hiSetCallbackStatus` with the form as the first argument and `t` as the second argument.

If the form is not currently displayed, `hiFormApply` run instead of this function.

Argument

<i>r_form</i>	Handle to the form.
	If the form is not currently displayed, <code>hiFormApply</code> runs instead of this function.

Value Returned

<i>t</i>	The function was successful.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiFormCancel](#)

[hiFormClose](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormDefaults

hiFormApply

hiFormFinish

hiFormFinish

```
hiFormFinish(  
    r_form  
)  
=> t
```

Description

Equivalent to clicking the *OK* button on a form. This function is identical to the `hiFormDone` function.

If the form is currently displayed, it gets unmapped according to the value of the `?unmapAfterCB` argument of `hiCreateAppForm` specified when the form was created. If `?unmapAfterCB` is `nil` (the default), the form is unmapped and the values of the form fields are saved before the form's callback function is called. If `?unmapAfterCB` is `t`, the form is unmapped and the field values are saved after the callback function is called, but only if the callback has called `hiSetCallbackStatus` with the form as the first argument and `t` as the second argument.

If the form is not currently displayed, `hiFormApply` runs instead of this function.

Argument

<code>r_form</code>	Handle to the form.
	If the form is not currently displayed, <code>hiFormApply</code> runs instead of this function.

Value Returned

<code>t</code>	The function was successful.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiFormCancel](#)

[hiFormClose](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormDefaults

hiFormApply

hiFormDone

hiFormList

```
hiFormList(  
    )  
=> l_formSymbols
```

Description

Returns a list of all form symbols currently defined in SKILL space.

Arguments

None

Value Returned

<i>l_formSymbols</i>	List of all form symbols currently defined in the SKILL space.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiFormUnmap

```
hiFormUnmap(  
    r_form  
)  
=> t / nil
```

Description

Unmaps the form from the screen before a callback is completed.

This function can be used with `hiSetCallbackStatus` if the *unmapAfterCB* argument to `hiCreateAppForm` has been set to `t`. It unmaps the form only if the callback status of the form is `t` and if it is called within the form `Done` callback. This function is especially useful for lengthy callback routines that must do error checking.

Argument

<i>r_form</i>	Form to be unmapped.

Values Returned

<code>t</code>	The form is unmapped.
<code>nil</code>	<i>r_form</i> is not a form created by <code>hiCreateAppForm</code> .

Examples

This example creates a form that prompts the user for a file name. The form callback immediately unmaps the form before processing the specified file, if the file name is valid.

```
; First, create the form.  
myForm = hiCreateAppForm( ?name 'myForm ?formTitle "title" ?callback 'myCB ?fields  
  
list ( hiCreateStringField( ?name 'fileName ?prompt "Enter filename:" ) )  
?help "" ?unmapAfterCB t )  
  
; myCB - form callback procedure  
procedure ( myCB ( form "r" )  
    ; is the filename valid?  
    if( isFile ( form->fileName->value ) then  
        hiSetCallbackStatus(form t); status is OK  
        hiFormUnmap ( form ); unmap the form  
; process filename...  
    else
```

Cadence User Interface SKILL Reference

Form Functions

```
        ; keep the form on screen
        hiSetCallbackStatus(form nil)
        ; print an error message
        printf("Invalid file '%s'.\n"
            form->fileName->value)
    )
)
```

Related Topics

[hiCreateAppForm](#)

[hiSetCallbackStatus](#)

[Form and Field Functions](#)

hiGetButtonAreaHeight

```
hiGetButtonAreaHeight(  
    r_form  
)  
=> x_height / nil
```

Description

Returns the button area height of an instantiated form.

Argument

<i>r_form</i>	Handle to the form whose button area height you want to retrieve, returned by <code>hiCreateAppForm</code> .

Values Returned

<i>x_height</i>	The button area height.
<i>nil</i>	The form is not instantiated or it is a form window.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetChoiceStrings

```
hiGetChoiceStrings(  
    o_field  
)  
=> l_string
```

Description

Returns strings of all choices in a list box, radio button, or cyclic field.

Argument

o_field Name of a list box, radio button, or cyclic field.

Value Returned

l_string Strings of all choices.

Examples

Returns a string of all choices in a list box, radio button, or cyclic field.

```
form->listbox->choices= (list '("layout" "app-layout.png")  
    '("schematic" (25166049 327 360)) "icde" '("icms" "app-icms.png"))  
  
(hiGetChoiceStrings form->listbox) => ("layout" "schematic" "icde" "icms")  
  
form->cyclic->choices = list("stringValue"  
    '("stringIconPair1" "app-layout.png")  
    '("stringIconPair2" (23448 120 32))  
    '(2394 120 32 "iconString1")  
    '(2345 120 32 iconSymbol1)  
    '(23456 120 32)  
    '("app-layout.png" 120 32)  
    '("app-layout.png" 120 32 "iconString2")  
    list("app-layout.png" 120 32 'iconSymbol2)  
    )  
  
(hiGetChoiceStrings form->cyclic)  
=> ("stringValue" "stringIconPair1" "stringIconPair2" "iconString1" "iconSymbol1"  
    nil nil "iconString2" "iconSymbol2")
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

hiGetCurrentField

```
hiGetCurrentField(  
    r_form  
)  
=> s_fieldName / nil
```

Description

Returns the symbol name of the field that currently has input focus.

Argument

<i>r_form</i>	Form handle. The form cannot be a scroll region field, but the current field may be within a scroll region field.

Values Returned

<i>s_fieldName</i>	The field where the cursor is located.
nil	There are no type-in fields on the form.

Related Topics

[Form and Field Functions](#)

hiGetCurrentForm

```
hiGetCurrentForm( )  
=> r_form
```

Description

Returns the current form, that is, the form that the cursor was last located.

Arguments

None

Values Returned

<i>r_form</i>	Returns the current form.

Related Topics

[Form and Field Functions](#)

hiGetCyclicValueString

```
hiGetCyclicValueString(  
    o_cyclic  
)  
=> t_value / nil
```

Description

Returns the current value of a cyclic field as a string. This function returns `nil` if the specified value is an icon list without the fourth member (optional).

Argument

<code>o_cyclic</code>	Specifies the cyclic field.

Values Returned

<code>t_value</code>	The current value of the cyclic field as a string. The value can be one of the following: ■ A string ■ The string member of a string-icon pair ■ The optional fourth member of an icon list (either a symbol or a string) as a string
<code>nil</code>	The value is an icon list without the fourth member (optional).

Examples

Returns the current values, such as `stringValue`, `stringIconPair1`, `iconSymbol`, and `stringIconPair2`. Also, returns `nil`, when the value is an icon list without the fourth member.

```
form->cyclic->choices = list("stringValue"  
    ("stringIconPair1" "app-layout.png")  
    ("stringIconPair2" (23448 120 32))  
    (2394 120 32 "iconString1")  
    (2345 120 32 iconSymbol1)  
    (23456 120 32)  
    ("app-layout.png" 120 32))
```

Cadence User Interface SKILL Reference

Form Functions

```
        ("app-layout.png" 120 32 "iconString2")
        list("app-layout.png" 120 32 'iconSymbol2)
    )
form->cyclic->value = "stringValue"
"stringValue"

hiGetCyclicValueString( form->cyclic )
=> "stringValue"

form->cyclic->value = list("stringIconPair1" "app-layout.png")
("stringIconPair1" "app-layout.png")

hiGetCyclicValueString( form->cyclic )
=> "stringIconPair1"

form->cyclic->value = list(2345 120 32 'iconSymbol)
(2345 120 32 iconSymbol)

hiGetCyclicValueString( form->cyclic )
=> "iconSymbol"

form->cyclic->value = '("app-layout.png" 120 32)
("app-layout.png" 124 32)

hiGetCyclicValueString( form->cyclic )
=> nil

form->cyclic->value
("app-layout.png" 124 32)

form->cyclic->value = list("stringIconPair2" list(23448 120 32))
("stringIconPair2" (23448 120 32))

hiGetCyclicValueString( form->cyclic )
=> "stringIconPair2"
```

Related Topics

Form and Field Functions

Cadence User Interface SKILL Reference

Form Functions

hiGetFieldAttachments

```
hiGetFieldAttachments(  
    g_formOrRegion  
    s_field  
)  
=> x_attachments | nil
```

Description

Returns the integer representing the combination of attachment bit values when the field is in an attachment form.

A form is an attachment form when the *?attachmentList* argument has been specified.

Arguments

<i>g_formOrRegion</i>	A form, scroll region, or tab page.
<i>s_field</i>	Symbol name of the field.

Values Returned

<i>x_attachments</i>	An integer representing the combination of attachment bit values.
<i>nil</i>	The operation fails and an error message is also issued.

Examples

Consider an attachment form *MyForm* with the field *MyCyclic* that has the following attachments: "hicLeftPercentSet | hicTopPositionSet | hicRightPositionSet".

In this scenario, the following command:

```
hiGetFieldAttachments( MyForm 'MyCyclic )
```

Returns the integer 41 because the sum of the values of *hicLeftPercentSet* (1), *hicTopPositionSet* (8), and *hicRightPositionSet* (32) is 41 (32 + 8 + 1).

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetFieldInfo

```
hiGetFieldInfo(  
    r_2DFormOr2DMenu  
    s_field  
)  
=> l_dimensions / nil
```

Description

Returns the dimensions of any form field or 2D menu item that is in an instantiated form. Do not use the return value of `hiGetFieldInfo` to restore a field to its original location. Instead, use the values you specified originally.

`hiGetFieldInfo` does not work for uninstantiated forms. You can instantiate a form with `hiInstantiateForm` and then use `hiGetFieldInfo`.

Arguments

<code>r_2DFormOr2DMenu</code>	Form or menu returned from <code>hiCreateAppForm</code> , <code>hiCreateForm</code> , or <code>hiCreate2DMenu</code> .
<code>s_field</code>	SKILL symbol representing the field in <code>r_form</code> to be queried. This symbol is the field handle (<code>?name</code>) argument when the field was created.

Value Returned

<code>l_dimensions</code>	A list containing the x and y coordinates, width, height, and the prompt box width or title height (if any) of the field. Note: Do not use the return value of <code>hiGetFieldInfo</code> to restore a field to its original location. Instead, use the values you specified originally.
<code>nil</code>	The list is not returned. An error message is displayed.

Examples

```
list ( ( x y ) ( width height ) pBoxWidth )
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[hiCreateAppForm](#)

[Two Dimensional Form Layout](#)

[Form and Field Functions](#)

hiGetFieldMinSize

```
hiGetFieldMinSize(  
    g_formRegionOrLayout  
    s_fieldName  
)  
=> l_widthHeightPrompt / nil
```

Description

Returns the minimum size for the field widget and prompt.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_fieldName

A symbol name of the field

Values Returned

l_widthHeightPrompt

Returns a list of the minimum width and height of a widget and the minimum width of a prompt if the prompt is to the left or right of the field. Otherwise, it returns the minimum height of a prompt, if the prompt is above the field.

nil

If the arguments are invalid, then a warning message is displayed and *nil* is returned.

Examples

Returns the minimum sizes for the field widget and prompt. If the field does not have a prompt, 0 is returned for the minimum size of the prompt. The default minimum sizes for most of these fields is 0. The default minimum width for a text field is 15.

```
hiSetFont()
```

Cadence User Interface SKILL Reference

Form Functions

```
=> t
hiGetFieldMinSize( hiFontForm 'fontText )
=> (0 0 0)
hiGetFieldMinSize( hiFontForm 'hiFontTextName )
=> (15 0 0)
```

Related Topics

[Form and Field Functions](#)

hiGetRadioValueString

```
hiGetRadioValueString(  
    o_radio  
)  
=> t_value
```

Description

Returns the current value of a radio field as a string.

Arguments

<i>o_radio</i>	Specifies the radio field.

Value Returned

<i>t_value</i>	Returns the string corresponding to the value. The value can be one of the following: ■ A string ■ The string member of a string-icon pair.

Examples

Returns the current values, such as hello, instance-table, and instance-add.

```
form->radio->choices = list( "hello"  
    list("instance-add" "instance-add.png")  
    list("instance-table" ("instance-table.png" 20 20))  
    "so long" )  
)
```

```
form->radio->value = "hello"  
=> "hello"  
hiGetRadioValueString(form->radio)  
=> "hello"
```

```
form->radio->value = ("instance-table" ("instance-table.png" 20 20))  
=> ("instance-table" ("instance-table.png" 20 20))  
hiGetRadioValueString(form->radio)
```

Cadence User Interface SKILL Reference

Form Functions

```
=> "instance-table"
```

```
form->radio->value = "instance-add"
```

```
=> "instance-add"
```

```
hiGetRadioValueString(form->radio)
```

```
=> "instance-add"
```

Related Topics

Form and Field Functions

hiLayoutUTp

```
hiLayoutUTp(  
    g_layout  
)  
=> t / nil
```

Description

Verifies that an argument is an instantiated hiLayout object.

Argument

<i>g_layout</i>	An instantiated hiLayout

Value Returned

t	If the object is an instantiated hiLayout object.
nil	If the object is not an instantiated hiLayout object.

Examples

The object, objName, is an instantiated hiLayout object.

```
hiLayoutUTp("objName")  
=> t
```

Related Topics

[Form and Field Functions](#)

hiGetFieldOverlaps

```
hiGetFieldOverlaps(  
    r_formHandle  
)  
=> t / nil
```

Description

Generates warnings if the form specified contains any fields whose bounding boxes overlap with other fields, or any fields in which the bounding box for the prompt portion of the field overlaps with the main portion of the field. The function generates an initial warning that overlaps were found, then additional warnings for each overlap. Returns `nil` if the form is uninstantiated.

Argument

<i>r_formHandle</i>	Handle to the form which is to be checked for overlapping fields. <i>r_formHandle</i> cannot take a scroll region field as its value.

Values Returned

<code>t</code>	The form is instantiated.
<code>nil</code>	If the form is uninstantiated or 1D / hiLayout forms are specified, then a warning message is displayed and <code>nil</code> is returned.

Related Topics

[Form and Field Functions](#)

hiGetFieldScope

```
hiGetFieldScope(  
    g_field  
)  
=> g_returnField
```

Description

Returns the scope containing the given field. The return value can be a scroll region, a tab page, a tab field, form, or nil.

Argument

<i>g_field</i>	Specifies the field for which you want to retrieve the scope

Values Returned

<i>g_returnField</i>	Returns the scope of the given field, which could be a scroll region, a tab page, a tab field, form, or nil

Example

```
hiGetFieldScope( field )
```

hiGetFormLocation

```
hiGetFormLocation(  
    r_form  
)  
=> l_location / nil
```

Description

Returns the x and y coordinates of the upper-left corner of the form you specify. If the form is not currently displayed, but has been displayed at least once, this function returns the last location at which it was displayed. If the form has never been displayed, this function returns *nil*.

Argument

<i>r_form</i>	Handle to the form, which is returned by <code>hiCreateAppForm</code> .

Value Returned

<i>l_location</i>	The x and y coordinates of the upper-left corner of the form, specified as a list: <code>list(<i>x-coordinate</i> <i>y-coordinate</i>)</code>
<i>nil</i>	<i>r_form</i> is not a valid form ID or it is a form that has not been displayed yet.

Related Topics

[hiCreateAppForm](#)

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetFormSize

```
hiGetFormSize(  
    r_form  
)  
=> l_width_height / 0:0
```

Description

Returns the width and height of the given form.

Argument

<i>r_form</i>	Handle to the form, which is returned by <code>hiCreateAppForm</code> .

Values Returned

<i>l_width_height</i>	Returns form <code>width:height</code> if instantiated form is valid.
<code>0:0</code>	Returns <code>0:0</code> if form is not instantiated, or it is not a form that has been passed in.

Related Topics

[hiCreateAppForm](#)

[Form and Field Functions](#)

hiGetInsertionPosition

```
hiGetInsertionPosition(  
    g_form  
    s_field  
)  
=> x_position / nil
```

Description

Gets the current position of the insertion cursor in a type-in field. You can use the `hiSetInsertionPosition` function to change the position of the cursor.

Arguments

<i>g_form</i>	Handle to the form, which is returned by <code>hiCreateAppForm</code> .
<i>s_field</i>	Symbol name of the field.

Value Returned

<i>x_position</i>	An integer that represents the position of the insertion cursor. The beginning of the field is 0, the first character is 1, and so on.
<i>nil</i>	The position of the insertion cursor could not be obtained.

Related Topics

[hiCreateAppForm](#)

[hiSetInsertionPosition](#)

[Form and Field Functions](#)

hiGetItemOfLayout

```
hiGetItemOfLayout(  
    x_index  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> x_index / nil
```

Description

Returns the description of all the items in the specified hiLayout as per the index order.

Arguments

<i>x_index</i>	The index position of the item.
<i>g_formRegionOrLayout</i>	Any of the following: ■ hiLayout form ■ Scroll region or tab page in hiLayout form ■ hiLayout
<i>s_layoutName</i>	A symbol name of a hiLayout, field in the form, or region.

Values Returned

<i>x_index</i>	The description of all the items in the specified hiLayout in index order.
<i>nil</i>	Nil, if the items in the specified hiLayout does not exist.

Related Topics

[Form and Field Functions](#)

hiGetLayerCyclicValue

```
hiGetLayerCyclicValue(  
    d_techFileId  
    r_form  
)  
=> d_layerId / nil
```

Description

Returns the layer object associated with the current value of the layer cyclic field.

This function should be called only if the form contains a layer cyclic field.

Arguments

<i>d_techFileId</i>	The techfile object associated with the layers of the cyclic field.
<i>r_form</i>	Form containing the layer cyclic field. This form handle is returned from <code>hiCreateAppForm</code> .

Values Returned

<i>d_layerId</i>	The layer (database object).
<i>nil</i>	An error occurred. An error message is also issued.

Related Topics

[hiCreateAppForm](#)

[hiCreateLayerCyclicField](#)

[Form and Field Functions](#)

hiGetLayoutChildLayouts

```
hiGetLayoutChildLayouts(  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> l_layouts / nil
```

Description

Returns the list of symbolic names of child-hiLayout items in the hiLayout.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_layoutName

A symbol name of a hiLayout, field in the form, or region.

Values Returned

l_layouts

The list of symbolic names of hiLayout items in the hiLayout.

nil

Nil, if no symbolic names exist in the hiLayout.

Examples

Returns the symbolic names of the hiLayout items `MyLayout1`, `MyLayout2`, and `MyLayout3` in the hiLayout from the child-layout, `MySubLayout`.

```
hiGetLayoutChildLayouts( MyForm 'MySubLayout )  
=>(MyLayout1 MyLayout2 MyLayout3)
```

Related Topics

Form and Field Functions

hiGetLayoutFields

```
hiGetLayoutFields(  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> x_fields / nil
```

Description

Returns the list of symbolic names of field items in hiLayout.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

If it is a form or region and *s_layoutName* is not specified, the main hiLayout of the form or region is used.

s_layoutName

A symbol name of a hiLayout, field in the form, or region.

Values Returned

l_fields

The list of symbolic names of the field items in the hiLayout

nil

Nil, if the symbolic names of the field items does not exist.

Examples

Returns the symbolic names of field items, MyField1, MyField2, and MyField3 from the layout, MySubLayout.

```
hiGetLayoutFields( MyForm ) ;; returns fields contained in MyMainLayout  
=> nil
```

```
hiGetLayoutFields( MyForm 'MySubLayout )  
=> MyField1 MyField2 MyField3
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

hiGetLayoutItems

```
hiGetLayoutItems(  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> x_index / nil
```

Description

Returns the description of all the items in the specified hiLayout in the index sequence.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_layoutName

A symbol name of a hiLayout, field in the form, or region.

Values Returned

x_index

The description of all the items in the specified hiLayout in the index sequence.

nil

The items in the specified hiLayout do not exist.

Related Topics

[Form and Field Functions](#)

hiGetListBoxValue

```
hiGetListBoxValue(  
    o_listBox  
)  
=> l_stringValues
```

Description

Provides the value of a list box field as a list of strings. This function is useful when the `valueByPosition` property of a list box field is set to `non-nil`, which means its value is managed and reported as a list of integers (indicating position) instead of a list of strings.

The return value of this function does not retain positional information, so if the list box field has multiple identical strings that you identify by their position, no information is provided by the return value.

Argument

<i>o_listBox</i>	Handle to the list box field.

Value Returned

<i>l_stringValues</i>	Value of the list box field as a list of strings.

Related Topics

[Form and Field Functions](#)

[hiCreateListBoxField](#)

hiGetLayoutFrame

```
hiGetLayoutFrame(  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> t_title / t / nil
```

Description

Returns the frame title of a hiLayout form, a scroll region, or a tab page in the hiLayout form.

Arguments

g_formRegionOrLayout

Any of the following values:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_layoutName

A hiLayoutSymbol for a hiLayout contained within the form or region. If this argument is not specified, then the main hiLayout of the form or region is used.

Value Returned

t_title

The frame title.

t

True, if the frame has no title.

nil

Nil, if the layout is not a frame layout.

Examples

Returns *nil* because the layout is not a main layout.

```
hiGetLayoutFrame( MyForm )  
=> nil
```

Returns *nil* because the layout is not a frame layout.

```
hiGetLayoutFrame( MyForm 'MyMainLayout )
```

Cadence User Interface SKILL Reference

Form Functions

```
=> nil
```

Returns the frame title, `Frame`.

```
hiGetLayoutFrame( MyForm 'MyTitleLayout )
```

```
=> "Frame"
```

Returns `t` because the frame, `'MyFrameLayout`, has no title.

```
hiGetLayoutFrame( MyForm 'MyFrameLayout )
```

```
=>t
```

Related Topics

[Form and Field Functions](#)

hiGetLayoutItemIndex

```
hiGetLayoutItemIndex(  
    g_formOrRegion  
    s_layoutOrField  
)  
=> l_layoutIndex / nil
```

Description

Returns a list containing the symbol name of the layout, which contains `s_layoutOrField`, and the index of that item within the layout.

Arguments

<code>g_formOrRegion</code>	A form, scroll region, or a tab page.
<code>s_layoutOrField</code>	A symbol name of a <code>hiLayout</code> , field in the form, or region.

Value Returned

<code>l_layoutIndex</code>	A list containing the symbol name of the layout that contains <code>s_layoutOrField</code> and the index of that item within the layout. Returned as the first argument.
<code>nil</code>	<code>hiLayout</code> is not found.

Examples

If `MyForm` contains `MyLayout`, which contains fields `MyField1`, `MyField2`, and `MyField3` in that order followed by `MyLayout1`. Then, `MyLayout1` returns `list('MyLayout 3)`, and `MyField2` returns `list('MyLayout 1)`.

```
hiGetLayoutItemIndex( MyForm 'MyLayout1 )  
=> list('MyLayout 3)  
hiGetLayoutItemIndex( MyForm 'MyField2 )  
=> list('MyLayout 1)
```

Related Topics

[Form and Field Functions](#)

hiGetLayoutType

```
hiGetLayoutType (
    g_formRegionOrLayout
    [ s_layoutName ]
)
=> s_type / nil
```

Description

Returns the type of hiLayout. The layout types that can be returned are 'hbox, 'vbox, 'grid, 'form, or 'stacked.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_layoutName

The hiLayoutSymbol for an hiLayout contained within the form or region. This argument is ignored if *g_formRegionOrLayout* is a hiLayout.

If *g_formRegionOrLayout* is a form or region and *s_layoutName* is not specified, the main hiLayout of the form or region is used.

Value Returned

s_type

Returns the type of the hiLayout.

nil

hiLayout is not valid.

Examples

A form, *MyForm*, with the main layout, *MyMainLayout*, has a vertical box layout 'vbox, which is returned.

```
hiGetLayoutType ( MyForm )
```


Cadence User Interface SKILL Reference

Form Functions

=> 'vbox

Related Topics

Form and Field Functions

hiGetStackedLayoutCurrentIndex

```
hiGetStackedLayoutCurrentIndex(  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> x_index / nil
```

Description

Returns the index of the current top item.

Arguments

g_formRegionOrLayout

Any of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_layoutName

The hiLayoutSymbol for a hiLayout contained within the form or region. This argument is ignored if *g_formRegionOrLayout* is a hiLayout.

If *g_formRegionOrLayout* is a form or region and *s_layoutName* is not specified, the main hiLayout of the form or region is used.

Values Returned

x_index

The index position of the current top item if the arguments resolve to a valid stacked hiLayout.

nil

Arguments do not resolve to a valid stacked hiLayout.

Examples

The function returns 2. A form, *MyForm*, within the main layout, *MyMainLayout*, is a stacked layout that has 5 items where the current top item is index 2.

```
hiGetStackedLayoutCurrentIndex( MyForm 'MyMainLayout )
```

Cadence User Interface SKILL Reference

Form Functions

=> 2

Related Topics

Form and Field Functions

hiSetFieldMinSize

```
hiSetFieldMinSize(  
    g_formRegionOrLayout  
    s_fieldName  
    [ ?widgetWidth x_widgetWidth ]  
    [ ?widgetHeight x_widgetHeight ]  
    [ ?promptWidth x_promptWidth ]  
)  
=> t / nil
```

Description

Sets the minimum width and height for the widget portion of a field.

Arguments

g_formRegionOrLayout

Any of the following:

- An instantiated hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_fieldName

The symbol name of a field in the form, region, or layout

?widgetWidth *x_widgetWidth*

The minimum width for the widget portion of a field.

?widgetHeight *x_widgetHeight*

The minimum height for the widget portion.

?promptWidth *x_promptWidth*

The minimum prompt width, if the prompt is towards the side of the field. Else, the minimum prompt height if the prompt is above the field.

Cadence User Interface SKILL Reference

Form Functions

Value Returned

<code>t</code>	True, if the form is instantiated and all the specified sizes are valid.
<code>nil</code>	Nil, if the form is not instantiated or any of the specified sizes are invalid.

Examples

The string field, `MyStringField`, exists in form, `MyForm`. The minimum prompt width is set to 100 and the minimum field width is set to 300.

```
hiSetFieldMinSize( MyForm 'MyStringField ?widgetWidth 300 ?promptWidth 100 )
```

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiSetKeyboardFocusField

```
hiSetKeyboardFocusField(  
    g_formOrRegion  
    s_field  
    [ g_limitSearch [ g_onlyCurrentTabPage ] ]  
)  
=> t / nil
```

Description

Sets the keyboard focus to field, `s_field`, if it is found in the form or region, `g_formOrRegion` and is allowed to have keyboard focus. It also enables you to do a hierarchical search if `g_limitSearch` is not specified or is set to `nil`. The search is limited to only the top-level of the form or scroll region field if, `g_limitSearch` is specified as a non-`nil` value.

Arguments

<code>g_formOrRegion</code>	The form, scroll region field, or tab field page in which you want to set the current field.
<code>s_field</code>	The field on which the keyboard focus needs to be set.
<code>g_limitSearch</code>	The level of the form or specified region up to which the search is limited.
<code>g_onlyCurrentTabPage</code>	The currently visible tab page of the tab field up to which the search is limited.

Values Returned

<code>t</code>	True, if the input focus is set to the field.
<code>nil</code>	Nil, if the input focus could not be set to the field.

Examples

Sets keyboard input focus on the *Place Manually* Boolean button field in the User Preferences dialog.

```
hiSetKeyboardFocusField( hiSysProps 'ciwHistoryLines )
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

hiSetLayoutFrameTitle

```
hiSetLayoutFrameTitle(  
    g_title  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> t_title / t / nil
```

Description

Sets the frame title of a hiLayout form, a scroll region, or a tab page in the hiLayout form.

Arguments

<i>g_title</i>	Set the title for a frame layout.
<i>g_formRegionOrLayout</i>	Any of the following values: ■ hiLayout form ■ Scroll region or tab page in hiLayout form ■ hiLayout
<i>s_layoutName</i>	A hiLayoutSymbol for a hiLayout contained with the form or region. If this argument is not specified, then the main hiLayout of the form or region is used.

Value Returned

<i>t_title</i>	The frame title.
<i>t</i>	True, if the frame has no title.
<i>nil</i>	Nil, if the layout is not a frame layout.

Examples

Set and returns the title, Frame for the frame layout, 'MyFrameLayout.

```
hiSetLayoutFrameTitle( "Frame" MyForm 'MyFrameLayout )  
=> Frame
```


Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

hiSetLayoutMargins

```
hiSetLayoutMargins(  
    l_margins  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> t / nil
```

Description

Set the top, bottom, left, and right margins of the layout.

Arguments

l_margins Accepts integer values or `nil` for 'left', 'top', 'right', and 'bottom' (in the same sequence) used to apply margins. If `nil` is specified, then the default margins are applied.

g_formRegionOrLayout

Any of the following:

- `hiLayout` form
- Scroll region or tab page in `hiLayout` form
- `hiLayout`

s_layoutName

The `hiLayoutSymbol` for an `hiLayout` contained with the form or region. This argument is ignored if `g_formRegionOrLayout` is a `hiLayout`.

If `g_formRegionOrLayout` is a form or region and `s_layoutName` is not specified, the main `hiLayout` of the form or region is used.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

<code>t</code>	True, if: ■ The arguments resolve to a valid hiLayout. ■ The <code>l_margins</code> argument is invalid but a valid hiLayout is referenced, a warning is issued and the margins are unchanged. ■ <code>l_margins</code> is valid and the margins are updated.
<code>nil</code>	Arguments does not resolve to a valid hiLayout.

Examples

In `MyForm`, with main layout, `MyMainLayout`, the left and right margins of the main layout are changed to 5 and the top margin to 8. The bottom margin remains unchanged.

```
hiSetLayoutMargins( list( 5 8 5 nil ) MyForm )
hiSetLayoutMargins( list( 5 8 5 -1 ) MyForm->MyMainLayout )
hiSetLayoutMargins( list( 5 8 5 nil ) MyForm 'MyMainLayout )
=> t
```

Related Topics

[Form and Field Functions](#)

hiGetGridLayoutItemLocation

```
hiGetGridLayoutItemLocation(  
    g_formOrRegion  
    s_layoutOrField  
)  
=> l_gridLayoutLocation | nil
```

Description

Searches for a list containing the symbol name of the layout that contains *s_layoutOrField*, and the row, column, row span, and column span of that item within the layout.

Arguments

<i>g_formOrRegion</i>	An instantiated form, scroll region, or a tab page.
<i>s_layoutOrField</i>	The symbol name of a hiLayout or field in the form or region.

Values Returned

l_gridLayoutLocation

A list containing the symbol name of the layout containing *s_layoutOrField*, and the row, column, row span, and column span of that item within the layout.

nil	<i>g_formOrRegion</i> is not instantiated, or either the field or hiLayout is not found in it.

Examples

If MyForm contains a grid layout that was created as follows:

```
MyNameLayout = hiCreateGridLayout( 'MyNameLayout ?frame "Enter Full Name" ?items  
list (  
    list ( hiCreateLabel( ?name 'MyFirstLabel ?labelText "First" ) 'row 0 'col 0 )  
    list ( hiCreateStringField( ?name 'MyFirstName ) 'row 0 'col 1 )  
    list ( hiCreateLabel( ?name 'MyMiddleLabel ?labelText "Middle" ) 'row 1 'col 0 )  
    list ( hiCreateStringField( ?name 'MyMiddleName ) 'row 1 'col 1 )  
    list ( hiCreateLabel( ?name 'MyLastLabel ?labelText "Last" ) 'row 2 'col 0 )  
    list ( hiCreateStringField( ?name 'MyLastName ) 'row 2 'col 1 )  
    list ( 'col_stretch 1 1 )  
) )
```

MyMiddleLabel is in the grid layout, MyNameLayout, in row 1 and column 0, and spans 1 row and 1 column.

Cadence User Interface SKILL Reference

Form Functions

```
hiGetGridLayoutItemLocation( MyForm 'MyMiddleLabel )  
=> (MyNameLayout 1 0 1 1)
```

Related Topics

[Form and Field Functions](#)

hiGetKeyboardFocusField

```
hiGetKeyboardFocusField(  
    g_formOrRegion  
)  
=> s_field / nil
```

Description

Returns the field that currently has keyboard focus or is `nil`.

Argument

<i>g_formOrRegion</i>	Form structure or region.

Values Returned

<i>s_field</i>	The last field that received the keyboard input focus.
<code>nil</code>	No field received the keyboard input focus.

Examples

Returns the last field that received the keyboard input focus in the User Preferences form.

```
hiGetKeyboardFocusField( hiSysProps )
```

Related Topics

[Form and Field Functions](#)

hiGetLayoutMargins

```
hiGetLayoutMargins(  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> l_margin / nil
```

Description

Returns the top, bottom, left, and right margins of the layout.

Arguments

g_formRegionOrLayout

Specifies one of the following:

- hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

s_layoutName

The hiLayoutSymbol for an hiLayout contained within the form or region. This argument is ignored if *g_formRegionOrLayout* is a hiLayout.

If *g_formRegionOrLayout* is a form or region and *s_layoutName* is not specified, the main hiLayout of the form or region is used.

Values Returned

l_margin

A list of four integers and/or nil values that specifies the left, top, right, and bottom margins (in this order). If the form or layout is not yet instantiated, the default values are used and *nil* is returned. However, if the form or layout is instantiated, the margin values are returned.

nil

Nil, if the hiLayout form does not exist or a valid hiLayout is not specified.

Cadence User Interface SKILL Reference

Form Functions

Examples

In form, `MyForm`, with main layout, `MyMainLayout`, the left and right margins are set to have the default values and the top margin is set to 8 and the bottom margin to 4.

```
hiGetLayoutMargins( MyForm )  
=> (nil 8 nil 4)
```

Related Topics

[Form and Field Functions](#)

hiGetNumVisibleItems

```
hiGetNumVisibleItems(  
    o_listbox  
)  
=> x_count / nil
```

Description

Returns the number of visible items for a list box field.

The field must have been instantiated for this function to return a value other than `nil`.

Argument

<code>o_listbox</code>	Name of the list box.

Value Returned

<code>x_count</code>	Number of visible items.
<code>nil</code>	The field is not instantiated.

Related Topics

[Form and Field Functions](#)

[hiGetListBoxFieldFit](#)

hiGetScrollBarInfo

```
hiGetScrollBarInfo(  
    g_formOrScrollRegion  
)  
=> l_scrollbarInfo
```

Description

Provides information about the horizontal and vertical scroll bars on a form or scroll region field. You can use this information to set the position of scroll bars with the `hiSetScrollBarValue` function.

Argument

g_formOrScrollRegion

Handle to the form or scroll region field.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

l_scrollbarInfo

A list of the following form:

```
list ( [ nil | list ( x_horzMin x_horzMax  
x_horzValue x_horzSliderSize ) ] [ nil |  
list( x_vertMin x_vertMax x_vertValue  
x_vertSliderSize ) ] )
```

The first element of the list provides information about the horizontal scroll bar; the second element of the list provides information about the vertical scroll bar. If the first element is *nil*, the form or scroll region field does not have a horizontal scroll bar; if the second item in the list is *nil*, the form or scroll region field does not have a vertical scroll bar.

x_horzMin and *x_vertMin* are the minimum pixel position (usually 0); *x_horzMax* and *x_vertMax* are the horizontal or vertical pixel size of the form or scroll region field; *x_horzValue* and *x_vertValue* are the current values of the scroll bars; and *x_horzSliderSize* and *x_vertSliderSize* are the visible areas of the form or scroll region field.

The maximum values for the *l_newValues* argument of the *hiSetScrollBarValue* function are:

x_horzMax - *x_horzSliderSize* and *x_vertMax* - *x_vertSliderSize*.

Related Topics

Form and Field Functions

hiSetScrollBarValue

hiGetTextFieldFit

```
hiGetTextFieldFit(  
    x_rows  
    x_columns  
    g_ignored  
    g_hasVerticalScrollbar  
    g_hasHorizontalScrollbar  
)  
=> l_dimensions
```

Description

Returns the size necessary for a multiline text field to display *x_rows* and *x_columns* of characters.

For more information about multiline text fields, see [hiCreateMLTextField](#).

Arguments

<i>x_rows</i>	The number of rows of text.
<i>x_columns</i>	The number of columns of text.
<i>g_ignored</i>	Any value that exists for backwards compatibility and is ignored.
<i>g_hasVerticalScrollbar</i>	Indicates if there should be a vertical scroll bar displayed for the multi-line text field.
<i>g_hasHorizontalScrollbar</i>	Indicates if there should be a horizontal scroll bar displayed for the multi-line text field.

Value Returned

<i>l_dimensions</i>	A list containing the width and height in pixels.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetTopListItem

```
hiGetTopListItem(  
    o_listbox  
)  
=> x_position / nil
```

Description

Returns the top item position of a listbox field.

Argument

<i>o_listbox</i>	SKILL handle to the listbox.

Values Returned

<i>x_position</i>	The top item position of the listbox field. 1 is the first item.
<i>nil</i>	The listbox is invalid or the form is not instantiated.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetTree

```
hiGetTree(  
    g_item  
)  
=> s_treeName / nil
```

Description

Gets the tree that is inserted under the item you specify.

Argument

<i>g_item</i>	The handle to the item, which is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<i>s_treeName</i>	The name of the tree that is inserted under the item.
<i>nil</i>	The item does not have a tree under it or there was an error.

Related Topics

[Form and Field Functions](#)

[hiltemRemoveTree](#)

[hiltemInsertTree](#)

[hiGetTreeItems](#)

[hiGetTreeParent](#)

[hiGetTreeItemParent](#)

[hiCreateTreeTable](#)

hiGetTreeItems

```
hiGetTreeItems(  
    g_tree  
)  
=> l_listofItems / nil
```

Description

Gets a list of all the items in a tree.

Argument

<i>g_tree</i>	The tree whose items you want to retrieve. This is the return value of <code>hiCreateTree</code> .

Values Returned

<i>l_listofItems</i>	The list of items in the tree.
<i>nil</i>	The tree does not have any items or there was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeRemoveItem](#)

[hiTreeRemoveAllItems](#)

[hiExpandTreeItem](#)

[hiCollapseTreeItem](#)

[hiTreeTableSelectItem](#)

[hiTreeTableSelectItems](#)

[hiTreeTableSelectAllItems](#)

hiGetTreeItemDescription

```
hiGetTreeItemDescription(  
    g_item  
)  
=> l_description / nil
```

Description

Returns the description of an item. The description is set initially with the *g_description* argument of `hiCreateTreeItem`. You can reset the description with `hiSetTreeItemDescription`.

Argument

<i>g_item</i>	The item whose description you want to retrieve. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<i>l_description</i>	The description of the item.
<i>nil</i>	The description could not be retrieved.

Related Topics

[Form and Field Functions](#)

[hiSetTreeItemDescription](#)

[hiCreateTreeItem](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

hiGetTreeItemIcons

```
hiGetTreeItemIcons(  
    r_treeItem  
)  
=> l_icons
```

Description

Gets the following icons associated with a tree item: default icon, expanded icon (displayed when the item is expanded), selection icon (displayed when the item is selected), and expanded selection icon (displayed when the item is both expanded and selected).

These icons are set when the tree item is created with the `hiCreateTreeItem` function or later with the `hiSetTreeItemIcons` function.

Argument

<i>r_treeItem</i>	Handle to the tree item, returned by <code>hiCreateTreeItem</code> .

Values Returned

<i>l_icons</i>	A list of four icons, in the following order: <i>g_selectionIcon</i> and <i>g_expandedSelectionIcon</i> are currently not displayed. (<i>g_itemIcon</i> <i>g_expandedIcon</i> <i>g_selectionIcon</i> <i>g_expandedSelectionIcon</i>)

Related Topics

[Form and Field Functions](#)

[hiSetTreeItemIcons](#)

[hiSwapTreeItemSelectionIcons](#)

[hiCreateTreeItem](#)

hiGetTreeItemParent

```
hiGetTreeItemParent(  
    g_item  
)  
=> l_treeNames / nil
```

Description

Gets the parent tree of an item, that is, the tree to which the item belongs. If the item is used in multiple places, this function returns all the parent trees.

Argument

<i>g_item</i>	The item whose parent you want to retrieve. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<i>l_treeNames</i>	List of symbols specifying the parent trees.
<i>nil</i>	The item does not have a parent or there was an error.

Related Topics

[Form and Field Functions](#)

[hiGetTreeParent](#)

[hiGetTree](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

hiGetTreeParent

```
hiGetTreeParent(  
    g_tree  
)  
=> l_itemNames / nil
```

Description

Gets the parent of a tree, that is, the item under which the tree is inserted. If the tree is inserted in multiple places, this function returns all the items under which it is inserted.

Argument

<i>g_tree</i>	The tree whose parent you want to retrieve. This is the return value of <code>hiCreateTree</code> .

Values Returned

<i>l_itemNames</i>	List of symbols specifying the parent items.
<i>nil</i>	The tree does not have a parent or there was an error.

Related Topics

[Form and Field Functions](#)

[hiGetTreeItemParent](#)

[hiGetTree](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

hiHighlightField

```
hiHighlightField(  
    g_form  
    s_field  
    g_highlightType  
)  
=> t / nil
```

Description

Highlights a type-in field depending on the value of *g_highlightType*.

Highlighting is allowed for type-in fields only, as these are the fields into which invalid data might be entered, or data may be required.

To find the field, this function also searches the scroll region fields within the form or scroll region field specified as *r_form*. It does a depth-first hierarchical search, which means that scroll region fields within a form or scroll region field are searched before the top level of the form or scroll region field.

Arguments

<i>g_form</i>	Form returned from <code>hiCreateAppForm</code> or <code>hiCreateForm</code> , or scroll region field returned from <code>hiCreateScrollRegion</code> . Scroll region fields within the form or scroll region field are also searched.
<i>s_field</i>	SKILL field handle representing the field you wish to highlight. The field must be a text-entry (type-in) field only.
<i>g_highlightType</i>	It can be either a symbol or an integer. As a symbol, it must be <code>error</code>, <code>highlight</code> or <code>background</code> to set the background color to an error, highlight, or background (default) color, respectively. As an integer, it must be <code>hicError</code>, <code>hicHighlight</code>, or <code>hicBackground</code>. If invalid data has been entered, error color should be specified. Highlight color may be used to indicate required data. To reset the color to the original background color, background color should be used.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	Highlighting is set.
nil	Highlighting is not set. An error message is also issued.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiCreateForm](#)

[hiCreateScrollRegion](#)

hiHypertextReadin

```
hiHypertextReadin(  
    g_hypertextField  
    t_textFile  
)  
=> t / nil
```

Description

Reads in a text file, which can include hyperlinks, into a hypertext field. This function resets the `value` of the hypertext field.

Arguments

<i>g_hypertextField</i>	Symbol representing the hypertext field that you want to update.
<i>t_textFile</i>	Text file, which can include hyperlinks, that you want to read into the hypertext field.

Values Returned

<code>t</code>	The text file was read in.
<code>nil</code>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiHypertextReplace](#)

[hiHypertextScroll](#)

[hiCreateHypertextField](#)

[hiCreateSimpleHypertextField](#)

[hiLaunchBrowser](#)

hiHypertextReplace

```
hiHypertextReplace(  
    g_hypertextField  
    t_newText  
)  
=> t / nil
```

Description

Replaces the text in a hypertext field with the new text that you provide. The text can contain hyperlinks. This function replaces the `value` of the hypertext field with the new text.

Arguments

<i>g_hypertextField</i>	Symbol representing the hypertext field that you want to update.
<i>t_newText</i>	The new text that you want to display in the hypertext field. The text can contain hyperlinks.

Values Returned

<i>t</i>	The text in the hypertext field was replaced.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiHypertextReadin](#)

[hiHypertextScroll](#)

[hiCreateHypertextField](#)

[hiCreateSimpleHypertextField](#)

[hiLaunchBrowser](#)

hiHypertextScroll

```
hiHypertextScroll(  
    g_hypertextField  
    t_linkName  
)  
=> t / nil
```

Description

Scrolls a hypertext field to display the text containing the link you specify. This function updates the ?pos argument of hiCreateHypertextField().

Arguments

<i>g_hypertextField</i>	Symbol representing the hypertext field that you want to update.
<i>t_linkName</i>	The link to which you want to scroll, specified as a string. The link name is defined with the NAME attribute of the anchor tag when you specify the value of the hypertext field.

Values Returned

t	The hypertext field was scrolled to display the link.
nil	There was an error.

Examples

If your field is myField and the link in the text is:

```
"<A NAME=myLink href=http://www.cadence.com> Cadence Design Systems</A>"
```

Use the following to scroll to the link:

```
hiHypertextScroll( myField "myLink" )
```

Related Topics

Form and Field Functions

hiCreateHypertextField

Cadence User Interface SKILL Reference

Form Functions

hiCreateSimpleHypertextField

hiHypertextReadin

hiHypertextReplace

Cadence User Interface SKILL Reference

Form Functions

hiIgnoreProp

```
hiIgnoreProp(  
    s_objType  
    t_propName  
)  
=> t / nil
```

Description

Does not display the specified property of a given object type in the Property List Editor form.

This function can be called multiple times on the same object type so that multiple properties can be ignored.

Arguments

<i>s_objType</i>	Symbol corresponding to the database object's <code>objType</code> attribute. For example: <code>cellView</code> , <code>inst</code> , <code>pin</code> , <code>lib</code> , <code>version</code> .
<i>t_propName</i>	Name of the property to ignore. If this property has been stored on an object of the specified <i>s_objType</i> , it is not displayed. Object attributes cannot be ignored.

Values Returned

<i>t</i>	The property is not displayed.
<i>nil</i>	The object type or property name is not valid. A warning is also issued.

Examples

The properties `privateProp` and `ignoreMe` are not displayed when displaying the Property List Editor form for a database `cellview`.

```
hiIgnoreProp( 'cellView "privateProp" )  
hiIgnoreProp( 'cellView "ignoreMe" )  
hiEditPropList( cell )
```

Related Topics

Form and Field Functions

Cadence User Interface SKILL Reference

Form Functions

hiInFormApply

```
hiInFormApply(  
    r_form  
)  
=> t / nil
```

Description

Checks whether the `Apply` button on a mapped form is selected.

Argument

<i>r_form</i>	Name of the form.

Values Returned

<code>t</code>	The <code>Apply</code> button on a mapped form is selected.
<code>nil</code>	The <code>Apply</code> button is not selected or the form is not mapped.

Related Topics

[Form and Field Functions](#)

hiInsertItemToLayout

```
hiInsertItemToLayout(  
    g_formRegionOrLayout  
    g_item  
    [ ?layoutControls l_layoutControls ]  
    [ ?index x_index ]  
    [ ?toLayout s_layoutName ]  
)  
=> t / nil
```

Description

Inserts the required controls to the layout.

Arguments

g_formRegionOrLayout

Any of the following:

- An instantiated hiLayout form
- Scroll region or tab page in hiLayout form
- hiLayout

g_item

A field, hiLayout, or spacer item to be added to the layout.

The spacer items are:

- 'col_min_width *x_column* *x_width*: Specifies the minimum width for a column.
- 'row_min_height *x_row* *x_height*: Specifies the minimum height for a row.
- 'col_stretch *x_column* *x_stretch_factor*: Specifies the horizontal stretch factor for a column.
- 'row_stretch *x_row* *x_stretch_factor*: Specifies the vertical stretch factor for a row.

?layoutControls *l_layoutControls*

Cadence User Interface SKILL Reference

Form Functions

The controls that are appropriate to the layout type to which the items are being added. If these items are being added to a grid layout, then the row and column controls must be specified.

The required layout controls are:

- `'row x_row`: Specifies the row in which the field or layout is placed. The topmost row is 0.
- `'col x_column`: Specifies the column in which the field or layout is placed. The leftmost column is 0.
- The optional layout control is: `'prompt t | nil | t_prompt`:
 - If `t` is specified then the label portion of the form layout row gets occupied by the field prompt (if any) or an empty string if either the field has no prompt or it is a layout object.
 - If `nil` is specified then the field prompt is ignored (if any) and the field or layout spans both the label and field column.
 - If `t_prompt` is specified then the label for the row and any prompt defined for the field is ignored.
 - If no value is specified then the field prompt is used (if any) or a blank prompt is assumed.

`?index x_index` The index position in which the item is to be inserted.

`?toLayout s_layoutName`

The `hiLayoutSymbol` for an `hiLayout` contained with the form or region. This argument is ignored if `g_formRegionOrLayout` is a `hiLayout`.

If `g_formRegionOrLayout` is a form or region and `s_layoutName` is not specified, the main `hiLayout` of the form or region is used.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The operation is successfully completed.
nil	The operation is not completed.

Related Topics

Form and Field Functions

hiInstantiateForm

```
hiInstantiateForm(  
    r_form  
)  
=> t / nil
```

Description

Instantiates the C structure of a form from SKILL.

If a form is uninstantiated, field callbacks do not get called, since the form is simply a SKILL structure at this point. Instantiating the form gives you full form functionality, but the form does not get displayed.

Argument

<i>r_form</i>	A form returned by <code>hiCreateAppForm</code> or <code>hiCreateForm</code> .

Value Returned

t	The form is instantiated.
nil	An error occurred. An error message is displayed.

Examples

This example instantiates the form `myform`.

```
hiInstantiateForm( myform ) => t
```

Related Topics

[hiCreateAppForm](#)

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiIsForm

```
hiIsForm(  
    g_FormOrMenu  
)  
=> t / nil
```

Description

Checks whether a form or menu is a valid one.

Argument

<i>g_FormOrMenu</i>	Form or menu whose validity is being checked.

Value Returned

t	The form or menu is valid.
nil	The form or menu is not valid.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiIsFormDisplayed

```
hiIsFormDisplayed(  
    r_form  
)  
=> t / nil
```

Description

Checks whether a form is displayed.

Argument

<i>r_form</i>	Form being checked for visibility.

Values Returned

t	The form is displayed.
nil	The form is not displayed or is not instantiated.

Examples

```
*****  
;Sample form and fields  
*****  
  
; Create string entry for the form.  
procedure( myformDefineFields()  
    strfldglb = hiCreateStringField(  
        ?name `strfldglb  
        ?prompt "sample string field"      ;;; FIELD1, a string type  
        ?defValue "initial string"  
        ?callback "my_strfldglb_cb()"   
        ?editable t  
    )  
  
    ; Create a Boolean button for the form.  
    boolfldglb = hiCreateBooleanButton(  
        ?name `boolfldglb  
        ?buttonText "sample boolean field"      ;;; FIELD2, a boolean type
```

Cadence User Interface SKILL Reference

Form Functions

```
?defValue t
?callback "my_boolfldglb_cb()"
)
list( strfldglb boolfldglb )
)
; Generate the form

procedure( myformcreateform()
  if( !boundp(`myformglb) || (myformglb == nil)
    then
      myformglb = hiCreateAppForm(
        ?name `myformglb
        ?fields myformDefineFields()
        ?formTitle "My first Form"
        ?callback "myformglb_cb()"
        ?buttonLayout `OKCancel
      )
    )
  myformglb
)

; Display the form
procedure( myformdisplayform()
  myformcreateform()      ;;; Creation and Display are separate
  hiDisplayForm(`myformglb)
)

; Check the current state of the form
hiIsFormDisplayed(myformglb) =>nil

myformdisplayform()
hiIsFormDisplayed(myformglb)
=>t
```

Related Topics

Form and Field Functions

Cadence User Interface SKILL Reference

Form Functions

hiIsInstantiated

```
hiIsInstantiated(  
    g_formRegionOrMenu  
)  
=> t / nil
```

Description

Displays a form at least once or a menu has been placed in a window. Returns `nil` if the form has been deleted or if *g_formRegionOrMenu* is not valid.

Argument

g_formRegionOrMenu

A form, scroll region, tab page, or menu.

Values Returned

<code>t</code>	The item has been displayed and/or can be used interactively.
<code>nil</code>	The item is not instantiated.

Examples

After creating a form, scroll region, tab page, or menu, the item only exists as a SKILL construct and does not verify changes made to it or its fields/menu items, and does not call callbacks when changes are made.

After the item is displayed the first time or otherwise explicitly instantiated (made interactive), `hiIsInstantiated` returns `t` and the callbacks works until the item such as form is deleted.

```
MyForm = hiCreateAppForm( ?name 'MyForm ?dontBlock t <other form arguments> )  
=> formStruct@0x8d0aea8  
hiIsInstantiated( MyForm )  
=> nil  
hiDisplayForm( MyForm )  
=> t  
hiIsInstantiated( MyForm )  
=> t
```

Cadence User Interface SKILL Reference

Form Functions

```
hiFormDone( MyForm )  
=> t  
hiIsInstantiated( MyForm )  
=> t  
hiDeleteForm( MyForm )  
=> t  
hiIsInstantiated( MyForm )  
=> nil
```

Related Topics

[Form and Field Functions](#)

hiIsInFieldCancel

```
hiIsInFieldCancel(  
    )  
=> r_formHandle / nil
```

Description

Query this function during a field callback to determine if the form is currently being canceled, the callback may be initiated by the cancel operation. If the form is being canceled, the form handle is returned; otherwise `nil` is returned.

Arguments

None

Values Returned

<i>r_formHandle</i>	The handle to the form if the form is currently being canceled.
<code>nil</code>	The form is not being canceled.

Related Topics

[Form and Field Functions](#)

hiItemInsertTree

```
hiItemInsertTree(  
    g_item  
    g_tree  
)  
=> t / nil
```

Description

Creates a sub-tree under an item. You need to specify the item and the sub-tree as the arguments for this function.

Arguments

<i>g_item</i>	The item under which you want to create a sub-tree. This is the return value of <code>hiCreateTreeItem</code> .
<i>g_tree</i>	The sub-tree. Use <code>hiCreateTree</code> to create the sub-tree and provide its return value as this argument.

Values Returned

<i>t</i>	The tree was inserted under the item.
<i>nil</i>	The tree was not inserted under the item.

Related Topics

[Form and Field Functions](#)

[hiCreateTree](#)

[hiCreateTreeItem](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

[hiItemRemoveTree](#)

[hiCreateTreeTable](#)

hiLayerMatchCyclicStr

```
hiLayerMatchCyclicStr(  
    l_choices  
    t_layer  
)  
=> l_layerValue
```

Description

Returns the layer cyclic value associated with the specified *t_layer*.

This is a convenience function that is used to search the specified list of icon layer choices (*l_choices*) for a value matching the *t_layer* string. This function should only be used for layer cyclic fields.

Arguments

<i>l_choices</i>	List of icon layer lists created from a call to <code>hiCreateLayerCyclicField</code> . This list can be retrieved from <code>form->hiLayerField->choices</code> .
<i>t_layer</i>	String representing the layer you wish to match. This string is in the form of <i>t_layerName (t_layerPurpose)</i> . (This format is equivalent to the fourth element of a layer icon value.)

Value Returned

<i>l_layerValue</i>	The cyclic value of the specified layer.

Related Topics

[Form and Field Functions](#)

[hiCreateLayerCyclicField](#)

Cadence User Interface SKILL Reference

Form Functions

hiLaunchBrowser

```
hiLaunchBrowser(  
    t_url  
    [ g_browserHint ]  
)  
=> t / nil
```

Description

Starts a Web browser, if needed, and opens the URL you specify.

If a browser window is already open, `hiLaunchBrowser` uses that browser and opens the specified URL in a new or existing window or tab, based on the value of the `g_browserHint` argument. If a browser is not already open, `hiLaunchBrowser` starts the browser specified in the *Web Browser* field of the User Preferences form.

The User Preferences form contains the value of the `ui webBrowser` variable set in the registry or custom `.cdsenv` file. (If you make a change in the form and save your defaults, the `.cdsenv` file is updated.)

If a `CDS_DEFAULT_BROWSER` environment variable is set, Virtuoso applications, when they start, they set the value of that variable in the User Preferences form (overriding the value read from the `.cdsenv` file). Therefore, that value is used by `hiLaunchBrowser`.

`hiLaunchBrowser` checks for and launches only Mozilla-based browsers: Netscape, Firefox, and Mozilla.

Arguments

<code>t_url</code>	The URL that you want the browser to display, specified as a string.
<code>g_browserHint</code>	One of the following symbols, specifying where to open the URL: 'newWindow: Opens URL in a new window. 'newTab: Open URL in a new tab. 'newLocation: Open URL in an existing window and tab. Default value: 'newLocation

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The browser was started.
nil	There was an error.

Related Topics

[Form and Field Functions](#)

[hiCreateHypertextField](#)

[hiCreateSimpleHypertextField](#)

[hiHypertextReadin](#)

[hiHypertextReplace](#)

[hiHypertextScroll](#)

hiLayerStringToLPP

```
hiLayerStringToLPP(  
    d_techFileId  
    t_layer  
)  
=> d_layerId / nil
```

Description

Returns the layer object in the specified *d_techFileId* associated with the specified *t_layer*.

This function is used in conjunction with layer cyclic fields.

Arguments

<i>d_techFileId</i>	The techfile object associated with the layers of the cyclic field.
<i>t_layer</i>	String representing the layer you wish to convert. This string is in the form of <i>t_layerName (t_layerPurpose)</i> . This format is equivalent to the fourth element of a layer icon value.

Values Returned

<i>d_layerId</i>	The corresponding database layer object.
<i>nil</i>	The layer is not found in the specified techfile.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiMoveField

```
hiMoveField(  
    r_2DFormOr2DMenu  
    s_field  
    l_location  
)  
=> t / nil
```

Description

Moves the specified field to a new location. This function cannot be used for fields that have field attachments. Use `hiReattachField` for fields with field attachments.

Arguments

<i>r_2DFormOr2DMenu</i>	2D form or menu returned from <code>hiCreateAppForm</code> , <code>hiCreateForm</code> , or <code>hiCreate2DMenu</code> .
<i>s_field</i>	SKILL symbol representing the field in <i>r_form</i> you wish to move.
<i>l_location</i>	Location to which you want to move the field. The location is relative to the upper-left corner of the form. <i>location</i> is of the form: <code>list(x_xLoc x_yLoc)</code> or <code>x_xLoc : y_yLoc</code>

Values Returned

<i>t</i>	The field is moved.
<i>nil</i>	The field is not moved. An error message is displayed.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiCreateForm](#)

hiMoveToFormField

```
hiMoveToFormField(  
    g_formOrRegion  
    s_field  
)  
=> moved / focused / nil
```

Description

Sets the input focus to the form, scrolls the form to bring the given field into the view, and warps the pointer to the field.

This SKILL function works only if the specified form is currently displayed, and the given field exists on the form or region, such as a scroll region, a tab field, or a tab page.

If the specified field is a text field and can accept input focus, it is set as the current field and the input focus is set to it.

Arguments

<i>g_formOrRegion</i>	The form or a region on the form.
<i>s_field</i>	The symbol name of the field.

Values Returned

<i>moved</i>	The input focus is set successfully to the form or region, but not to the field because it is not a text field.
<i>focused</i>	The input focus is set successfully to the specified field on the given form or region.
<i>nil</i>	The input focus could not be set to the specified form or region, and the pointer was not warped because the given field does not exist (or is invisible).

Examples

Assuming that the User Preferences form is displayed, the following command moves the pointer to the *Web Browser* field and set the input focus to it:

```
hiMoveToFormField( hiSysProps 'hiWebBrowser )
```

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

hiMoveInsBarToEnd

```
hiMoveInsBarToEnd(  
    r_form  
    s_field  
)  
=> t / nil
```

Description

Places the cursor (insertion point) at the end of a text field, scrolling the field so the cursor is visible.

This function does not make the field active if it is inactive.

The field must be instantiated and must be the current field with input focus.

Arguments

<i>r_form</i>	Form containing the field where the cursor is to be placed.
<i>s_field</i>	Field where the cursor is placed. The field must be a type-in field: string, float, integer, list. Must exist in the specified form.

Values Returned

t	The cursor was placed at the end of the field.
nil	The field is not on the form or is not a type-in field. An error message is also issued.

Examples

```
hiMoveInsBarToEnd( myform `myTypeinField )  
=> t
```

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiOffsetField

```
hiOffsetField(  
    r_2DFormOr2DMenu  
    s_field  
    l_offsets  
)  
=> t / nil
```

Description

Offsets one field from its current position. This function cannot be used for fields that have field attachments. Use `hiReattachField` for fields with field attachments.

This routine adds the x and y offset values and the current x and y values to determine the new field position. An offset of 0 has no effect. Negative offsets may be specified, but the minimum value of the new x and y is 0. Both offset values are required.

Use this function to offset a single field. To offset more than one field, use [hiOffsetFields](#).

The form must be instantiated.

Arguments

<i>r_2DFormOr2DMenu</i>	Two-dimensional form or menu returned from <code>hiCreateAppForm</code> , <code>hiCreateForm</code> , or <code>hiCreate2DMenu</code> .
<i>s_field</i>	SKILL symbol representing the field in <i>r_form</i> you wish to offset.
<i>l_offsets</i>	Distance in pixels the field is set off from the current position. <i>offsets</i> is of the form: <code>list(x_xOffset x_yOffset)</code> or <code>x_xOffset : x_yOffset</code>

Cadence User Interface SKILL Reference

Form Functions

Value Returned

<code>t</code>	The field is offset.
<code>nil</code>	The field is not offset. An error message is also issued.

Examples

Offsets the `'myField` field by `10:20` pixels from the current location.

```
hiOffsetField ( testform 'myField 10:20 )  
=> t
```

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiCreateForm](#)

hiOffsetFields

```
hiOffsetFields(  
    r_2DFormOr2DMenu  
    l_fields  
    l_offsets  
)  
=> t / nil
```

Description

Offsets several fields from their current positions. All fields are offset by the same distance, they move as a block. This function cannot be used for fields that have field attachments. Use `hiReattachField` for fields with field attachments.

This routine adds the `x` and `y` offset values and the current `x` and `y` values to determine the new field position. An offset of 0 have no effect. Negative offsets can be specified, but the minimum value of the new `x` and `y` is 0. Both offset values are required.

The form must be instantiated.

To offset only one field, use `hiOffsetField`.

Arguments

<i>r_2DFormOr2DMenu</i>	Two-dimensional form or menu returned from <code>hiCreateAppForm</code> , <code>hiCreateForm</code> , or <code>hiCreate2DMenu</code> .
<i>l_fields</i>	List representing the fields in <i>r_form</i> you wish to offset.
<i>l_offsets</i>	Distance in pixels the field is set off from the current position. <i>offsets</i> is of the form: <pre>list(x_xOffset x_yOffset) or x_xOffset : x_yOffset</pre>

Cadence User Interface SKILL Reference

Form Functions

Value Returned

t	The fields are offset.
nil	The fields are not offset. An error message is also issued.

Examples

Offsets the fields `\(myField myField2 )` by `-10:0` pixels from the current location

```
hiOffsetFields ( testform \(myField myField2 ) -10:0 )  
=> t
```

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiCreateForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiResizeField

```
hiResizeField(  
    r_2DForm  
    s_field  
    l_resizeDescription  
)  
=> t / nil
```

Description

Changes the width, height, or *promptBox* width of a specified field. This function cannot be used for fields that have field attachments. Use `hiReattachField` for fields with field attachments.

Arguments

<i>r_2DForm</i>	2D form returned from <code>hiCreateAppForm</code> or <code>hiCreateForm</code> .
<i>s_field</i>	SKILL field handle representing the field you wish to resize.
<i>l_resizeDescription</i>	

New dimensions, in the following form:

```
list( x_width x_height [x_promptWidth])
```

x_width and *x_height*: The new width and height of the field.

x_promptWidth: Optional, and is considered only if a prompt exists. If this value is not specified for a field which has a prompt, the prompt box for the field does not change.

For separator fields the height is ignored and the width as the length of the separator whether it is vertical or horizontal.

Values Returned

t	The field characteristics are changed.
nil	The field characteristics are not changed. An error message is also issued.

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiReattachField

```
hiReattachField(  
    g_formOrScrollRegion  
    s_field  
    l_positionalInfo  
    i_fieldAttachment  
)
```

Description

Reattaches a field in a two-dimensional form or scroll region that has field attachments. This function places the field according to the new position and attachment information. This function can be used only with forms or scroll regions that already have field attachments. For forms or scroll regions that do not have field attachments, use the `hiMoveField`, `hiResizeField`, `hiOffsetField`, and `hiOffsetFields` functions.

Arguments

g_formOrScrollRegion

Handle to the two-dimensional form or scroll region that contains the field you want to reattach.

s_field

Handle to the field you want to reattach.

i_positionalInfo

The two-dimensional attributes for the field. Use the following syntax:

```
list ( x:y width:height [  
    x_promptBoxWidthOrTitleHeight ] )
```

i_fieldAttachment

A bitwise OR of a list of constants. The field attachment determines how the two-dimensional attributes of a field are used to position the field in the form. This argument is valid only for two-dimensional forms.

Values Returned

t

The field characteristics are changed.

nil

The field characteristics are not changed. An error message is also issued.

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[Two Dimensional Form Layout](#)

[Field Attachments](#)

hiReportSelectItem

```
hiReportSelectItem(  
    g_reportField  
    g_itemIndex  
    [ g_notify ]  
)  
=> t / nil
```

Description

Selects an item in a report field. This function also executes the callback of the report field if you specify `t` as the value of the `g_notify` argument.

This function is the programmatic equivalent of selecting an item with the mouse or keyboard keys.

To select more than one item, use `hiReportSelectItems`. To select all items in a field, use `hiReportSelectAllItems`.

Arguments

<code>g_reportField</code>	The report field that contains the item.
<code>g_itemIndex</code>	The item you want to select, specified as an integer indicating the item's position in <code>?choices</code> . The first item in <code>?choices</code> is 0, the second is 1, and so on. The <code>?choices</code> argument is specified when the field is created with <code>hiCreateReportField</code> .
<code>g_notify</code>	Triggers the callback procedure of the report field. The default value is <code>nil</code> .

Values Returned

<code>t</code>	The item was selected.
<code>nil</code>	The item was not selected.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateReportField

hiReportSelectItems

hiReportSelectAllItems

hiReportDeselectItem

hiReportDeselectItems

hiReportDeselectAllItems

hiReportGetSelectedItems

hiCreateReportField

hiReportSelectItems

```
hiReportSelectItems(  
    g_reportField  
    l_itemIndexList  
    [ g_notify ]  
)  
=> t / nil
```

Description

Selects items in a report field. This function also executes the callback of the report field if you specify *t* as the value of the *g_notify* argument.

This function is the programmatic equivalent of selecting items with the mouse or keyboard keys.

To select more than one item, use `hiReportSelectItems`. To select all items in a field, use `hiReportSelectAllItems`.

Arguments

<i>g_reportField</i>	The report field that contains the items.
<i>l_itemIndexList</i>	The items you want to select, specified as a list of integers indicating the items' positions in <i>?choices</i>. The first item in <i>?choices</i> is 0, the second is 1, and so on. You can specify a range of items as a list within the list. For example: <pre>'(1 3 6); selects items 1, 3, 6 '('(2 4)); selects items 2, 3, 4 '('(1 3) 9 11); selects items 1, 2, 3, 9, 11</pre> The <i>?choices</i> argument is specified when the field is created with <code>hiCreateReportField</code>.
<i>g_notify</i>	Triggers the callback procedure of the report field. The default value is <code>nil</code>.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The items were selected.
nil	The items were not selected.

Related Topics

[Form and Field Functions](#)

[hiReportSelectItem](#)

[hiReportSelectAllItems](#)

[hiReportDeselectItem](#)

[hiReportDeselectItems](#)

[hiReportDeselectAllItems](#)

[hiReportGetSelectedItems](#)

[hiCreateReportField](#)

hiReportSelectAllItems

```
hiReportSelectAllItems(  
    g_reportField  
    [ g_notify ]  
)  
=> t / nil
```

Description

Selects all items in a report field. This function also executes the callback of the report field if you specify *t* as the value of the *g_notify* argument.

This function is the programmatic equivalent of selecting items with the mouse or keyboard keys.

To select more than one item, use `hiReportSelectItems`. To select all items in a field, use `hiReportSelectAllItems`.

Arguments

<i>g_reportField</i>	The report field that contains the items.
<i>g_notify</i>	Triggers the callback procedure of the report field. The default value is <code>nil</code> .

Value Returned

<i>t</i>	The items were selected.
<code>nil</code>	The items were not selected.

Related Topics

[Form and Field Functions](#)

[hiReportSelectItem](#)

[hiReportDeselectItem](#)

[hiReportDeselectItems](#)

[hiReportDeselectAllItems](#)

Cadence User Interface SKILL Reference

Form Functions

hiReportGetSelectedItems

hiCreateReportField

hiReportDeselectItem

```
hiReportDeselectItem(  
    g_reportField  
    g_itemIndex  
    [ g_notify ]  
)  
=> t / nil
```

Description

Deselects an item in a report field. This function also executes the callback of the report field if you specify *t* as the value of the *g_notify* argument.

To deselect more than one item, use `hiReportDeselectItems`. To deselect all selected items in a field, use `hiReportDeselectAllItems`.

This function is the programmatic equivalent of deselecting items with the mouse or keyboard keys.

Arguments

<i>g_reportField</i>	The report field that contains the item.
<i>g_itemIndex</i>	The item you want to deselect, specified as an integer indicating the item's position in <i>?choices</i> . The first item in <i>?choices</i> is 0, the second is 1, and so on. The <i>?choices</i> argument is specified when the field is created with <code>hiCreateReportField</code> .
<i>g_notify</i>	Triggers the callback procedure of the report field. The default value is <i>nil</i> .

Values Returned

<i>t</i>	The item was deselected
<i>nil</i>	The item was not deselected.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiReportSelectItem

hiReportDeselectItems

hiReportDeselectAllItems

hiReportGetSelectedItems

hiCreateReportField

hiReportDeselectItems

```
hiReportDeselectItems(  
    g_reportField  
    l_itemIndexList  
    [ g_notify ]  
)  
=> t / nil
```

Description

Deselects items in a report field. This function also executes the callback of the report field if you specify *t* as the value of the *g_notify* argument.

To deselect more than one item, use `hiReportDeselectItems`. To deselect all selected items in a field, use `hiReportDeselectAllItems`.

This function is the programmatic equivalent of deselecting items with the mouse or keyboard keys.

Arguments

<i>g_reportField</i>	The report field that contains the items.
<i>l_itemIndexList</i>	The items you want to deselect, specified as a list of integers indicating the items' positions in <i>?choices</i>. The first item in <i>?choices</i> is 0, the second is 1, and so on. You can specify a range of items as a list within the list. For example: <pre>'(1 3 6); deselects items 1, 3, 6 '('(2 4)); deselects items 2, 3, 4 '('(1 3) 9 11); deselects items 1, 2, 3, 9, 11</pre> The <i>?choices</i> argument is specified when the field is created with <code>hiCreateReportField</code>.
<i>g_notify</i>	Triggers the callback procedure of the report field. The default value is <code>nil</code>.

Cadence User Interface SKILL Reference

Form Functions

Value Returned

t	The items were deselected.
nil	The items were not deselected.

Related Topics

[Form and Field Functions](#)

[hiReportSelectItem](#)

[hiReportDeselectItems](#)

[hiReportDeselectAllItems](#)

[hiReportGetSelectedItems](#)

[hiCreateReportField](#)

hiReportDeselectAllItems

```
hiReportDeselectAllItems(  
    g_reportField  
    [ g_notify ]  
)  
=> t / nil
```

Description

Deselects all selected items in a report field. This function also executes the callback of the report field if you specify *t* as the value of the *g_notify* argument.

To deselect more than one item, use `hiReportDeselectItems`. To deselect all selected items in a field, use `hiReportDeselectAllItems`.

This function is the programmatic equivalent of deselecting items with the mouse or keyboard keys.

Arguments

<i>g_reportField</i>	The report field.
<i>g_notify</i>	Triggers the callback procedure of the report field. The default value is <code>nil</code> .

Value Returned

<i>t</i>	The items were deselected.
<code>nil</code>	The items were not deselected.

Related Topics

[Form and Field Functions](#)

[hiReportSelectItem](#)

[hiReportSelectItems](#)

[hiReportSelectAllItems](#)

[hiReportDeselectItem](#)

Cadence User Interface SKILL Reference

Form Functions

hiReportDeselectItems

hiReportGetSelectedItems

hiCreateReportField

hiReportGetSelectedItems

```
hiReportGetSelectedItems(  
    g_reportField  
)  
=> l_selectedItemsIndex / nil
```

Description

Gets a list of all the items that are currently selected in a report field.

Argument

<i>g_reportField</i>	The report field.

Value Returned

l_selectedItemsIndex

List of integers representing the items that are currently selected in the report field. The integers represent the positions of the selected items in `?choices`. The first item in `?choices` is 0, the second is 1, and so on. The `?choices` argument is specified when the field is created with `hiCreateReportField`.

<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiReportSelectItem](#)

[hiReportSelectItems](#)

[hiReportSelectAllItems](#)

[hiReportDeselectItem](#)

[hiReportDeselectItems](#)

[hiReportDeselectAllItems](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateReportField

hiSetButtonIcon

```
hiSetButtonIcon(  
    g_button  
    g_icon  
)  
=> t / nil
```

Description

Changes a button icon after the button is created. This function updates all instances of the button.

hiSetButtonIcon does not change an instance original size.

Arguments

<i>g_button</i>	The handle to the button returned by <code>hiCreateButton</code> .
<i>g_icon</i>	The new icon.

Values Returned

<i>t</i>	The icon was changed.
<i>nil</i>	The icon was not changed.

Related Topics

[Form and Field Functions](#)

[hiCreateButton](#)

[hiSetButtonLabel](#)

hiSetButtonLabel

```
hiSetButtonLabel(  
    g_Button  
    g_Label  
)  
=> t / nil
```

Description

Changes a button label text or icon after the button is created. This function updates all instances of the button.

`hiSetButtonLabel` does not change an instance's original size.

Arguments

<i>g_Button</i>	The handle to the button returned by <code>hiCreateButton</code> .
<i>g_Label</i>	The new label text (string) or icon. If you specify a string, this function always assumes it is the label text and displays it accordingly. If you want to specify an icon as a string (the icon file name), use the function <code>hiSetButtonIcon</code> instead.

Values Returned

<i>t</i>	The button label was changed.
<i>nil</i>	The button label was not changed.

Related Topics

[Form and Field Functions](#)

[hiCreateButton](#)

[hiSetButtonIcon](#)

hiSetCallbackStatus

```
hiSetCallbackStatus(  
    r_form  
    g_booleanStatus  
)  
=> t / nil
```

Description

Overrides the removal (unmapping) of a form when the form's *OK* button is pressed.

This function can be used in the form's callback and is useful if you enter invalid form data and the application instructs you to re-enter a response.

To use this function, you must set the *unmapAfterCB* flag argument of *hiCreateAppForm* to *t*.

Arguments

<i>r_form</i>	Form description returned from <i>hiCreateAppForm</i> or <i>hiCreateForm</i> .
<i>g_booleanStatus</i>	Desired status of the form. This <i>booleanStatus</i> is a toggle value, and must be reset to <i>t</i> once it has been set to <i>nil</i> . The form will not be brought down until this value is reset. This routine should only be used with standard forms; the user should never try to override the actions of an options form. Each form will have its initial status set (default is <i>t</i>) to hide the form when <i>OK</i> is pressed. This value is reset to <i>t</i> if the user cancels the form.

Value Returned

<i>t</i>	The function was successful.
<i>nil</i>	An error occurred. An error message is also issued.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateForm

hiSetCurrentField

```
hiSetCurrentField(  
    g_form  
    s_fieldName  
    [ g_limitSearch ]  
)  
=> t / nil
```

Description

Makes the specified type-in field the current field (sets input focus to the field). If the field is not visible, *hiSetCurrentField* scrolls the form or scroll region field so the field becomes visible.

The fields that accept focus are determined by the *?fieldFocus* argument of the *hiCreateAppForm* function.

The function first searches for the field at the top level of the form or scroll region field specified. If the field is not found at the top level, all scroll region fields within the form or scroll region field are searched hierarchically. The first field that matches *s_fieldName* is set as the current field. You can limit the search to only the top level of the form or scroll region field by specifying a non-*nil* value for the third argument, *limitSearch*.

This function can only be used when the form is visible on the screen.

Arguments

<i>g_form</i>	The form or scroll region field in which you want to set the current field.
<i>s_fieldName</i>	Field to be made current. If the field is not an editable text field, it is scrolled into view but not given focus. If the field is not found at the top-level of the form or scroll region field, all scroll region fields within the form or scroll region field are searched hierarchically. The first match is selected.
<i>g_limitSearch</i>	Any non- <i>nil</i> value. Use this argument if you want to limit the search for the field to only the top level of the form or scroll region field specified. If this argument is not used, the function first searches the top level of the form or scroll region field and then, if the field is not found, does a hierarchical search of all scroll region fields within the form or scroll region field.

Cadence User Interface SKILL Reference

Form Functions

Values Returned

t	The field was made current and input focus was set to the field.
nil	Input focus could not be set to the field.

Related Topics

Form and Field Functions

hiCreateAppForm

Cadence User Interface SKILL Reference

Form Functions

hiSetCurrentForm

```
hiSetCurrentForm(  
    g_form  
    [ g_setIfNotDisplayed [ g_setIfNotInstantiated ] ]  
)  
=> t / nil
```

Description

Makes the form you specify the current form. If the form is displayed, moves the focus to the form.

Arguments

g_form Form handle returned by `hiCreateAppForm` or the symbol that was set by the *?name* argument of `hiCreateAppForm` when the form was created.

g_setIfNotDisplayed Displays the current form if it is instantiated or if not instantiated and *g_setIfNotInstantiated* is set to `t`.

If the form is not displayed and *g_setIfNotDisplayed* is `nil` or not specified, then a warning is displayed and the function returns `nil` without changing the current form.

g_setIfNotInstantiated Displays the current form even if it is not instantiated.

Value Returned

`t` The specified form is the current form.

`nil` The specified form is not instantiated and either *g_setIfNotDisplayed* or *g_setIfNotInstantiated* is `nil`. Invalid ID or the form is not displayed and *g_setIfNotDisplayed* is `nil`.

Related Topics

Form and Field Functions

Cadence User Interface SKILL Reference

Form Functions

hiCreateAppForm

hiSetFieldEditable

```
hiSetFieldEditable(  
    r_field  
    g_editable  
)  
=> t / nil
```

Description

Sets the specified field editable or noneditable, in all forms where it appears.

This function should be used after the form containing the field has been created. You can also make a field editable or noneditable using:

```
form->field->editable=t/nil
```

Arguments

<i>r_field</i>	Field must be a text-entry (type-in) field returned from a field creation routine. If the field is already placed in a form, the field must be specified through the form handle, that is, <code>form->field</code> .
<i>g_editable</i>	Makes the field editable. Read-only fields can accept focus through tabbing or clicking the mouse only if the <code>?fieldFocus</code> argument of <code>hiCreateAppForm</code> is set to <code>'allTypein</code> . On eight-plane displays, the text entry area of non-editable fields is greyed out and the field prompt is stippled.

Value Returned

t	The editable flag is set.
nil	The editable flag is not set. An error message is displayed.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

hiSetFieldEnabled

```
hiSetFieldEnabled(  
    g_field  
    g_enabled  
)  
=> t / nil
```

Description

Enables or disables a form field. A disabled field is greyed-out. This function updates all instances of the field. You can use `hiSetFieldEnabled` with all types of fields except separator fields.

You can also enable or disable a field with the following property:

```
form->field->enabled = t | nil
```

To enable or disable buttons on a form, use `hiSetFormButtonEnabled`.

Arguments

<i>g_field</i>	Handle to field returned from functions that create fields.
<i>g_enabled</i>	The field is enabled.

Values Returned

t	The field was changed according to your specification.
nil	The field was not changed.

Related Topics

[Form and Field Functions](#)

[hiSetFormButtonEnabled](#)

hiSetFormButtonEnabled

```
hiSetFormButtonEnabled(  
    r_form  
    s_buttonSym  
    g_enabled  
)  
=> t / nil
```

Description

Enables or disables a button on a form. To enable or disable fields on a form, use the `hiSetFieldEnabled` function.

Arguments

<i>r_form</i>	Form handle returned by <code>hiCreateAppForm</code> .
<i>s_buttonSym</i>	Button symbol.
<i>g_enabled</i>	<code>t</code> or <code>nil</code> . If you specify <code>t</code> , the button is enabled; if you specify <code>nil</code> , the button is disabled, that is, greyed-out.

Values Returned

<code>t</code>	The function was successful.
<code>nil</code>	There was an error.

Examples

```
hiSetFormButtonEnabled( myForm 'Apply nil )  
hiSetFormButtonEnabled( myForm 'Prev nil )  
hiSetFormButtonEnabled( myForm 'Help nil )
```

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

hiSetFormBlock

```
hiSetFormBlock(  
    r_form  
    [ g_blockUpdates ]  
    [ g_unmanage ]  
)  
=> t / nil
```

Description

Prevents time-consuming multiple updates to a form during multiple changes (such as adding, deleting, moving, or resizing elements). You can also blank out the form during extensive changes. Updates are blocked until you call `hiSetFormBlock(r_form nil)` or `hiUpdateFormBlock(r_form)`.

Use the `hiSetFormBlock` function only if you are making extensive changes to a form.

Arguments

<i>r_form</i>	Handle to the form, returned by <code>hiCreateAppForm</code> .
<i>g_blockUpdates</i>	Blocks update to the form. Specify <code>nil</code> to reset the form to allow updates. You can also reset the form with the <code>hiUpdateFormBlock</code> function.
<i>g_unmanage</i>	Blanks out the form during extensive changes. Specify <code>nil</code> when you reset the form to allow updates. You can also reset the form with the <code>hiUpdateFormBlock</code> function.

Values Returned

<i>t</i>	In all cases except when <i>r_form</i> is not valid.
<i>nil</i>	<i>r_form</i> is not valid.

Examples

```
hiSetFormBlock ( r_form t )  
<add/delete/move/resize functions>  
hiSetFormBlock ( r_form nil )
```


Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[hiUpdateFormBlock](#)

hiSetFormHighlights

```
hiSetFormHighlights(  
    g_form  
    g_highlightType  
)  
=> t / nil
```

Description

Highlights all type-in fields for *form*, depending on the value of *highlightType*.

Highlighting is allowed for type-in fields only, as these are the fields into which invalid data might be entered, or data may be required.

To find type-in fields, the function searches the form or scroll region field specified as *r_form*. It also does a hierarchical search of all scroll region fields within the form or scroll region field.

Arguments

<i>g_form</i>	Form or scroll region field.
<i>g_highlightType</i>	It can be either a symbol or an integer. As a symbol, it must be <code>error</code> , <code>highlight</code> , or <code>background</code> to set the background color to an error, highlight, or background (default) color, respectively. As an integer, it must be <code>hicError</code> , <code>hicHighlight</code> , or <code>hicBackground</code> .

Values Returned

<i>t</i>	The highlighting flag is set.
<i>nil</i>	The highlighting flag is not set. An error message is displayed.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

hiSetFormMinMaxSize

```
hiSetFormMinMaxSize(  
    r_form  
    g_minSize  
    g_maxSize  
)  
=> t / nil
```

Description

Sets the minimum or maximum size of the form. The size excludes the window manager title and border.

Arguments

<i>r_form</i>	Form whose minimum or maximum size you want to set.
<i>g_minSize</i>	Minimum size of the form, specified as a list of width and height or <i>width:height</i> ; or <i>nil</i> .
<i>g_maxSize</i>	Maximum size of the form, specified as a list of width and height or <i>width:height</i> ; or <i>nil</i> .

Values Returned

<i>t</i>	The minimum or maximum size of the form was set.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

hiSetFormName

```
hiSetFormName (  
    r_form  
    t_newTitle  
)  
=> t
```

Description

Changes the form title, which is displayed in the window frame banner. The function can be called by its alias `hiChangeFormTitle`.

Arguments

<i>r_form</i>	Specifies the form.
<i>t_newTitle</i>	New title.

Value Returned

t	The title is changed.

Related Topics

[Form and Field Functions](#)

[hiChangeFormTitle](#)

hiSetFormPosition

```
hiSetFormPosition(  
    l_location  
)  
=> t / nil
```

Description

Sets the default initial location for forms displayed without an explicit location argument.

It gets overridden by specifying a location to `hiDisplayForm`.

If the user has set the `formPlacement` resource to something other than the default, this location is not used.

The default value can be set in your `.Xdefaults` file with the line

```
Opus.formPlacement: default
```

or by setting the `formPlacement` property of the Command Interpreter Window to "default" with the line

```
hiGetCIWindow ( ) => formPlacement = "default"
```

Argument

<i>l_location</i>	Location being specified. This must be a list of two integers.

Value Returned

t	The form position is set.
nil	An error occurred. An error message is displayed.

Related Topics

[Form and Field Functions](#)

[hiDisplayForm](#)

Cadence User Interface SKILL Reference

Form Functions

hiSetFormSize

```
hiSetFormSize(  
    r_form  
    l_size  
    [ g_reattachFields ]  
)  
=> t / nil
```

Description

Resizes a form programmatically. If the optional argument *g_reattachFields* is non-`nil`, field attachments is recalculated based on the new form size and original field sizes.

Arguments

<i>r_form</i>	Handle to the form, which is returned by <code>hiCreateAppForm</code> .
<i>l_size</i>	New size of the form, specified as <i>width:height</i> or a list of width and height. The size includes the field area and the button area of the form but excludes the window manager border.
<i>g_reattachFields</i>	Recalculates field attachments based on the new form size and original field sizes. If this argument is <code>nil</code> or unspecified, fields with field attachments are resized in the same way as they are when you resize the form manually. This argument only applies to forms with field attachments and is ignored for non-attached forms.

Values Returned

<code>t</code>	The form was resized.
<code>nil</code>	The form could not be resized.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

hiSetFormToDefaults

```
hiSetFormToDefaults(  
    r_form  
)  
=> t / nil
```

Description

Sets each field in the form to its default value, except the disclosure field value.

Callbacks for the fields is called if it results in the value of a field change.

Argument

<i>r_form</i>	Form to reset.

Value Returned

t	The default values are set.
nil	An error occurred. An error message is displayed.

Related Topics

[Form and Field Functions](#)

hiSetInsertionPosition

```
hiSetInsertionPosition(  
    g_form  
    s_field  
    x_newPosition  
)  
=> t / nil
```

Description

Resets the position of the insertion cursor in a type-in field. You can use the `hiGetInsertionPosition` function to get the current position of the cursor.

Arguments

<i>g_form</i>	Handle to the form, which is returned by <code>hiCreateAppForm</code> .
<i>s_field</i>	Symbol name of the field.
<i>x_newPosition</i>	The new position of the insertion cursor, specified as an integer. A value of 0 places the cursor at the beginning of the field, 1 places it after the first character, 2 places it after the second character, and so on. A negative number, or a number higher than the number of characters in the field, places the cursor at the end of the string.

Value Returned

<i>t</i>	The position of the insertion cursor was changed.
<i>nil</i>	The position of the insertion cursor could not be changed.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiGetInsertionPosition](#)

hiSetLayerCyclicValue

```
hiSetLayerCyclicValue(  
    d_techFileId  
    l_lpp  
    r_form  
)  
=> t / nil
```

Description

Sets the current value of the layer cyclic field.

This function should only be called if the form (*r_form*) contains a layer cyclic field.

Arguments

<i>d_techFileId</i>	The techfile object associated with the layers of the cyclic field.
<i>l_lpp</i>	List containing the layer name (or number) and purpose. This layer must exist in <i>d_techFileId</i> . (This argument is what is specified as the second argument to <code>dbGetLayer</code> .)
<i>r_form</i>	Form containing the layer cyclic field. This form handle is returned from <code>hiCreateAppForm</code> .

Value Returned

<i>t</i>	The value is set.
<i>nil</i>	An error occurred, or the layer-purpose pair was not found in the techfile. An error message is displayed.

Related Topics

[hiCreateAppForm](#)

[hiCreateLayerCyclicField](#)

hiSetStackedLayoutCurrentIndex

```
hiSetStackedLayoutCurrentIndex(  
    x_index  
    g_formRegionOrLayout  
    [ s_layoutName ]  
)  
=> x_index / nil
```

Description

Sets the index of the current top item of a valid stacked hiLayout.

Arguments

<i>x_index</i>	The integer index of the item in the stacked layout that needs to be brought to the top.
<i>g_formRegionOrLayout</i>	A hiLayout form, a scroll region, or a tab page in the hiLayout form or hiLayout.
<i>s_layoutName</i>	The <code>hiLayoutSymbol</code> for hiLayout contained within the form or region. This argument is ignored if <i>g_formRegionOrLayout</i> is hiLayout.

Value Returned

<i>x_index</i>	The index of the current top item, if the arguments resolve to a valid stacked hiLayout.
<i>nil</i>	The arguments do not resolve to a valid stacked hiLayout.

Examples

Returns the new index of 4. The form, `MyForm`, with a main layout, `MyMainLayout`, is a stacked layout that has five items. The previous top item is index 0. Now, to change the top item to the last item, the index value is changed to 4.

```
hiSetStackedLayoutCurrentIndex( 4 MyForm 'MyMainLayout )
```

hiSetScrollBarValue

```
hiSetScrollBarValue(  
    g_formOrScrollRegion  
    l_newValues  
)  
=> l_values
```

Description

Sets the horizontal and vertical scroll bars in a form or scroll region field to the position you specify, therefore allowing you to display a specific part of the form or scroll region field programmatically. Use `hiGetScrollBarInfo` to get information about the form or scroll region field before you use this function.

Arguments

g_formOrScrollRegion

Handle to the form or scroll region field.

l_newValues

The new values, in pixels, for the scroll bars. Use the following syntax for *l_newValues*:

```
list ( [ nil | x_horzValue ] [ nil | x_vertValue ] )
```

The first element of the list is the new pixel value for the horizontal scroll bar or `nil` to leave the horizontal scroll bar position unchanged. The second element of the list is the new pixel value for the vertical scroll bar or `nil` to leave the scroll bar position unchanged.

To get the minimum and maximum limits for *x_horzValue* and *x_vertValue*, use the `hiGetScrollBarInfo` function. The minimum value is usually 0. The maximum value is the size of the form or scroll region field minus the slider size (the visible part of the form or scroll region field).

If you provide a value that is less than the minimum, the minimum value is used. If you provide a value that is more than the maximum (the size of the form or scroll region field minus the slider size), the maximum value is used.

If the vertical or horizontal scroll bar is not visible, the new value for that scroll bar is not set.

Cadence User Interface SKILL Reference

Form Functions

Value Returned

l_values

The scroll bar values in the following form:

```
list ( [ nil | x_newHorzValue ] [ nil | x_newVertValue ] )
```

The first element of the list is the new position of the horizontal scroll bar or `nil` if the scroll bar is not visible. The second element of the list is the new position of the vertical scroll bar or `nil` if the scroll bar is not visible.

Related Topics

[Form and Field Functions](#)

[hiGetScrollBarInfo](#)

hiSetTopListItem

```
hiSetTopListItem(  
    r_listBoxField  
    x_itemPosition  
)  
=> t / nil
```

Description

Sets the first visible item in a list box field to the specified position (*x_itemPosition*). A position value of 1 indicates the first item, a position value of 2 indicates the second item, and so on. A value of 0 (zero) specifies the last item in the list.

This function is only used with the list box fields in forms.

Arguments

<i>r_listBoxField</i>	List box field to reset.
<i>x_itemPosition</i>	The position of the item that is made the first visible item.

Values Returned

t	The first visible item is set.
nil	The position is not within the specified range.

Related Topics

[Form and Field Functions](#)

hiSetTreeItemDescription

```
hiSetTreeItemDescription(  
    g_item  
    g_description  
)  
=> t / nil
```

Description

Changes the item description. The description is initially set with the *g_description* argument of `hiCreateTreeItem`. To get the current description, use `hiGetTreeItemDescription`.

Arguments

<i>g_item</i>	The item whose description you want to change. This is the return value of <code>hiCreateTreeItem</code> .
<i>g_description</i>	The new description for the argument, specified as a list of strings or symbols.

Values Returned

<i>t</i>	The new description was set for the item.
<i>nil</i>	The new description could not be set for the item.

Related Topics

[Form and Field Functions](#)

[hiGetTreeItemDescription](#)

[hiCreateTreeItem](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

hiSetTreeItemIcons

```
hiSetTreeItemIcons (  
    r_treeItem  
    l_icons  
)  
=> t / nil
```

Description

Sets the following icons for a tree item: default icon, expanded icon (displayed when the item is expanded), selection icon (displayed when the item is selected), and expanded selection icon (displayed when the item is both expanded and selected).

Arguments

<i>r_treeItem</i>	Handle to the tree item, returned by <code>hiCreateTreeItem</code> .
<i>l_icons</i>	A list of four icons, in the following order: (<i>g_itemIcon</i> <i>g_expandedIcon</i> <i>g_selectionIcon</i> <i>g_expandedSelectionIcon</i>). <i>g_selectionIcon</i> and <i>g_expandedSelectionIcon</i> are currently not displayed.

Value Returned

<i>t</i>	The icons are set. Note that <i>g_selectionIcon</i> and <i>g_expandedSelectionIcon</i> are updated but are currently not displayed in the tree.
<i>nil</i>	<i>r_treeItem</i> or an icons in <i>l_icons</i> is invalid.

Related Topics

[Form and Field Functions](#)

[hiGetTreeItemIcons](#)

[hiSwapTreeItemSelectionIcons](#)

[hiCreateTreeItem](#)

hiSetListItemVisible

```
hiSetListItemVisible(  
    o_listBoxField  
    x_position  
)  
=> t / nil
```

Description

Brings the specified list box item into view in the list box. Takes a list box field of a form as the first argument and the numeric position of the item you want to bring into view as the second argument. If the numeric position specified is 0 or less than 0, the last item in the list is scrolled into view.

Arguments

<i>o_listBoxField</i>	The list box field.
<i>x_position</i>	Numeric position of the list box item you want to bring into view. The first item is represented by 1. If <i>x_position</i> is 0 or less than 0, the last item in the list is scrolled into view.

Values Returned

<i>t</i>	In all cases except when <i>x_position</i> is greater than the number of items in the list box.
<i>nil</i>	<i>x_position</i> is greater than the number of items in the list box.

Related Topics

[Form and Field Functions](#)

hiSetListItemCenter

```
hiSetListItemCenter(  
    o_listBoxField  
    x_position  
)  
=> t / nil
```

Description

Brings the specified list box item into view in the center of the list box, if possible. Takes a list box field of a form as the first argument and the numeric position of the item you want to bring into view as the second argument.

If the numeric position specified is 0 or less than 0, the last item in the list is scrolled into view.

If you specify the first or last item in the list, white space is never created above or below the item to bring it into the center of the list box.

Arguments

<i>o_listBoxField</i>	The list box field.
<i>x_position</i>	Numeric position of the list box item you want to bring into view. The first item is represented by 1. If <i>x_position</i> is 0 or less than 0, the last item in the list is scrolled into view.

Values Returned

<i>t</i>	In all cases except when <i>x_position</i> is greater than the number of items in the list box.
<i>nil</i>	<i>x_position</i> is greater than the number of items in the list box.

Related Topics

[Form and Field Functions](#)

hiShowFieldBorders

```
hiShowFieldBorders(  
    g_onOrOff  
)  
=> t / nil
```

Description

Displays or hides borders around form fields. Used as a debugging aid to avoid overlapping fields when laying out forms.

The default value of the flag is `nil`. The argument `g_onOrOff` can be any non-`nil` value or `nil`. If the argument is non-`nil`, the function sets a global flag that causes borders to be drawn around fields in a form when that form is instantiated. After a form is instantiated, the borders remain drawn or undrawn until the form is deleted and re-created.

The color used for drawing the borders is controlled by the X resource `Opus.borderColor`.

Argument

<code>g_onOrOff</code>	State of the field borders.
	Valid values:
■	Any non- <code>nil</code> value: borders on
■	<code>nil</code> : borders off

Values Returned

<code>t</code>	The value passed in is non- <code>nil</code> .
<code>nil</code>	The value passed in is <code>nil</code> .

Related Topics

[Form and Field Functions](#)

hiStoreFormLocation

```
hiStoreFormLocation(  
    s_formName  
    l_location  
)  
=> t
```

Description

Stores a form location coordinates so that the form can be restored later to the same position.

This function stores the name and associated xy position of a form in an internal table. The function `hiDisplayForm` accesses this information prior to displaying the form. The location is updated when the form is moved so that the form does not go back to its old location when closed and reshown. The information stored overrides any positional information specified in the `hiDisplayForm` function.

This function is primarily for restoring a saved session.

Arguments

<i>s_formName</i>	Name or base name of the form whose location is to be stored. This argument can contain either the form name or the base name passed by the argument in the <code>hiCreateAppForm</code> function.
<i>l_location</i>	Coordinates to be stored for the specified form.

Value Returned

t	The location of the form is stored successfully.

Related Topics

[Form and Field Functions](#)

[hiCreateAppForm](#)

[hiDisplayForm](#)

hiSwapTreeItemSelectionIcons

```
hiSwapTreeItemSelectionIcons(  
    r_treeItem  
)  
=> t / nil
```

Description

Swaps the item icon and expanded icon with the selection icon and expanded selection icon respectively for the specified tree item. For information about setting these icons, see [hiSetTreeItemIcons](#).

Argument

<i>r_treeItem</i>	Handle to the tree item, returned by hiCreateTreeItem .

Value Returned

<i>t</i>	The icons are swapped.
<i>nil</i>	<i>r_treeItem</i> is invalid.

Related Topics

[Form and Field Functions](#)

[hiGetTreeItemIcons](#)

[hiSetTreeItemIcons](#)

[hiCreateTreeItem](#)

hiTreeAppendItem

```
hiTreeAppendItem(  
    g_tree  
    g_item  
)  
=> t / nil
```

Description

Adds an item at the end of a tree. To add an item to the beginning of the tree, use `hiTreePrependItem`.

Arguments

<code>g_tree</code>	The tree to which you want to add the item. This is the return value of <code>hiCreateTree</code> .
<code>g_item</code>	The item you want to add to the tree. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<code>t</code>	The item was added to the tree.
<code>nil</code>	The item was not added to the tree.

Related Topics

[Form and Field Functions](#)

[hiTreeAppendItems](#)

[hiTreePrependItem](#)

[hiCreateTreeItem](#)

[hiCreateTree](#)

[hiItemInsertTree](#)

[hiCreateTreeTable](#)

hiTreeAppendItems

```
hiTreeAppendItems (
    r_tree
    l_items
)
=> t / nil
```

Description

Adds items at the end of a tree. To add items to the beginning of the tree, use `hiTreePrependItems`.

Arguments

<i>r_tree</i>	The tree to which you want to add the item. This is the return value of <code>hiCreateTree</code> .
<i>l_items</i>	The item you want to add to the tree. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<i>t</i>	The items were added to the tree.
<i>nil</i>	The items were not added to the tree.

Example

```
hiTreeAppendItems ( myTree myItemsList )
```

Related Topics

[Form and Field Functions](#)

[hiTreePrependItem](#)

[hiCreateTreeItem](#)

[hiCreateTree](#)

[hiltemInsertTree](#)

Cadence User Interface SKILL Reference

Form Functions

hiCreateTreeTable

hiTreePrependItem

```
hiTreePrependItem(  
    g_tree  
    g_item  
)  
=> t / nil
```

Description

Adds an item to the beginning of a tree. To add an item at the end of the tree, use `hiTreeAppendItem`.

Arguments

<code>g_tree</code>	The tree to which you want to add the item. This is the return value of <code>hiCreateTree</code> .
<code>g_item</code>	The item you want to add to the tree. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<code>t</code>	The item was added to the tree.
<code>nil</code>	The item was not added to the tree.

Related Topics

[Form and Field Functions](#)

[hiTreePrependItems](#)

[hiTreeAppendItem](#)

[hiCreateTreeItem](#)

[hiCreateTree](#)

[hiItemInsertTree](#)

[hiCreateTreeTable](#)

hiTreePrependItems

```
hiTreePrependItems (  
    r_tree  
    l_items  
)  
=> t / nil
```

Description

Adds items to the beginning of a tree. To add items at the end of the tree, use `hiTreeAppendItems`.

Arguments

<i>r_tree</i>	The tree to which you want to add the item. This is the return value of <code>hiCreateTree</code> .
<i>l_items</i>	The item you want to add to the tree. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<i>t</i>	The items were added to the tree.
<i>nil</i>	The items were not added to the tree.

Related Topics

[Form and Field Functions](#)

[hiTreePrependItem](#)

[hiTreeAppendItem](#)

[hiCreateTreeItem](#)

[hiCreateTree](#)

[hiItemInsertTree](#)

[hiCreateTreeTable](#)

Cadence User Interface SKILL Reference

Form Functions

hiTreeRemoveItem

```
hiTreeRemoveItem(  
    g_tree  
    g_item  
)  
=> t / nil
```

Description

Removes an item from a tree. To remove more than one item, use `hiTreeRemoveItems`. To remove all the items, use `hiTreeRemoveAllItems`.

You can add the item again with `hiTreeAppendItem` or `hiTreePrependItem`.

Arguments

<code>g_tree</code>	The tree from which you want to remove the item. This is the return value of <code>hiCreateTree</code> .
<code>g_item</code>	The item you want to remove from the tree. This is the return value of <code>hiCreateTreeItem</code> .

Values Returned

<code>t</code>	The item was removed.
<code>nil</code>	The item was not removed.

Related Topics

[Form and Field Functions](#)

[hiTreeRemoveItems](#)

[hiTreeRemoveAllItems](#)

[hiItemRemoveTree](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetTreeItems

hiCreateTreeTable

hiTreeRemoveItems

```
hiTreeRemoveItems (  
    g_tree  
    g_itemList  
)  
=> t / nil
```

Description

Removes the specified items from a tree. To remove a single item, use `hiTreeRemoveItem`; to remove all items, use `hiTreeRemoveAllItems`.

You can add the item again with `hiTreeAppendItem` or `hiTreePrependItem`.

Arguments

<i>g_tree</i>	The tree from which you want to remove the items. This is the return value of <code>hiCreateTree</code> .
<i>g_itemList</i>	The items you want to remove from the tree. Specify this argument as a list of item handles, each of which is returned from <code>hiCreateTreeItem</code> .

Values Returned

<i>t</i>	The items were removed.
<i>nil</i>	The items were not removed.

Related Topics

[Form and Field Functions](#)

[hiTreeRemoveItem](#)

[hiTreeRemoveAllItems](#)

[hiItemRemoveTree](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

Cadence User Interface SKILL Reference

Form Functions

hiGetTreeItems

hiCreateTreeTable

hiTreeRemoveAllItems

```
hiTreeRemoveAllItems(  
    g_tree  
)  
=> t / nil
```

Description

Removes all the items from a tree. To remove a single item, use `hiTreeRemoveItem`. To remove more than one item, use `hiTreeRemoveItems`.

You can add the item again with `hiTreeAppendItem` or `hiTreePrependItem`.

Argument

<code>g_tree</code>	The tree from which you want to remove all items. This is the return value of <code>hiCreateTree</code> .

Values Returned

<code>t</code>	The items were removed from the tree.
<code>nil</code>	The items were not removed from the tree.

Related Topics

[Form and Field Functions](#)

[hiTreeRemoveItem](#)

[hiTreeRemoveItems](#)

[hiItemRemoveTree](#)

[hiTreeAppendItem](#)

[hiTreePrependItem](#)

[hiGetTreeItems](#)

[hiCreateTreeTable](#)

hiTreeTableDeselectItem

```
hiTreeTableDeselectItem(  
    g_treeField  
    g_item  
    [ g_notify ]  
)  
=> t / nil
```

Description

Deselects an item in a tree table field. To deselect more than one item, use `hiTreeTableDeselectItems`. To deselect all selected items in a field, use `hiTreeTableDeselectAllItems`.

This function is the programmatic equivalent of deselecting items with the mouse or keyboard keys.

Arguments

<i>g_treeField</i>	The tree table field that contains the item.
<i>g_item</i>	The item you want to deselect. It is a symbol of the tree item, which can be obtained from the return value of the <code>hiCreateTreeItem</code> .
<i>g_notify</i>	Opens the callback procedure of the tree table field. Specify <code>nil</code> if you do not want to execute the callback procedure.

Values Returned

<i>t</i>	The item was deselected.
<i>nil</i>	The item was not deselected.

Related Topics

[Form and Field Functions](#)

[hiTreeTableDeselectItems](#)

[hiTreeTableDeselectAllItems](#)

Cadence User Interface SKILL Reference

Form Functions

hiTreeTableSelectItem

hiCreateTreeTable

hiTreeTableDeselectAllItems

```
hiTreeTableDeselectAllItems(  
    g_treeField  
    [ g_notify ]  
)  
=> t / nil
```

Description

Deselects all selected items in a tree table field. To deselect a single item, use `hiTreeTableDeselectItem`; to deselect more than one item, use `hiTreeTableDeselectItems`.

This function is the programmatic equivalent of deselecting items with the mouse or keyboard keys.

Arguments

<i>g_treeField</i>	The tree table field.
<i>g_notify</i>	Opens the callback procedure of the tree table field.

Values Returned

<i>t</i>	The items were deselected.
<i>nil</i>	The items were not deselected.

Related Topics

[Form and Field Functions](#)

[hiTreeTableDeselectItem](#)

[hiTreeTableDeselectItems](#)

[hiTreeTableSelectAllItems](#)

[hiCreateTreeTable](#)

hiTreeTableDeselectItems

```
hiTreeTableDeselectItems(  
    g_treeField  
    l_items  
    [ g_notify ]  
)  
=> t / nil
```

Description

Deselects items in a tree table field. To deselect a single item, use `hiTreeTableDeselectItem`; to deselect all items in a field, use `hiTreeTableDeselectAllItems`.

This function is the programmatic equivalent of deselecting items with the mouse or keyboard keys.

Arguments

<i>g_treeField</i>	The tree table field that contains the items.
<i>l_items</i>	The items you want to select. <i>l_items</i> can either be a list of item names (symbols) or a list of ranges, specified as follows: (<i>s_parentItem</i> <i>x_topRow</i> <i>x_bottomRow</i>) where <i>s_parentItem</i> is the symbol of the parent item, and <i>x_topRow</i> and <i>x_bottomRow</i> are intergers specifying the range of items to select. (The first item is assumed to be 0, the second 1, and so on.)
<i>g_notify</i>	Opens the callback procedure of the tree table field. Specify <code>nil</code> if you do not want to execute the callback procedure.

Values Returned

<i>t</i>	The items were deselected.
<i>nil</i>	The items were not deselected.

Related Topics

Form and Field Functions

Cadence User Interface SKILL Reference

Form Functions

hiTreeTableDeselectItem

hiTreeTableDeselectAllItems

hiTreeTableSelectItems

hiCreateTreeTable

hiTreeTableGetExpandedItems

```
hiTreeTableGetExpandedItems(  
    g_treeField  
)  
=> l_expandedItems / nil
```

Description

Gets a list of all the items that are currently expanded in a tree table field.

Argument

<i>g_treeField</i>	The tree table field.

Value Returned

<i>l_expandedItems</i>	List of items that are currently expanded in the tree table field.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeTableGetExpandedItemCount](#)

[hiTreeTableGetItems](#)

[hiTreeTableGetSelectedItems](#)

[hiCreateTreeTable](#)

hiTreeTableGetExpandedItemCount

```
hiTreeTableGetExpandedItemCount (
    g_treeField
)
=> x_expandedItemCount / nil
```

Description

Returns the number of items that are currently expanded in a tree table field.

Argument

<i>g_treeField</i>	The tree table field.

Values Returned

<i>x_expandedItemCount</i>	The number of items that are currently expanded in the tree table field.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeTableGetExpandedItems](#)

[hiTreeTableGetItemCount](#)

[hiTreeTableGetSelectedItemCount](#)

[hiCreateTreeTable](#)

hiTreeTableGetItems

```
hiTreeTableGetItems(  
    g_treeField  
)  
=> l_listOfItems / nil
```

Description

Gets a list of all the items in a tree table field.

Argument

<i>g_treeField</i>	The tree table field.

Values Returned

<i>l_listOfItems</i>	List of items in the tree table field.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeTableGetItemCount](#)

[hiTreeTableGetSelectedItems](#)

[hiTreeTableGetExpandedItems](#)

[hiTreeTableSelectItem](#)

[hiTreeTableSelectItems](#)

[hiTreeTableSelectAllItems](#)

[hiCreateTreeTable](#)

hiTreeTableGetItemCount

```
hiTreeTableGetItemCount(  
    g_treeField  
)  
=> x_itemCount / nil
```

Description

Returns the number of items in a tree table field.

Argument

<i>g_treeField</i>	The tree table field.

Values Returned

<i>x_itemCount</i>	The number of items in the tree table field.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeTableGetItems](#)

[hiTreeTableGetSelectedItemCount](#)

[hiTreeTableGetExpandedItemCount](#)

[hiCreateTreeTable](#)

hiTreeTableGetSelectedItems

```
hiTreeTableGetSelectedItems(  
    g_treeField  
)  
=> l_selectedItems / nil
```

Description

Gets a list of all the items that are currently selected in a tree table field.

Argument

<i>g_treeField</i>	The tree table field.

Values Returned

<i>l_selectedItems</i>	List of items that are currently selected in the tree table field.
<i>nil</i>	No items are selected or there was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeTableGetSelectedItemCount](#)

[hiTreeTableGetItems](#)

[hiTreeTableGetExpandedItems](#)

[hiCreateTreeTable](#)

hiTreeTableGetSelectedItemCount

```
hiTreeTableGetSelectedItemCount(  
    g_treeField  
)  
=> x_selectedItemCount / nil
```

Description

Returns the number of items that are currently selected in a tree table field.

Argument

<i>g_treeField</i>	The tree table field.

Values Returned

<i>x_selectedItemCount</i>	The number of items that are currently selected in the tree table field.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeTableGetSelectedItems](#)

[hiTreeTableGetItemCount](#)

[hiTreeTableGetExpandedItemCount](#)

[hiCreateTreeTable](#)

hiTreeTableSelectItem

```
hiTreeTableSelectItem(  
    g_treeField  
    g_item  
    [ g_notify ]  
)  
=> t / nil
```

Description

Selects an item in a tree table field. This function also executes the callback of the tree table field if you specify `t` as the value of the `g_notify` argument.

This function is the programmatic equivalent of selecting an item with the mouse or keyboard keys.

To select more than one item, use `hiTreeTableSelectItems`. To select all items in a field, use `hiTreeTableSelectAllItems`.

Arguments

<code>g_treeField</code>	The tree table field that contains the item.
<code>g_item</code>	The item you want to select. It is a symbol of the tree item, which can be obtained from the return value of the <code>hiCreateTreeItem</code> .
<code>g_notify</code>	Runs the callback of the tree table field. Specify <code>nil</code> if you do not want to execute the callback.

Values Returned

<code>t</code>	The item was selected.
<code>nil</code>	The item was not selected.

Related Topics

[Form and Field Functions](#)

[hiTreeTableSelectItems](#)

Cadence User Interface SKILL Reference

Form Functions

hiTreeTableSelectAllItems

hiTreeTableDeselectItem

hiTreeTableGetSelectedItems

hiTreeTableGetSelectedItemCount

hiCreateTreeTable

hiTreeTableSelectAllItems

```
hiTreeTableSelectAllItems(  
    g_treeField  
    [ g_notify ]  
)  
=> t / nil
```

Description

Selects all items in a tree table field. This function also executes the callback of the tree table field if you specify `t` as the value of the `g_notify` argument.

This function is the programmatic equivalent of selecting items with the mouse or keyboard keys.

To select more than one item, use `hiTreeTableSelectItems`. To select all items in a field, use `hiTreeTableSelectAllItems`.

Call `hiTreeTableSelectAllItems` after the form is displayed, that is, after `hiDisplayForm` is called to avoid any errors.

Arguments

<code>g_treeField</code>	The tree table field that contains the items.
<code>g_notify</code>	Runs the callback procedure of the tree table field. Specify <code>nil</code> if you do not want to execute the callback procedure.

Values Returned

<code>t</code>	The items were selected.
<code>nil</code>	The items were not selected.

Related Topics

[Form and Field Functions](#)

[hiTreeTableSelectItem](#)

[hiTreeTableSelectItems](#)

Cadence User Interface SKILL Reference

Form Functions

hiTreeTableDeselectAllItems

hiTreeTableGetSelectedItems

hiTreeTableGetSelectedItemCount

hiCreateTreeTable

hiTreeTableSelectItems

```
hiTreeTableSelectItems(  
    g_treeField  
    l_items  
    [ g_notify ]  
)  
=> t / nil
```

Description

Selects items in a tree table field. This function also executes the callback of the tree table field if you specify *t* as the value of the *g_notify* argument.

This function is the programmatic equivalent of selecting items with the mouse or keyboard keys.

To select more than one item, use `hiTreeTableSelectItems`. To select all items in a field, use `hiTreeTableSelectAllItems`.

Arguments

<i>g_treeField</i>	The tree table field that contains the items.
<i>l_items</i>	The items you want to select. <i>l_items</i> can either be a list of item names (symbols) or a list of ranges, specified as follows: (<i>s_parentItem</i> <i>x_topRow</i> <i>x_bottomRow</i>) where <i>s_parentItem</i> is the symbol of the parent item, and <i>x_topRow</i> and <i>x_bottomRow</i> are intergers specifying the range of items to select. (The first item is assumed to be 0, the second 1, and so on.)
<i>g_notify</i>	Runs the callback procedure of the tree table field. Specify <code>nil</code> if you do not want to execute the callback procedure.

Values Returned

<i>t</i>	The items were selected.
<code>nil</code>	The items were not selected.

Cadence User Interface SKILL Reference

Form Functions

Related Topics

[Form and Field Functions](#)

[hiTreeTableSelectItem](#)

[hiTreeTableSelectAllItems](#)

[hiTreeTableDeselectItems](#)

[hiTreeTableGetSelectedItems](#)

[hiTreeTableGetSelectedItemCount](#)

[hiCreateTreeTable](#)

hiItemRemoveTree

```
hiItemRemoveTree(  
    g_item  
)  
=> t / nil
```

Description

Removes the sub-tree from an item in the tree. The item remains in the tree as a leaf; only the sub-tree under it is removed. The expansion icon next to the item is removed.

You can add the tree again with `hiItemInsertTree`.

Argument

<i>g_item</i>	The item from which to remove the sub-tree. It is a symbol of the tree item, which can be obtained from the return value of the <code>hiCreateTreeItem</code> .

Values Returned

<i>t</i>	The tree was removed.
<i>nil</i>	There was an error.

Related Topics

[Form and Field Functions](#)

[hiTreeRemoveItem](#)

[hiTreeRemoveItems](#)

[hiTreeRemoveAllItems](#)

[hiCreateTreeTable](#)

hiUpdateFormBlock

```
hiUpdateFormBlock(  
    r_form  
)  
=> t / nil
```

Description

Used to reset a form if you have blocked multiple updates with the `hiSetFormBlock` function. The functions `pf hiUpdateFormBlock(r_form)` is equivalent to `hiSetFormBlock(r_form nil)`.

Use the `hiSetFormBlock` and `hiUpdateFormBlock` functions only if you are making extensive changes to a form.

Argument

<i>r_form</i>	Handle to the form you want to reset.

Values Returned

<i>t</i>	In all cases except when <i>r_form</i> is not valid.
<i>nil</i>	<i>r_form</i> is not valid.

This section includes programming samples for creating forms.

Related Topics

[Form and Field Functions](#)

Cadence User Interface SKILL Reference

Form Functions

Window Functions

This topic provides a list of Cadence® SKILL functions associated with windows.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiGetDbuPoint](#)
- [hiGetDrawThruDelta](#)
- [hiCreateDockWindow](#)
- [hiUndockWindow](#)
- [hilsWindowSpecifier](#)
- [swindow](#)
- [windowp](#)
- [hilsWindowSpecifier](#)
- [hiGetActiveTabWindow](#)
- [hiSetDrawThruDelta](#)
- [hiSetSessionFocus](#)
- [hiSetTabWindowTitle](#)
- [hiSetTabWindowIcon](#)
- [hiSetTabWindowTip](#)
- [hiSetWindowAtts](#)
- [hiSetWinStyle](#)
- [hiRegCloseProc](#)
- [hiVectorPan](#)

Cadence User Interface SKILL Reference

Window Functions

- [hiUnregCurWindowTrigger](#)
- [hiSetSplashBackground](#)
- [hiSetSplashDefaultBackground](#)
- [hiSetSplashIcon](#)
- [hiSetSplashFamily](#)
- [hiSetSplashProduct](#)
- [hiSetSplashLicense](#)
- [hiGetMagnifierSize](#)
- [hilsMagnifierAnchored](#)
- [hilsMagnifierOpened](#)
- [hiResizeMagnifier](#)
- [hiScaleMagnifier](#)
- [hiToggleAnchorMagnifier](#)
- [hiToggleMagnifier](#)
- [hiToggleMagnifierResolution](#)
- [hiUpdateMagOptions](#)
- [hiUsingMagnifierResolution](#)
- [hiZoomInMagnifier](#)
- [hiZoomOutMagnifier](#)

Cadence User Interface SKILL Reference

Window Functions

dwindow

```
dwindow(  
    g_window  
)  
=> w_dockWindowID / nil
```

Description

Gets the window ID of a dockable window.

Argument

<i>g_window</i>	Sets the window number or the window handle with the ?handle argument when the window is created.

Values Returned

<i>w_dockWindowID</i>	The dock window ID.
nil	There is no dock window that matches the window number or handle specified.

Examples

```
dwindow( 'myWin )
```

Related Topics

[Window Functions](#)

[hlsWindowSpecifier](#)

[swindow](#)

Cadence User Interface SKILL Reference

Window Functions

dwindowp

```
dwindowp(  
    g_object  
)  
=> t / nil
```

Checks whether the specified object is a valid dock window.

Argument

g_object The object that you want to check.

Value Returned

t	<i>g_object</i> is a valid dock window.
nil	<i>g_object</i> is not a valid dock window.

Examples

```
dwindowp( myWindowID )
```

Related Topics

[Window Functions](#)

[dwindowp](#)

[Viewing Functions](#)

[dwindowp](#)

getCurrentWindow

The `getCurrentWindow` function is an alias for the `hiGetCurrentWindow` function.

getMaxScreenCoords

The `getMaxScreenCoords` function is an alias for the `hiGetMaxScreenCoords` function.

hiAddFixedMenu

```
hiAddFixedMenu(  
    [ ?window w_window ]  
    ?fixedMenu g_fixedMenu  
    [ ?menuSide s_menuSide ]  
    [ ?append g_append ]  
    [ ?newLine g_newLine ]  
)  
=> t / nil
```

Description

Note: Fixed menus are no longer supported in windows. Use the new toolbar functions to create toolbars instead for windows. Existing fixed menus that have been added to windows are no longer displayed, although no warnings are being generated. Replace these menus with toolbars.

Adds or replaces a vertical fixed menu on a window.

You can have multiple fixed menus in a window. To add the new menu to any existing fixed menus, set the `?append` argument to `t`; to replace all existing fixed menus with the new menu, set `?append` to `nil`.

The menu can be placed on the left, right, or top edge of the window. When you create the menu, you can specify the location with the `?menuSide` argument.

To place a fixed menu on the root window (screen), instead of attaching it to a graphics window, use the `hiDisplayFixedMenu` function.

Arguments

`?window w_window` Window on which to add the fixed menu. This window must be of type `graphics`, `browser`, `form`, `text`, or `encap`. It cannot be a session window or a dockable window. If you do not specify `w_window`, the current window is used.

`?fixedMenu g_fixedMenu`
Vertical fixed menu created with `hiCreateVerticalFixedMenu`, specified as a symbol. This menu can be placed on the left, right, or top edge of the window.

`?menuSide s_menuSide`

Cadence User Interface SKILL Reference

Window Functions

Side of the window where the fixed menu is to be placed. This argument must be either 'left, 'right, 'top, or 'default. If you do not specify *s_menuSide*, the default is used. The default is determined by the settings in the CIW or the .cdsenv file.

`?append g_append` *t* or *nil*. *t* appends the menu to any existing fixed menus on the window; *nil* replaces all existing fixed menus with the new menu.

The default is *nil*.

`?newLine g_newLine`

t or *nil*. *t* places the menus in a new row or column.

Values Returned

<i>t</i>	The fixed menu is added.
<i>nil</i>	The fixed menu is not added.

Examples

```
menu = hiCreateVerticalFixedMenu(...)
hiAddFixedMenu(?window window(5) ?fixedMenu menu
?menuSide 'right)
```

Related Topics

[Window Functions](#)

[hiDisplayFixedMenu](#)

[hiCreateVerticalFixedMenu](#)

Cadence User Interface SKILL Reference

Window Functions

hiCloseWindow

```
hiCloseWindow(  
    w_windowId  
)  
=> t / nil
```

Description

Closes and destroys a window, dock window, or session window.

If the window is of widgetType `form`, the form contained in the window is not destroyed when the window is closed. If the window is a dock window that is currently docked, it is first closed and then destroyed. If the window is a session window, all the windows contained in its main canvas are closed and destroyed. If the window is the only window contained in a session window, the session window is also closed and destroyed.

If the window closed is the current window, an attempt is made to make an equivalent window current (such as another window displaying the same cellview). If an equivalent window is not found, the last-created window is made current. If this window has the property `neverCurrentWindow` set to `t`, another window is made current. If no window can be made current (the CIW can never be the current window), the current window is `nil`.

Note: This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<code>w_windowId</code>	Window, dock window, or session window to close.

Value Returned

<code>t</code>	The window is closed.
<code>nil</code>	The window is not closed. An error message is also returned.

Related Topics

[Window Functions](#)

[hiRegCloseProc](#)

Cadence User Interface SKILL Reference

Window Functions

hiUnregCloseProc

hiCreateDockWindow

```
hiCreateDockWindow(  
    ?appType t_appType  
    ?widgetType t_widgetType  
    [ ?handle s_handle ]  
    [ ?dockSizeHint l_dockSizeHint ]  
    [ ?minimumSizeHint l_minimumSizeHint ]  
    [ ?floatSizeHint l_floatSizeHint ]  
    [ ?form g_form ]  
    [ ?help t_help ]  
    [ ?scrollBar g_scrollBar ]  
    [ ?title t_title ]  
    [ ?visibilityProc s_visibilityProc ]  
)  
=> w_dwindowID / nil
```

Description

Creates a dockable window, which can be docked in a session window with the `hiDockWindow` function. Dockable windows can be of the following types: form, hypertext, html, or text. Dockable windows cannot be of type graphics.

A dockable window can be associated with only one session window.

You can create dockable form windows by specifying the form handle as the `?form` argument and the string "form" as the `?widgetType` argument.

You can add menus to dockable windows with the functions listed in [Menu Functions](#)

This function returns a `dwindowID`. The SKILL output format is:

```
dwindow:#
```

If you specify the `?handle` argument, the dockable window is logged, in log files, as:

```
dwindow('handle)
```

A dockable window can never become the current window.

Cadence User Interface SKILL Reference

Window Functions

Arguments

?appType *t_appType*

String specifying the dockable window application type or name. This string is also used as the window's bindkey prefix.

?widgetType *t_widgetType*

String specifying the type of window widget to be created. One of the following: `text`, `html`, `hypertext`, or `form`.

For information about how to create hypertext, see [Hypertext Viewfile Window](#).

?handle *s_handle* Symbol specifying a handle for the dockable window. You can reference it with `dwindowID->hiHandle`

?dockSizeHint *l_dockSizeHint*

List specifying the initial size of the window when it is docked:

`list(x_width x_height)`

The default dock size is `list(150 150)`.

This value is used the first time that the window is docked. Subsequently, the size that was last set is used.

Here the size refers to the size of the dock window only and does not include the window frame.

?minimumSizeHint *l_minimumSizeHint*

List specifying the minimum size of the window:

`list(x_width x_height)`

The default minimum size is `list(0 0)`.

This specifies the minimum required size of the dock window under normal circumstances.

?floatSizeHint *l_floatSizeHint*

Cadence User Interface SKILL Reference

Window Functions

List specifying the initial size of the window when it is floating:

```
list(x_width x_height)
```

If this argument is not specified, the Qt default is used.

This value is used the first time that the window is floated. Subsequently, the last size and location are used.

Here the size refers to the size of the dock window only and does not include the window frame.

?form *g_form*

SKILL representation of a form returned by `hiCreateAppForm`. Use this argument if you want to display a form in the dockable window. This argument is valid only for dock windows whose `t_widgetType` is "form".

When the dock window containing the form is closed, the form is not destroyed; it can be redisplayed in another window.

Cadence User Interface SKILL Reference

Window Functions

?help *t_help*

Help string for the dock window, which is used to reference a *.tgf* help file. If you do not specify this argument or specify "" (an empty string), the help argument is set to the *appName* (which is the value of the ?*appType* argument).

The help string for a window can also be set at any time using the *hiSetWindowsAtts* functions and can be retrieved with the *hiGetHelp* function.

Cadence applications may modify their window help strings at any time.

When a user clicks *Help* in a window, the help string is used for the help symbol that is looked for in the *.tgf* file. The *appName* is used to determine the *.tgf* file to search.

The help string can also be used to modify the *appName* by assigning an *hiHelpAppName* property to the symbol representation of the string.

For example, `putprop( concat( help_string app_name_string 'hiHelpAppName)`

set the *appID* used for Help to *app_name_string* whenever *help_string* is used as the help string for any window.

For more information about the online help mechanism, see [Online Help Mechanism](#).

For windows help to work correctly, either the *t_help* or *t_appType* argument or both must be set when you create the dock window.

?scrollBar *g_scrollbars*

t or *nil*. If set to *t*, scroll bars appear on the right and bottom of the dock window. If not specified, the value of the property *useScrollbars* is used. The property can be set from the CIW's User Preferences form.

?title *t_title*

String specifying the window title that is displayed in the window frame and in the context popup menu.

?visibilityProc *s_visibilityProc*

Cadence User Interface SKILL Reference

Window Functions

This argument can be used to monitor the visibility of a docked window. Set the argument, at creation, with a function name in either symbol or string format. The function then be called whenever there is a visibility change in the docked window and pass two arguments, the `dockwindow id` and `t/nil` to represent visible or invisible.

The docked window should not be closed from within the visibility procedure (that is, do not call `hiCloseWindow`).

Values Returned

<code>w_dwindowID</code>	The window ID of the dock window.
<code>nil</code>	There was an error.

Examples

```
hiCreateDockWindow( ?appType "text" ?widgetType "text" ?handle 'dw ?title "My Window" )
```

Additional Information

Dockable windows are windows that occupy dock areas within a session window. They typically provide helper functionality for the application. These windows can be docked in a session window or left free. A dockable window can be associated with only one session window.

They are created with `hiCreateDockWindow` and can be of the following types: `browser`, `text`, `html`, `hypertext`, or `form`. You can create dockable form windows by calling `hiCreateDockWindow` and specifying a `?widgetType` of `form`.

Related Topics

[Window Functions](#)

[hiCreateAppForm](#)

[hiUndockWindow](#)

[windowp](#)

hiCreateWindow

```
hiCreateWindow(  
    g_boxSpec  
    t_widgetType  
    t_appType  
    [ t_help ]  
    [ g_scrollbars ]  
    [ g_form ]  
    [ l_iconPosition ]  
)  
=> w_windowId / nil
```

Description

Creates, but does not display, a window of one of the following types: session, graphics, text, hypertext, html, encap, or form. This window becomes the current window.

If you do not want this window to ever become the current window, you can set the window property `neverCurrentWindow` to `t`:

```
( window->neverCurrentWindow = t )
```

When this property is set on a current window, the current window is reset to the last current window or to another open window. If another window cannot be found, the current window is reset to `nil`.

Note: Setting the `neverCurrentWindow` property might affect previously saved replay files that contain this window, and these log files might not replay correctly.

If you set the `neverCurrentWindow` property to `t` and you use `.tgf` help files, you need to specify the correct application ID for help to work correctly. See the note in the description of the `t_help` argument for details.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Cadence User Interface SKILL Reference

Window Functions

Arguments

g_boxSpec

Lower left and upper right corners of the window in screen coordinates (assuming 0:0 as the lower left corner of the screen):

```
list(x1:y1 x2:y2 [iconCoordinate|nil])
```

where *x1* is the lower left x coordinate, *y1* is the lower left y coordinate, *x2* is the upper right x coordinate and *y2* is the upper right y coordinate. See the *l_iconPosition* argument for information about the optional, third element of the list.

If *g_boxSpec* is nil, either you are prompted with a rubberbanding box to enter a window size or a default size is used, depending upon the value of the global variable *hiWinStyle*.

You can use the *hiIsWindowSpecifier* function to check if your *l_boxSpec* value is valid.

t_widgetType

One of the following strings: *session*, *graphics*, *text*, *hypertext*, *html*, *encap*, *form*, or *none* (plot windows) that represent the type of window widget to be created.

For information about how to create hypertext, see [Hypertext Viewfile Window](#).

t_appType

String representing the window application type or name. This string is also used as the window's bindkey prefix.

Cadence User Interface SKILL Reference

Window Functions

t_help

Help string for the window, which is used to reference a .tgf help file. If you do not specify this argument or specify " " (an empty string), the help argument is set to the *appName* (which is the value of the *t_appType* argument).

The help string for a window can also be set at any time using the *hiSetWindowsAtts* functions and can be retrieved with the *hiGetHelp* function.

Cadence applications may modify their window help strings at any time.

When a user clicks *Help* in a window, the help string is used for the help symbol that is looked for in the .tgf file. The *appName* is used to determine the .tgf file to search.

The help string can also be used to modify the *appName* by assigning an *hiHelpAppName* property to the symbol representation of the string.

For example, `putprop( concat( help_string app_name_string 'hiHelpAppName)`

set the *appID* used for Help to *app_name_string* whenever *help_string* is used as the help string for any window.

For more information about the online help mechanism, see [Online Help Mechanism](#)

For windows that cannot become the current window (because the *neverCurrentWindow* property is set to *t*), to specify the correct application ID for accessing help, specify a help name string for the *t_help* argument of *hiCreateWindow()* or *hiOpenWindow()* and then attach the *hiHelpAppName* property to the symbol of the same name as the help string. For example:

```
hiOpenWindow( ?appType "MyApp" ?help
"MyAppHelp" )
putprop( 'MyAppHelp "MyApp" 'hiHelpAppName)
```

For windows help to work correctly, either the *t_help* or *t_appType* argument or both must be set when you create the window.

Cadence User Interface SKILL Reference

Window Functions

<i>g_scrollbars</i>	Either <code>t</code> or <code>nil</code> to specify the use of scroll bars. This field is valid only for windows with <i>widgetType</i> equal to <code>graphics</code> . If set to <code>t</code> , scroll bars appear on the right and bottom of the window. If not specified, the property <code>useScrollbars</code> from the CIW is used. This value can be set from the User Preferences form.
<i>g_form</i>	SKILL representation of a form returned by <code>hiCreateAppForm</code> . This argument is only valid for windows with a <i>widgetType</i> of <code>form</code> . When the window containing the form is closed, the form is not destroyed. The form can be redisplayed in another window.
<i>l_iconPosition</i>	Icon location (in X Window System screen coordinates) of the window. This causes the window to be iconified initially. If <i>l_iconPosition</i> is <code>nil</code> or not specified, the window is not iconified initially. The <i>g_boxSpec</i> argument can also contain the icon position. If both methods are used, this icon position overrides the position passed in <i>g_boxSpec</i> unless <i>l_iconPosition</i> is <code>nil</code> .

Value Returned

<i>w_windowId</i>	The window ID of the created window, in the following format: window:# (all windows except session windows) swindow:# (session windows) where # is the window number.
<code>nil</code>	The window is not created. An error message is also issued.

Examples

```
hiCreateWindow( 'default "graphics" "MyApp" )  
=> window:4
```

Related Topics

[Window Functions](#)

[hiCreateAppForm](#)

Cadence User Interface SKILL Reference

Window Functions

hlsWindowSpecifier

hiDeiconifyWindow

```
hiDeiconifyWindow(  
    w_windowId  
)  
=> t / nil
```

Description

Converts an icon to a window. If the window is contained in a session window, the session window is deiconified.

w_windowId can also be a session window but it cannot be a dock window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

Argument

<i>w_windowId</i>	Window you want to deiconify. <i>w_windowId</i> can be a window or a session window. It cannot be a dock window.

Values Returned

t	The window is deiconified.
nil	The window is not deiconified. An error message is also issued.

Related Topics

[Window Functions](#)

[hilconifyWindow](#)

[hiLowerWindow](#)

hiDisplayWindow

```
hiDisplayWindow(  
    w_windowId | w_sessionWindowID  
    [ w_sessionWindowId ]  
    [ g_tabTitle ]  
    [ g_tabIcon ]  
    [ g_tabTip ]  
)  
=> t / nil
```

Description

Displays any window created by `hiCreateWindow`, including session windows; does not display dockable windows or empty session windows. This function always displays graphics windows in a session window—if you do not specify a session window, this function creates a new one and displays the graphics window in it.

Use the second argument to specify the session window in which to place the window. If the session window already contains windows, a new tab is created for the new window, and the new tab and new window are raised to the top. If the session window is not yet displayed, it is displayed.

If a session window is the first argument, no other arguments are accepted. You cannot display an empty session window; it must contain a window.

This function does not display dockable windows. To display dockable windows, use `hiDockWindow`.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

Arguments

w_windowId | *w_sessionWindowID*

The window or session window that you want to display. You cannot display an empty session window—it must contain a window.

Cadence User Interface SKILL Reference

Window Functions

<i>w_sessionWindow</i>	The session window in which you want to place <i>w_windowID</i>. <i>w_windowID</i> must not be associated with another session window. This argument is used only if the first argument is not a session window.
<i>g_tabTitle</i>	The tab title for the new window. If the session window already contains windows, a tab is created for the new window. After the window is created, you can use the <code>hiSetTabWindowTitle</code> function to change the tab title.
<i>g_tabIcon</i>	The icon to be displayed on the tab for the new window. After the window is created, you can use the <code>hiSetTabWindowIcon</code> function to change the tab icon.
<i>g_tabTip</i>	String specifying the tooltip that gets displayed whenever the cursor hovers over the tab. After the window is created, you can use the <code>hiSetTabWindowTip</code> function to change the tab tooltip.

Values Returned

<i>t</i>	The window is displayed.
<i>nil</i>	There was an error. An error message is also issued.

Examples

Associates a window with a session window and displays it:

```
hiDisplayWindow( myWin mySessionWin )
```

Displays a graphics window in a new session window:

```
hiDisplayWindow( myGraphicsWin )
```

Displays a non-empty session window:

```
hiDisplayWindow( mySessionWindow )
```

Related Topics

Window Functions

hiDockWindow

```
hiDockWindow(  
    ?window w_dwindowID  
    [ ?session g_sessionWindowID ]  
    [ ?side s_side ]  
    [ ?index x_index ]  
    [ ?unmapped g_unmapped ]  
    [ ?tabOnto w_dwindowID ]  
)  
=> t / nil
```

Description

Docks a dockable window in a session window. You must specify the *g_sessionWindowID* argument the first time that you display a dock window with this function.

The initial size of the docked window depends on the dock size hint specified when the window was created. Subsequently, the size that was last set interactively is used.

Arguments

?window w_dwindowID

The ID of the dockable window, returned by *hiCreateDockWindow*.

?session g_sessionWindowID

The ID of the session window in which you want to dock the window, returned by *hiCreateWindow* or obtained by *swindow*.

You must specify this argument the first time that you display the dockable window so that it is associated with the session window. This argument is optional in subsequent calls to the function. A dockable window can only be associated with one session window.

?side s_side

One of the following symbols: 'left, 'right, 'top, or 'bottom, specifying the dock area in which to dock the window.

Default value: 'left

Cadence User Interface SKILL Reference

Window Functions

`?index x_index` The position before which to insert the dockable window. This is based on a 0-based index of all the dockable windows in a dock area.

The default value is -1, which places the window at the end of the dock area.

`?unmapped g_unmapped`

Unmaps the dockable window to be docked.

Default value: `nil`

`?tabOnto w_dwindowID`

The ID of the docked window (as returned by `hiCreateDockWindow`) that the window is to be tabbed onto, or `nil`.

If set to `nil` (the default), current behavior remains unchanged. However, if a window ID is specified, then the current setting for `?side` gets ignored,. Additionally, if `?session` has been specified, then it must be the session window that `?tabOnto` is to be docked to.

It is therefore not recommended that `?session` be specified when `?tabOnto` is being used.

Values Returned

`t` The dockable window was docked in the session window it is associated with.

`nil` There was an error.

Examples

```
hiDockWindow( ?window myDockAsst ?session mainSession ?side 'right )
```

Related Topics

[Window Functions](#)

[hiUndockWindow](#)

[hiCreateDockWindow](#)

Cadence User Interface SKILL Reference

Window Functions

hlsWindowSpecifier

hiFocusToCursor

```
hiFocusToCursor(  
    g_cursorFocus  
)  
=> t
```

Description

Sets the window focus mechanism.

If *g_cursorFocus* is set to `t`, the current window is set by moving the cursor into a window. If set to `nil`, you must click in a window to make it current.

The current window is always the last window that contained the cursor. The Command Interpreter Window (CIW) can never be the current window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<i>g_cursorFocus</i>	<code>t</code> or <code>nil</code> . If set to <code>t</code> , the current window is set by moving the cursor into a window. If set to <code>nil</code> , you must click in a window to make it current.

Values Returned

<code>t</code>	Always returned.

Related Topics

[Window Functions](#)

hiGetAbsWindowScreenBBox

```
hiGetAbsWindowScreenBBox(  
    w_windowId  
    [ g_includeWMOffsets ]  
)  
=> l_bBox
```

Description

Returns the absolute screen coordinates of a window, assuming 0:0 is the lower-left corner of the screen.

Arguments

w_windowId Window whose screen coordinates you want to get.
w_windowId can be a window, session window, or dockable window.

g_includeWMOffsets Offsets the returned coordinates by the window manager frame size. Setting this argument to `t` makes this function useful with `hiResizeWindow()`.

Value Returned

l_bBox A list containing the lower-left and upper-right screen coordinates of *w_windowId* in the following format:

((lowerLeft_Xcoord lowerLeft_Ycoord)
 (upperRight_Xcoord upperRight_Ycoord)

If *w_windowID* is a window contained in a session window, the screen coordinates of the session window are returned. If *w_windowID* is a dockable window that is docked in a session window, the screen coordinates of the session window are returned. If *w_windowID* is a dockable window that is floating, the dockable window's screen coordinates are returned.

Examples

Returns the screen coordinates of `window (2)`.

Cadence User Interface SKILL Reference

Window Functions

```
hiGetAbsWindowScreenBBox(window(2))
=> ((221 296) (1121 964))
```

The following sample creates a form that allows the user to change the title or size of a given window.

```
;;; retrieving the height of the bounding box
procedure( trGetBBoxHeight( bBox )
  let( ( ll ur lly ury )
    ll = lowerLeft( bBox )
    lly = yCoord( ll )
    ur = upperRight( bBox )
    ury = yCoord( ur )
    ury - lly
  ) ; let
) ; procedure

;;; retrieving the width of the bounding box
procedure( trGetBBoxWidth( bBox )
  let( ( ll ur llx urx )
    ll = lowerLeft( bBox )
    llx = xCoord( ll )
    ur = upperRight( bBox )
    urx = xCoord( ur )
    urx - llx
  ) ; let
) ; procedure

;;; retrieving the window height
procedure( trGetWindowHeight( wid )
  trGetBBoxHeight( hiGetAbsWindowScreenBBox( wid ) )
) ; procedure

;;; retrieving the window width
procedure( trGetWindowWidth( wid )
  trGetBBoxWidth( hiGetAbsWindowScreenBBox( wid ) )
) ; procedure

;;; resizing the window to the given bounding box
procedure( trResizeWindow( @key ( id hiGetCurrentWindow() ) ( height nil ) ( width
nil ) )
  let( ( bBox ll ur llx lly urx ury urNew )
    bBox = hiGetAbsWindowScreenBBox( id )
    ll = lowerLeft( bBox )
    ur = upperRight( bBox )
    llx = xCoord( ll )
    lly = yCoord( ll )
    urx = xCoord( ur )
    ury = yCoord( ur )

    unless( height height = ury - lly )
    unless( width width = urx - llx )

    urNew = llx + width : lly + height

    hiResizeWindow( id list( ll urNew ) )

    id
  ) ; let
) ; procedure
```

Cadence User Interface SKILL Reference

Window Functions

```
;;; creating a form that acts upon the window with the
;;; window id (wid)
procedure( trCreateWindowForm( wid )
  let( ( nameField heightField widthField theFormSymbol theForm )

    nameField =
      hiCreateStringField(
        ?prompt "Name"
        ?name 'nameField
        ?value hiGetWindowName( wid )
        ?defValue hiGetWindowName( wid )
        ?callback "println( 'nameField )"
      )

    heightField =
      hiCreateScaleField(
        ?prompt "Height"
        ?name 'heightField
        ?value trGetWindowHeight( wid )
        ?defValue trGetWindowHeight( wid )
        ?range list( 0 yCoord( hiGetMaxScreenCoords() ) )
        ?callback "println( 'heightField )"
      )

    widthField =
      hiCreateScaleField(
        ?prompt "Width"
        ?name 'widthField
        ?value trGetWindowWidth( wid )
        ?defValue trGetWindowWidth( wid )
        ?range list( 0 xCoord( hiGetMaxScreenCoords() ) )
        ?callback "println( 'widthField )"
      )

    ;;; specifying a unique symbol to allow multiple
    ;;; instances of the form to be active at a single
    ;;; time
    theFormSymbol = gensym( 'WindowForm )

    ;;; building the form's data structure
    theForm =
      hiCreateAppForm(
        ?name theFormSymbol
        ?formTitle sprintf( nil "%L" wid )
        ?callback "trWindowFormCB( hiGetCurrentForm() )"
        ?fields list( nameField heightField widthField )
        ?unmapAfterCB t
      )

    ;;; associating the wid with the form via a user
    ;;; defined slot
    theForm->wid = wid

    theForm ;;; the return value

  ) ; let
) ; procedure
```


Cadence User Interface SKILL Reference

Window Functions

```
;;; The following callback for the form handles the
;;; renaming and resizing of the window.
procedure( trWindowFormCB( theForm )
  let( ( wid newName newHeight newWidth )
    ;;; picking up the current form values
    wid = theForm->wid
    ;;; user specified slot linking the form to the window
    newName = theForm->nameField->value
    ;;; accessing the current heightField
    newHeight = theForm->heightField->value
    ;;; accessing the current widthField
    newWidth = theForm->widthField->value

    ;;; setting the window name
    hiSetWindowName( wid newName )

    ;;; resizing the window
    trResizeWindow(
      ?id wid
      ?height newHeight
      ?width newWidth
    ) ; trResizeWindow

    t      ;;; the return value
  ) ; let
) ; procedure
```

To test your application using the CIW, type:

```
wf1 = trCreateWindowForm( window(1))
hiDisplayForm(wf1)
```

After the form is displayed, change the values of the Height and Width fields and type a new title for the CIW in the Name field. Select the Apply button to see the changes in the title and size of the CIW.

Related Topics

Window Functions

hiGetActiveTabWindow

```
hiGetActiveTabWindow(  
    w_sessionWindowID  
)  
=> w_windowID / nil
```

Description

Returns the window ID of the window that is the active tab in the session window.

Argument

w_sessionWindowID The session window ID.

Value Returned

<i>w_windowID</i>	The window ID of the active tab.
<i>nil</i>	The session window you specified is not valid.

Example

```
hiGetActiveTabWindow( swindow ( 6 ) )
```

Related Topics

[Window Functions](#)

[hiSetDrawThruDelta](#)

[hiSetTabWindowTitle](#)

[hiSetTabWindowIcon](#)

[hiSetTabWindowTip](#)

[hiOpenWindow](#)

hiGetAppType

```
hiGetAppType (  
    w_windowId  
)  
=> t_applicationType / nil
```

Description

Returns the application type of a window. The application type is set when the window is created.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<i>w_windowId</i>	Window ID of the window, session window, or dockable window.

Value Returned

<i>t_applicationType</i>	The application type of the window. The application type is set when the window is created with <code>hiCreateWindow</code> , <code>hiOpenWindow</code> , or <code>deOpen</code> , for example, Schematics or "Command Interpreter. The application type of a session window is always "session".
<i>nil</i>	<i>w_windowId</i> is invalid.

Related Topics

[Window Functions](#)

[hiOpenWindow](#)

hiGetCurrentEnterFunPoints

```
hiGetCurrentEnterFunPoints()  
    => l_points
```

Description

Returns the list of points entered in the active enterfunction of the current window.

Arguments

None.

Value Returned

<i>l_points</i>	List of points entered in the active enterfunction of the current window.

Examples

In this example, the last point in the list of points is used along with the current pointer location to determine the general direction of the current segment being entered for a line or path.

```
(procedure GetCurrentSegmentDirection()  
  (let ((currPt (hiGetPoint (hiGetCurrentWindow)))  
        (errMsg "Unable to determine direction")  
        (fnName "GetCurrentSegmentDirection") prevPt  
        (efunPts (hiGetCurrentEnterFunPoints)))  
    prevPt = (caar (last efunPts))  
    (cond  
      (!efunPts (warn "%s: No enterfun points.  %s" fnName errMsg))  
      (!currPt (warn "%s: No current point.  %s" fnName errMsg))  
      ((2 * (abs (car prevPt) - (car currPt)))  
        < (abs (cadr prevPt) - (cadr currPt))) "vertical")  
      ((abs (car prevPt) - (car currPt))  
        > (2 * (abs (cadr prevPt) - (cadr currPt)))) "horizontal")  
      (t "diagonal")  
    )  
  )  
)
```

Cadence User Interface SKILL Reference

Window Functions

Related Topics

Window Functions

Cadence User Interface SKILL Reference

Window Functions

hiGetCurrentWindow

```
hiGetCurrentWindow()  
=> w_windowId / nil
```

Description

Returns the ID of the current window. If the current window is in a session window, the ID of the window, not the session window, is returned. Session windows and dockable windows can never be the current window.

Arguments

None.

Value Returned

<i>w_windowId</i>	The window ID of the current window.
<i>nil</i>	There is no current window.

Related Topics

[Window Functions](#)

[hiSetCurrentWindow](#)

Cadence User Interface SKILL Reference

Window Functions

hiGetCursor

```
hiGetCursor(  
    w_windowId  
)  
=> x_cursor
```

Description

Identifies the type of cursor that is currently set for the specified window.

Argument

<i>w_windowId</i>	The window ID.

Value Returned

<i>x_cursor</i>	The integer ID associated with the cursor type that is currently set for the window.

Related Topics

[Window Functions](#)

[hiSetCursor](#)

hiGetCursorByName

```
hiGetCursorByName(  
    t_cursorName  
)  
=> x_cursorID
```

Description

Returns the ID associated with the specified cursor name.

Arguments

<i>t_cursorName</i>	Name of a cursor. For example, the cursor name that is returned by the <code>hiGetCursorName</code> SKILL function.

Value Returned

<i>x_cursorID</i>	An integer that is the ID of the specified cursor name. If the cursor name is "unknown", 1 is returned, which is the value of <code>hicArrow</code> . If the name is invalid, 0 is returned, which is the value of <code>hicNoCursor</code> .

Examples

Using `hiGetCursorName()` to get the name of specified cursor ID.

```
hiGetCursorByName("arrow")  
=> 1  
; 1 is the value of hicArrow
```

```
hiGetCursorByName("ibeam")  
=> 5  
; 5 is the value of hicIbeam
```

```
hiGetCursorByName("unknown")  
=> 1
```


Cadence User Interface SKILL Reference

Window Functions

```
; 1 is the value of hicArrow  
  
hiGetCursorByName("invalidName")  
=> 0  
  
; 0 is the value of hicNoCursor
```

Related Topics

[Window Functions](#)

[hiGetCursorName](#)

hiGetDbuPoint

```
hiGetDbuPoint(  
    w_windowId  
)  
=> l_point / nil
```

Description

Returns the current location of the pointer, or `nil` if the pointer is not within the drawable area of the window or the window is hidden behind other tabs in a session window.

Even if the pointer is currently over another window that obscures `w_windowId`, this function returns its location as long as it is within the boundaries of `w_windowId`.

Before returning the current location of the pointer, `hiGetDbuPoint` calls the appropriate transform and adjust procedures.

Argument

<code>w_windowId</code>	The window ID. <code>w_window</code> can be any window outside a session window or a graphics window within a session window. <code>w_window</code> cannot be a dockable window.

Value Returned

<code>l_point</code>	Location of the pointer in database units.
<code>nil</code>	The pointer is not within the drawable area of the window or <code>w_window</code> is hidden behind other tabs in a session window.

Related Topics

[Window Functions](#)

[hiOpenWindow](#)

[hiGetDrawThruDelta](#)

hiGetDrawThruDelta

```
hiGetDrawThruDelta(  
    [ g_detail ]  
)  
=> x_defaultDelta / ( x_defaultDelta ( x_highButton1Delta x_highButton2Delta  
    x_highButton3Delta ) )
```

Description

When no argument is presented, or if the *g_detail* argument is nil, returns the default draw-thru delta.

The draw-thru delta being the number of pixels that a mouse cursor must travel, with the mouse-button pressed, in order to be interpreted as a draw-thru command; if the mouse is released after double-click time expires. If the *g_detail* argument is non-nil, then the return value is a list, where the first member is the default draw-thru delta, and the second member is the list of high draw-thru deltas for mouse buttons 1, 2, and 3, respectively.

A high draw-thru delta is the distance that the mouse cursor must move, in pixels, to be interpreted as a draw-thru command, if the times between the mouse-button press and the mouse-button release, is less than the double-click time.

Cadence User Interface SKILL Reference

Window Functions

Argument

<i>g_detail</i>	Optional argument which creates a list to be returned if results are non-nil. This list consists of the old delta value first, followed by the pre-timeout delta value second.

Values Returned

<i>x_defaultDelta</i>	The default draw-thru delta value.
<i>x_highButton1Delta</i>	The draw-thru delta value for mouse button 1.
<i>x_highButton2Delta</i>	The draw-thru delta value for mouse button 2.
<i>x_highButton3Delta</i>	The draw-thru delta value for mouse button 3.

Related Topics

Related Topics

[Window Functions](#)

[hiOpenWindow](#)

[hiGetDbuPoint](#)

Cadence User Interface SKILL Reference

Window Functions

hiGetEFDrawThruDelta

```
hiGetEFDrawThruDelta()  
=> DrawThruDeltas / nil
```

Description

Returns the list of EnterFunction draw-thru deltas for the three mouse buttons.

The default return value is (500 10 10, if the default values have not been changed using hiSetEFDrawThruDelta.

Arguments

None.

Value Returned

DrawThruDeltas	List of EnterFunction draw-thru deltas for mouse buttons (x3).
nil	Command failed.

Example

```
hiGetEFDrawThruDelta() =>  
(x_button1EFdelta x_button2EFdelta x_button3EFdelta)
```

Related Topics

[Window Functions](#)

[hiSetEFDrawThruDelta](#)

hiGetEFunCursor

```
hiGetEFunCursor(  
    w_window  
)  
=> t / nil
```

Description

Gets the cursor for an active enterfunction.

Argument

<i>w_window</i>	The window ID of the window for which you want to get the active enterfunction cursor.

Value Returned

<i>t</i>	The currently set cursor.
<i>nil</i>	There was no active enterfunction for the specified window.

Related Topics

[Window Functions](#)

[hiSetCursor](#)

Cadence User Interface SKILL Reference

Window Functions

hiGetHelp

```
hiGetHelp(  
    w_windowId  
)  
=> t_help
```

Description

Retrieves the help string that is defined for the window.

Argument

<i>w_windowId</i>	The ID of the window, session window, or dock window whose help string you want to retrieve.

Values Returned

<i>t_help</i>	The help string that is defined for the window.

Related Topics

[Window Functions](#)

Cadence User Interface SKILL Reference

Window Functions

hiGetIconName

```
hiGetIconName(  
    w_windowId  
)  
=> t_iconName / nil
```

Description

Gets the name that is displayed on a window icon when the window is minimized. This name is set with the `hiSetIconName` function.

A session window, when minimized, displays the icon name of the window that is currently active within it, unless it is not set, in which case it displays the session window icon name. This function, however, always returns the session icon name when passed a session window.

Note: This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<code>w_windowId</code>	The ID of the window, session window, or dockable window.

Value Returned

<code>t_iconName</code>	The icon name.
<code>nil</code>	The window is invalid or an icon name has not been set for it.

Related Topics

[Window Functions](#)

[hiSetIconName](#)

[hiSetWindowIcon](#)

hiGetScreenPoint

```
hiGetScreenPoint(  
    w_windowId  
)  
=> l_point / nil
```

Description

Returns the current location of the pointer relative to the upper-left corner of the drawable area of a window. This function does not apply to session windows or dockable windows.

Even if the pointer is currently over another window that obscures *w_windowId*, this function returns its location as long as it is within the boundaries of *w_windowId*. The location is in pixels relative to the upper left corner of the drawable area.

Argument

<i>w_windowId</i>	ID of the window. The window cannot be a session window or dockable window.

Value Returned

<i>l_point</i>	Location of the pointer in pixels relative to the upper left corner of the drawable area.
<i>nil</i>	The pointer is not within the drawable area of the window or the window is hidden behind other tabs in a session window.

Related Topics

[Window Functions](#)

[hiGetDrawThruDelta](#)

[hiOpenWindow](#)

[hiGetDbuPoint](#)

hiGetSessionWindow

```
hiGetSessionWindow(  
    w_windowID  
)  
=> w_sessionWindowID / nil
```

Description

Returns the ID of the session window that contains the specified window or dock window.

Argument

<i>w_windowID</i>	The window ID or dock window ID.

Value Returned

<i>w_sessionWindowID</i>	The session window that contains the window or dock window.
<i>nil</i>	<i>w_windowID</i> is not associated with a session window.

Examples

```
hiGetSessionWindow( swindow ( 6 ) )
```

Related Topics

[Window Functions](#)

[hiGetActiveTabWindow](#)

[hiGetCurrentWindow](#)

[hilsWindowSpecifier](#)

[windowp](#)

[swindow](#)

[hiUndockWindow](#)

[hiOpenWindow](#)

Cadence User Interface SKILL Reference

Window Functions

hiGetWindowState

```
hiGetWindowState(  
    w_windowId  
)  
=> s_state / nil
```

Description

Returns the display state of a window: mapped, unmapped, or iconified.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<i>w_windowId</i>	The ID of the window, session window, or dockable window.

Value Returned

<i>s_state</i>	Returns one of the following symbols: mapped, unmapped, or iconified.
<i>nil</i>	The window, or the session window that contains it, has not been displayed yet with <code>hiDisplayWindow</code> . If <i>w_windowID</i> is a session window, the session window, or any window it contains, has not been displayed yet.

Example

Returns the display state of the `win3` window.

```
hiGetWindowState( win3 )  
=> unmapped
```

Returns the display state of the `sessionWin` window.

```
hiGetWindowState( sessionWin )  
=> mapped
```

Related Topics

[Window Functions](#)

Cadence User Interface SKILL Reference

Window Functions

hiGetWindowDisplayState

hiGetWindowDisplayState

```
hiGetWindowDisplayState(  
    w_windowId  
)  
=> l_state / nil
```

Description

Returns detailed information about the display state of a window.

Argument

<i>w_windowID</i>	The ID of the window, session window, or dockable window.

Value Returned

<i>l_state</i>	A list of keywords that describe the display state of the window. Valid values for the first element of the list are: ■ mapped ■ unmapped ■ iconified Valid values for the second element of the list are: ■ session: a session window ■ topTab: a window in a session window that is currently displayed ■ hiddenTab: a window in a session window that is hidden behind other tabs ■ docked: a docked dockable window ■ floated: a floating dockable window ■ main: a window that is either not within a session window or is the only window within a session window.

Cadence User Interface SKILL Reference

Window Functions

`nil` The window, or the session window that contains it, has not been displayed yet with `hiDisplayWindow`. If `w_windowID` is a session window, the session window, or any window it contains, has not been displayed yet.

Examples

Returns detailed information of the display state of the `win3` window.

```
hiGetWindowDisplayState( win3 )  
=> (unmapped hiddenTab)
```

Returns detailed information of the display state of the `win6` window.

```
hiGetWindowDisplayState( win6 )  
=> (mapped topTab)
```

Returns detailed information of the display state of the `sessionWin` window.

```
hiGetWindowDisplayState( sessionWin )  
=> (mapped session)
```

Related Topics

[Window Functions](#)

[hiGetWindowState](#)

hiGetWindowIconifyState

```
hiGetWindowIconifyState(  
    w_windowId  
)  
=> l_iconPosition / nil
```

Description

Returns the icon position of an iconified window. If the window is contained in a session window, this function returns the icon position of the session window.

w_windowId can also be a session window; it cannot be a dock window.

To ensure that a restored window is in the correct icon state when a saved session is restored, you must pass the value returned by this function as the *l_iconPosition* argument of the function `hiOpenWindow` or `hiCreateWindow` when you log these functions to the save/restore file.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Cadence User Interface SKILL Reference

Window Functions

Argument

<code>w_windowId</code>	ID of the window for which you need the icon position. The window can be a session window but it cannot be a dock window.

Value Returned

<code>l_iconPosition</code>	The icon position (a list of the x and y coordinates) of the iconified window. The x and y coordinates correspond to the upper-left corner of the icon (0:0 is the upper-left corner of the screen).
<code>nil</code>	The window is not iconified.

Related Topics

Related Topics

[Window Functions](#)

[hiOpenWindow](#)

[hiUndockWindow](#)

[hiIconifyWindow](#)

[hiDeiconifyWindow](#)

hiGetWindowList

```
hiGetWindowList(  
    [ s_windowType ]  
)  
=> l_windowId
```

Description

Returns a list of window IDs of all windows that have been created but not closed, including those that are not currently displayed. You can specify the type of window with the *s_windowType* argument. By default, all windows except session windows and dock windows are returned.

The Command Interpreter Window is always included in this list.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<i>s_windowType</i>	Specifies one of the following symbols: 'all: Returns all windows including session windows and dockable windows 'session: Returns all session windows 'dockable: Returns all dockable windows 'window: Returns all windows except session windows and dockable windows Default value: 'window

Values Returned

<i>l_windowId</i>	A list of window IDs.

Examples

Returns a list of all the active windows excluding session and dockable windows.

```
hiGetWindowList( )  
=> (window:1 window:2 window:3)
```

Cadence User Interface SKILL Reference

Window Functions

```
hiGetWindowList('window)
=> (window:1 window:2 window:3)
```

Returns a list of all the active session windows.

```
hiGetWindowList('session)
=> (swindow:1 swindow:2)
```

Returns a list of all the active dockable windows.

```
hiGetWindowList('dockable)
=> (dwindow:1 dwindow:3 dwindow:4)
```

Returns a list of all the active windows.

```
hiGetWindowList('all)
=> (window:1 window:2 window:3 swindow:1 dwindow:1 dwindow:3 swindow:2 dwindow:4)
```

Related Topics

Window Functions

hiGetWindowName

```
hiGetWindowName(  
    w_windowId  
)  
=> t_windowName / nil
```

Description

Returns the window name, which is displayed in the window manager frame of a window. This name is set with the `hiSetWindowName` function.

A session window displays the window name of the window that is currently active within it, unless it is not set, in which case it displays the session window name. This function, however, always returns the session window name when passed a session window.

Note: This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Argument

<code>w_windowId</code>	The ID of the window, session window, or dockable window whose name you want to get.

Value Returned

<code>t_windowName</code>	The window name.
<code>nil</code>	The window is invalid or the window does not have a name associated with it (such as a plot window).

Related Topics

[Window Functions](#)

[hiSetWindowName](#)

hiOpenWindow

```
hiOpenWindow(  
    [ ?bBox l_boxSpec ]  
    [ ?type t_widgetType ]  
    [ ?appType t_applicationType ]  
    [ ?menus l_menus ]  
    [ ?labels l_labels ]  
    [ ?help t_helpString ]  
    [ ?scroll g_scrollbars ]  
    [ ?form g_form ]  
    [ ?closeProc s_closeProc ]  
    [ ?iconPosition l_iconPosition ]  
    [ ?session g_sessionWindow ]  
    [ ?tabTitle g_tabTitle ]  
    [ ?tabIcon g_tabIcon ]  
    [ ?tabTip g_tabTip ]  
)  
=> w_windowId / nil
```

Description

Creates and displays a window. This function displays the new window in a session window if the session window argument is specified or if the new window is a graphics window.

If the session window already contains windows, a new tab is created for the new window and it is raised to the top.

The new window becomes the current window. If you do not want this window ever to become the current window, you can set the window property `neverCurrentWindow` to `t`:

```
window->neverCurrentWindow = t
```

When this property is set on a current window, the current window gets reset to the last current window or to another open window. If another window cannot be found, the current window resets to `nil`.

Setting the `neverCurrentWindow` property can affect previously saved replay files that contain this window, and these log files might not replay correctly.

If you set the `neverCurrentWindow` property to `t` and you use `.tgf` help files, you need to specify the correct application ID for help to work correctly. See the note in the description of the `t_help` argument for details.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

Cadence User Interface SKILL Reference

Window Functions

Arguments

- `?bBox` *l_boxSpec* Lower left and upper right corners of the window in screen coordinates (assuming 0:0 as the lower left corner of the screen):
- ```
list(x1:y1 x2:y2 [iconCoordinate|nil])
```
- where *x1* is the lower left *x* coordinate, *y1* is the lower left *y* coordinate, *x2* is the upper right *x* coordinate and *y2* is the upper right *y* coordinate. See the *l\_iconPosition* argument for information about the optional, third element of the list.
- The symbols `'default` and `'interactive` are also valid values for this argument.
- If *l\_boxSpec* is `nil`, either you are prompted with a rubberbanding box to enter a window size, or a default size is used, depending upon the value of the global variable `hivWinStyle`.
- You can use the `hiIsWindowSpecifier` function to check if your *l\_boxSpec* value is valid.
- `?type` *t\_widgetType* One of the following strings: `graphics`, `text`, `hypertext`, `html`, `encap`, `form`, or `none` (plot windows) representing the type of window widget to be created. If *widgetType* is not specified, the default is `graphics`. A graphics window is always enclosed in a session window, regardless of whether you specify the `?session` argument.
- For information about how to create hypertext, see [Hypertext Viewfile Window](#).
- `?appType` *t\_applicationType* String representing the window application type or name. This string is also used as the window's bindkey prefix. If not specified, *t\_applicationType* defaults to `" "`.
- `?menus` *l\_menus* List of menus to appear in the window banner. Each menu should be either the return value of [hiCreatePulldownMenu](#) or the menu symbol representing that value. These menus are placed from left to right on the window banner in the order listed. Each menu can be placed only once on a window banner.

## Cadence User Interface SKILL Reference

### Window Functions

---

`?labels l_labels` List of strings to appear in the label banner. These labels are placed from left to right on the label banner in the order listed. A maximum of 4 labels can be placed on each window.

`?help t_helpString`

Help string for the window, which is used to reference a `.tgf` help file. If you do not specify this argument or specify `" "` (an empty string), the help argument is set to the `appName` (which is the value of the `t_appType` argument).

The help string for a window can also be set at any time using the `hiSetWindowsAtts` functions and can be retrieved with the `hiGetHelp` function.

Cadence applications may modify their window help strings at any time.

When a user clicks *Help* in a window, the help string is used for the help symbol that is looked for in the `.tgf` file. The `appName` is used to determine the `.tgf` file to search.

The help string can also be used to modify the `appName` by assigning an `hiHelpAppName` property to the symbol representation of the string.

For example, `putprop ( concat( help_string )  
app_name_string 'hiHelpAppName )`

set the `appID` used for Help to `app_name_string` whenever `help_string` is used as the help string for any window.

For more information about the online help mechanism, see [Online Help Mechanism](#)

For windows that cannot become the current window (because the `neverCurrentWindow` property is set to `t`), to specify the correct application ID for accessing help, specify a help name string for the `t_help` argument of `hiCreateWindow()` or `hiOpenWindow()` and then attach the `hiHelpAppName` property to the symbol of the same name as the help string. For example: `putprop( 'MyAppHelp "MyApp"  
'hiHelpAppName)`

For windows help to work correctly, either the `t_help` or `t_appType` argument or both must be set when you create the window.

## Cadence User Interface SKILL Reference

### Window Functions

---

`?scroll g_scrollbars`

Either `t` or `nil` to specify the use of scroll bars. This field is valid only for windows with a *widgetType* of `graphics`. If set to `t`, scroll bars appear on the right and bottom of the window. If not specified, the property `useScrollbars` from the CIW is used. This value can be set from the User Preferences form.

`?form g_form`

SKILL representation of a form returned by `hiCreateAppForm` or `hiCreateForm`. This argument is valid only for windows with a *widgetType* of `form`. When the window containing the form is closed, the form is not destroyed. It can be redisplayed in another window.

`?closeProc S_closeProc`

String or symbol for the name of the procedure to be performed when the specified window is closed. This closing procedure is strictly for clean up and must not call `hiCloseWindow`, any enter functions, or any blocking functions that display forms or dialog boxes. Because this clean-up procedure is called no matter how you close the window (from a window manager or through Cadence software), you might need to delete existing calls to clean-up functions from the user interface.

A close procedure is called before any SKILL close procedures. If the C procedure returns `nil`, none of the SKILL close procedures are called and the window remains open.

`?iconPosition l_iconPosition`

Icon location (in X Window System screen coordinates) for the window. This causes the window to be iconified initially. If *l\_iconPosition* is `nil` or not specified, the window is not iconified initially.

In addition to the `?iconPosition` keyword argument, the *l\_boxSpec* argument can also contain the icon position. If both methods are used, `?iconPosition` overrides the position passed in *l\_boxSpec* unless *l\_iconPosition* is `nil`.

`?session g_sessionWindow`

The session window in which you want to display the new window.

## Cadence User Interface SKILL Reference

### Window Functions

---

*?tabTitle g\_tabTitle*

The tab title for the new window. If the session window already contains windows, a tab is created for the new window.

After the window is created, you can use the `hiSetTabWindowTitle` function to change the tab title.

*?tabIcon g\_tabIcon*

The icon to be displayed on the tab for the new window.

After the window is created, you can use the `hiSetTabWindowIcon` function to change the tab icon.

*?tabTip g\_tabTip*

The tooltip that is displayed whenever the cursor hovers over the tab.

After the window is created, you can use the `hiSetTabWindowTip` function to change the tab tooltip.

### Values Returned

*w\_windowId*

The window ID of the created window.

*nil*

The window is not created. An error message is also issued.

### Examples

```
hiOpenWindow(?bBox list (0:0 300:300)
?type "graphics"
?menus list (hiOpenMenu 'hiDMMenu)
?labels '("label 1" "label 2")
?scroll t
)
=> window:2

hiOpenWindow(?bBox 'interactive
?type "form"
?appType "formdow"
?form hiCreateForm('myform "My Form" ""
list (hiCreateStringField(?name 's ?prompt "Enter name:"))
"")
)
=> window:3
```

### Related Topics

#### Window Functions



## Cadence User Interface SKILL Reference

### Window Functions

---

hiUndockWindow

hiSetTabWindowTitle

hiSetTabWindowIcon

hiSetTabWindowTip

hilsWindowSpecifier

hilsWindowSpecifier

## hiUndockWindow

```
hiUndockWindow(
 x_dwindowID
)
=> t / nil
```

### Description

Undocks a dockable window from a session window. The window is floated. You can dock the window again with `hiDockWindow`.

The first time a dock window is floated, its location is the default Qt location and its size is the initial float hint specified when the window was created. Subsequently, the last location and size are used.

### Argument

|                          |                                             |
|--------------------------|---------------------------------------------|
| <code>x_dwindowID</code> | The dockable window that you want to float. |
|--------------------------|---------------------------------------------|

### Value Returned

|                  |                          |
|------------------|--------------------------|
| <code>t</code>   | The window was undocked. |
| <code>nil</code> | There was an error.      |

### Example

```
hiUndockWindow(dw)
```

### ***Related Topics***

[Window Functions](#)

[hiCreateDockWindow](#)

[hiUndockWindow](#)

[hilsWindowSpecifier](#)

## hIsWindowSpecifier

```
hIsWindowSpecifier(
 g_winspec
)
=> g_winspec / nil
```

### Description

Validates a window specifier, which is passed as the *boxSpec* argument to the `hiCreateWindow` and `hiOpenWindow` functions.

### Argument

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>g_winspec</i> | Window specifier.<br>Valid values are: <ul style="list-style-type: none"><li>■ 'default</li><li>■ 'interactive</li><li>■ A bounding box: <code>list( <i>x1</i>:<i>y1</i> <i>x2</i>:<i>y2</i>)</code><br/>where <i>x1</i> is the lower left x coordinate, <i>y1</i> is the lower left y coordinate, <i>x2</i> is the upper right x coordinate, and <i>y2</i> is the upper right y coordinate.</li><li>■ A list of bounding box and iconify state:<br/><code>list( <i>bBox</i> <i>iconifyState</i>)</code><br/>where <i>iconifyState</i> is nil or a point.</li></ul> |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Values Returned

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>g_winspec</i> | The window specifier.                             |
| nil              | <i>g_winspec</i> is not a valid window specifier. |

### ***Related Topics***

[Window Functions](#)

[hiOpenWindow](#)

[hiUndockWindow](#)

## hiSetDrawThruDelta

```
hiSetDrawThruDelta(
 x_defaultDelta
 [x_highDelta [x_mouseButton]]
)
=> t
```

### Description

Sets the default draw-thru delta and, optionally, the high draw-thru delta for mouse buttons 1, 2 or 3.

### Arguments

*x\_defaultDelta*

The default draw-thru delta. The default number of pixels that a mouse cursor must travel, with a mouse button held down, to be interpreted as a draw-thru command; if the mouse button is released after the double-click time expires.

The default draw-thru delta cannot be less than 3. If the value passed is less than 3, it automatically resets to 3.

If a default draw-thru value is set that is larger than the current default draw-thru delta and it is larger than any of the high draw-thru deltas current set any high draw-thru delta smaller than the new default delta value gets changed to equal to the new default delta value.

*x\_highDelta*

The distance that the mouse cursor must move, in pixels, to be interpreted as a draw-thru command, if the time between the mouse-button press and the mouse-button release is less than the double-click time.

To change a *x\_highDelta*, without changing the *x\_defaultDelta* value, specify 0 for *x\_defaultDelta*.

If *x\_highDelta* is specified, it cannot be less than *x\_defaultDelta*, and resets to the value of the default delta if a lower value is specified.

## Cadence User Interface SKILL Reference

### Window Functions

---

*x\_mouseButton* Specifies the mouse button (1, 2, or 3) that *x\_highDelta* is to be applied to.

If this argument is not specified, then *x\_highDelta* is applied to mouse button 1.

If a number other than 1, 2, or 3 is specified, a warning is displayed.

#### Value Returned

*t* Always returns *t*. If the argument specified is invalid, a warning is also displayed.

#### Examples

- `hiSetDrawThruDelta( 6 )` changes the default draw-thru delta to 6.
- `hiSetDrawThruDelta( 5 10 )` sets the default draw-thru delta to 5 and the high draw-thru delta for mouse button 1 to 10.
- `hiSetDrawThruDelta( 0 25 2 )` sets the high draw-thru delta for mouse button 2 to 25 without changing the default draw-thru delta value.
- `hiSetDrawThruDelta( 0 )` does nothing.

#### ***Related Topics***

[Window Functions](#)

[hiGetDrawThruDelta](#)

[hiGetDbuPoint](#)

[hiOpenWindow](#)

## hiSetSessionFocus

```
hiSetSessionFocus(
 w_dockWindowOrTabWindow
)
=> t / nil
```

### Description

Sets the keyboard focus to a session window's active tab window or docked window when the session window becomes active. The active tab window or docked window that you specify must be part of a session window and must contain a sub-widget that is enabled, visible, and that can get focus through the `Tab` key. The focus is set to a sub-widget.

Keyboard focus will be active only when the session window has the window manager focus; this function does not take focus away from other applications or top-level windows.

### Argument

*w\_dockWindowOrTabWindow*    A session window active tab window or docked window.

### Value Returned

|            |                                                                     |
|------------|---------------------------------------------------------------------|
| <i>t</i>   | Session focus was set to a sub-widget of the specified window.      |
| <i>nil</i> | The specified window is invalid or a sub-widget could not be found. |

### Examples

```
hiSetSessionFocus(hiGetActiveTabWindow(swindow (1)))
```

### ***Related Topics***

[Window Functions](#)

## hiSetTabWindowTitle

```
hiSetTabWindowTitle(
 w_windowID
 t_title
)
=> t / nil
```

### Description

Sets the title of the specified window's tab. Tabs are displayed when multiple windows are open in a session window.

This function overrides any initial value that might have been set when the window was created.

### Arguments

|                   |                            |
|-------------------|----------------------------|
| <i>w_windowID</i> | The window ID.             |
| <i>t_title</i>    | The new title for the tab. |

### Value Returned

|            |                        |
|------------|------------------------|
| <i>t</i>   | The tab title was set. |
| <i>nil</i> | There was an error.    |

### Example

```
hiSetTabWindowTitle(window (4) "myDesign")
```

### ***Related Topics***

[Window Functions](#)

[hiOpenWindow](#)

[hiSetTabWindowIcon](#)

[hiSetTabWindowTip](#)

## hiSetTabWindowIcon

```
hiSetTabWindowIcon(
 w_windowID
 g_icon
)
=> t / nil
```

### Description

Sets the icon in the specified window's tab. Tabs are displayed when multiple windows are open in a session window.

This function overrides any initial value that might have been set when the window was created.

### Arguments

|                   |                           |
|-------------------|---------------------------|
| <i>w_windowID</i> | The window ID.            |
| <i>g_icon</i>     | The new icon for the tab. |

### Values Returned

|            |                       |
|------------|-----------------------|
| <i>t</i>   | The tab icon was set. |
| <i>nil</i> | There was an error.   |

### Examples

```
hiSetTabWindowIcon(window (4))
```

### ***Related Topics***

[hiOpenWindow](#)

[hiSetTabWindowTitle](#)

[hiSetTabWindowTip](#)



## hiSetTabWindowTip

```
hiSetTabWindowTip(
 w_windowID
 t_toolTip
)
=> t / nil
```

### Description

Sets the tooltip for the specified window's tab. Tabs are displayed when multiple windows are open in a session window.

This function overrides any initial value that might have been set when the window was created.

### Arguments

|                   |                                                                                     |
|-------------------|-------------------------------------------------------------------------------------|
| <i>w_windowID</i> | The window ID.                                                                      |
| <i>t_toolTip</i>  | The new tooltip. The tooltip is displayed when the mouse hovers over the tab title. |

### Values Returned

|            |                      |
|------------|----------------------|
| <i>t</i>   | The tooltip was set. |
| <i>nil</i> | There was an error.  |

### Example

```
hiSetTabWindowTip(window (4) "Top Cell: myDesign")
```

### ***Related Topics***

[hiOpenWindow](#)

[hiSetTabWindowTitle](#)

[hiSetTabWindowIcon](#)

## hiSetWindowAtts

```
hiSetWindowAtts(
 [?window w_windowId]
 [?appType t_applicationType]
 [?help t_help]
 [?scroll g_scrollbars]
)
=> t / nil
```

### Description

Changes the application type, help string, or scroll bar attributes of a window. You must register the new application type before setting it with this function.

Only the help string can be set for session windows.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

*?window w\_windowId*

The ID of the window, dockable window, or session window whose attributes you want to change. Only the help string can be set for session windows. If *w\_windowId* is not specified, the current window is used.

*?appType t\_applicationType*

String representing the window application type or name.

You must register the new application type before using this function. For example, if you are setting an application type to "chpr", you must have registered it using the command:

```
hiRegisterBindKeyPrefix("chpr" "encap")
```

*?help t\_help*

String used as one of the keywords that Help logic uses when it constructs the names of help files and help symbols. See [Online Help Mechanism](#) for more information. If this argument is not specified, it defaults to the value of *t\_applicationType*.

*?scroll g\_scrollbars*

## Cadence User Interface SKILL Reference

### Window Functions

---

Either `t` or `nil` to specify the use of scroll bars. If set to `t`, scroll bars appear on the right and bottom of the window. If not specified, the property `useScrollbars` from the CIW is used.

#### Value Returned

|                  |                                                                                                                                                                                                                                 |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | The application type or help string, or both, are set. This function returns <code>t</code> whether <code>g_scrollbars</code> is set or not.                                                                                    |
| <code>nil</code> | Neither the application type nor the help string are set. If you are setting both the application type and the help string at the same time, then <code>nil</code> gets return even if only one of these attributes is not set. |

#### ***Related Topics***

[hiRegisterBindKeyPrefix](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiSetWinStyle

```
hiSetWinStyle(
 s_style
)
=> t / nil
```

#### Description

Sets the window creation style to 'default, 'interactive or 'bBox.

#### Argument

|                |                                                                                                                                                                                                                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>s_style</i> | One of the following symbols: 'default, 'interactive or 'bBox. If you specify 'default, new windows appear with the size and location specified programmatically. If you specify 'interactive, you must place new windows yourself interactively. 'bBox is an alternative value for 'interactive. |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Values Returned

|     |                     |
|-----|---------------------|
| t   | The style is set.   |
| nil | There was an error. |

#### ***Related Topics***

[Window Functions](#)

[hiUndockWindow](#)

[hiOpenWindow](#)

## hiRegCloseProc

```
hiRegCloseProc(
 w_windowId
 s_closeProc
)
=> t
```

### Description

Registers the procedure to be called when a window is closed. You can register closing procedures for windows, session windows, and dockable windows. A session window's closing procedure is called before the closing procedure of any of the windows it contains.

If the session window's closing procedure returns `nil`, the closing procedures of the windows it contains are aborted.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>  | ID of the window, session window, or dockable window for which you are registering the closing procedure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>s_closeProc</i> | <p>Name of the procedure to be executed when the window is closed. The format for this procedure should be <code>closeProc(windowId)</code>. This closing procedure is strictly for clean up and must not call <code>hiCloseWindow</code>, any enterfunctions, or any blocking functions that display forms or dialog boxes. Because this procedure is called regardless of how you close the window (from a window manager or through Cadence software), you might need to delete existing calls to clean-up functions from the user interface.</p> <p>A C closing procedure is called before any SKILL closing procedures. If the C procedure returns <code>nil</code>, none of the SKILL closing procedures are called and the window remains open.</p> |

## Cadence User Interface SKILL Reference

### Window Functions

---

#### Value Returned

t                                      The closing procedure is registered.

#### Example

```
hiRegCloseProc(window (2) 'myFunction)
=> t
```

#### ***Related Topics***

[Window Functions](#)

## hiUnregCloseProc

```
hiUnregCloseProc(
 w_windowId
 s_closeProc
)
=> t / nil
```

### Description

Unregisters the closing procedure for a window.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Arguments

|                    |                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------|
| <i>w_windowId</i>  | ID of the window, session window, or dockable window associated with the closing procedure. |
| <i>s_closeProc</i> | Closing procedure to be unregistered.                                                       |

### Value Returned

|            |                                            |
|------------|--------------------------------------------|
| <i>t</i>   | The closing procedure is unregistered.     |
| <i>nil</i> | The closing procedure is not unregistered. |

### Example

```
hiUnregCloseProc(window (2) 'myFunction)
```

### Related Topics

[Window Functions](#)

## hiSwitchWindowType

```
hiSwitchWindowType (
 w_windowId
 t_widgetType
 t_appType
 t_help
 [g_scroll]
)
=> t / nil
```

### Description

Switches the widget type of a window between graphics, text, or hypertext. The window must be in a session window.

The application type, help string, and scroll bar option can also be changed.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                     |                                                                                                                                                                                                                                                                                                                                              |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>   | Window you want to change, which must be one of the following types: graphics, text, or hypertext. The window must be contained in a session window. <i>w_windowId</i> cannot be a dock window or a session window.                                                                                                                          |
| <i>t_widgetType</i> | One of the following strings: text, hypertext, or graphics.                                                                                                                                                                                                                                                                                  |
| <i>t_appType</i>    | String representing the window application type or name.                                                                                                                                                                                                                                                                                     |
| <i>t_help</i>       | String used as one of the key words that Help logic uses when it is constructing the names of the various help files and/or help symbols searched for in the help files. Refer to <a href="#">Online Help Mechanism</a> for more information. If this argument is not specified, it defaults to the same value as <i>t_applicationType</i> . |
| <i>g_scroll</i>     | Either <i>t</i> or <i>nil</i> to specify the use of scroll bars. If <i>g_scroll</i> is set to <i>t</i> , scroll bars appear on the right and bottom of the window. If <i>g_scroll</i> is not specified or <i>nil</i> , scroll bars are not displayed.                                                                                        |



## Cadence User Interface SKILL Reference

### Window Functions

---

#### Value Returned

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| <code>t</code>   | The window type is switched.                                      |
| <code>nil</code> | The window type is not switched. An error message is also issued. |

#### Examples

Switches the window type of `window (4)`.

```
hiSwitchWindowType(window (4) "text" "Layout" "layoutHelp")
=> t
```

#### ***Related Topics***

[Window Functions](#)

## hilconifyWindow

```
hilconifyWindow(
 w_windowId
)
=> t / nil
```

### Description

Converts a window to its icon. If the window is contained in a session window, the session window is iconified.

*w\_windowId* can be a window or a session window but it cannot be a dock window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Argument

|                   |                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want to iconify. <i>w_windowId</i> can be a window or a session window. It cannot be a dock window. |
|-------------------|----------------------------------------------------------------------------------------------------------------|

### Values Returned

|     |                                                             |
|-----|-------------------------------------------------------------|
| t   | The window is iconified.                                    |
| nil | The window is not iconified. An error message is displayed. |

### Related Topics

[Window Functions](#)

[hiDeiconifyWindow](#)

[hiLowerWindow](#)

## hiLowerWindow

```
hiLowerWindow(
 w_windowId
)
=> t / nil
```

### Description

Lowers a window to the bottom of the list of visible windows.

When this function is:

- Passed a session window ID – The specified window is lowered
- Passed a window ID and the session window does not contains multiple tabs – The specified window is lowered
- Passed a window ID and the session window contains multiple tabs – The function behaves similarly to [hiLowerTab](#).

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Argument

|                   |                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want to lower. <i>w_windowId</i> can also be a session window or a floating dock window. |
|-------------------|-----------------------------------------------------------------------------------------------------|

### Value Returned

|            |                                                                                       |
|------------|---------------------------------------------------------------------------------------|
| <i>t</i>   | The window is lowered to the bottom of the list.                                      |
| <i>nil</i> | The window is not lowered to the bottom of the list. An error message is also issued. |

### ***Related Topics***

[Window Functions](#)

[hiRaiseWindow](#)

[hiLowerTab](#)

## hiRaiseWindow

```
hiRaiseWindow(
 w_window
 [g_silent]
)
=> t / nil
```

### Description

Raises a window to the top of the list of visible windows.

If the window is within a session window, its tab is raised to the top of the tab stack, it becomes the active window, and the session window that contains it will be raised. If the window is a session window, it will be raised. If the window is a floating dock window, it is raised. If the window is docked, the session window that contains it gets raised and the visibility of tabs does not get affected.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                 |                                                                                                                                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | Window you want to raise. <i>w_windowId</i> can be either a window ID of a regular, session, or dock window; or the window number of a regular window.                                             |
| <i>g_silent</i> | If this argument is specified, the <code>Window does not exist</code> message does not get displayed. This argument can only be specified if the value of the <i>w_window</i> argument is integer. |

### Values Returned

|            |                                                                                   |
|------------|-----------------------------------------------------------------------------------|
| <i>t</i>   | The window is raised to the top of the list.                                      |
| <i>nil</i> | The window is not raised to the top of the list. An error message is also issued. |

### Related Topics

[Window Functions](#)

[hiLowerWindow](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiRaiseTab

hiLowerTab

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiRaiseTab

```
hiRaiseTab(
 w_windowID
)
=> t / nil
```

#### Description

Raises a tab contained in a multi-tabbed session window to the top of the list and makes it the active tab. If there is only a single tab, the function does nothing.

#### Argument

|                   |                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------|
| <i>w_windowID</i> | The ID of the window you want to raise. The window must be contained in a session window. |
|-------------------|-------------------------------------------------------------------------------------------|

#### Values Returned

|            |                           |
|------------|---------------------------|
| <i>t</i>   | The window is raised.     |
| <i>nil</i> | The window is not raised. |

#### ***Related Topics***

[Window Functions](#)

[hiLowerTab](#)

[hiRaiseWindow](#)

[hiLowerWindow](#)

## hiLowerTab

```
hiLowerTab(
 w_windowID
)
=> t / nil
```

### Description

Lowers a tabbed window contained in a session window to the bottom of the list of tabbed windows.

If there is only a single tab, the function does nothing.

If there are multiple tabs, and the function is passed the window ID of the current tab, and it cycles to the next tab in the sequence.

If passed the window ID of one of the tabs that is not the current tab, the function does nothing.

Only a window ID should be specified for this function. If a session window ID is specified, the function displays a warning and return `nil`.

### Argument

|                         |                                                                                           |
|-------------------------|-------------------------------------------------------------------------------------------|
| <code>w_windowID</code> | The ID of the window you want to lower. The window must be contained in a session window. |
|-------------------------|-------------------------------------------------------------------------------------------|

### Values Returned

|                  |                            |
|------------------|----------------------------|
| <code>t</code>   | The window is lowered.     |
| <code>nil</code> | The window is not lowered. |

### ***Related Topics***

[Window Functions](#)

[hiRaiseTab](#)

[hiRaiseWindow](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiLowerWindow



## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiPickWindow

```
hiPickWindow()
=> w_windowId / nil
```

#### Description

Lets you select a window without changing the current window. When this function is called, the pointer is grabbed and not released until either the mouse button or the *Escape* key is pressed. Any other keyboard event is ignored and the mouse click is not passed on to anything else.

#### Argument

None

#### Value Returned

|                         |                                                                                                          |
|-------------------------|----------------------------------------------------------------------------------------------------------|
| <code>w_windowId</code> | The <code>windowId</code> of the window in which the user clicks the mouse button.                       |
| <code>nil</code>        | The user selects the <i>Escape</i> key or clicks the mouse button when the pointer is not over a window. |

#### ***Related Topics***

[Window Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiMapWindow

```
hiMapWindow(
 w_windowId
)
=> t / nil
```

### Description

Displays a window, dockable window, or session window on screen.

If the window is within a session window, its tab is raised to the top and displayed. If the window is a dock window, it is mapped to its last state—docked or floating. If the window has been iconified, it is deiconified.

The window must have been displayed at one time with `hiDisplayWindow`, `hiOpenWindow`, or `hiDockWindow`. This function is usually used in conjunction with `hiUnmapWindow`, which removes a window from the screen without closing it.

**Note:** Do not use this function to display dock windows initially, use `hiDockWindow` instead.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Argument

|                         |                                                              |
|-------------------------|--------------------------------------------------------------|
| <code>w_windowId</code> | Window, dock window, or session window that you want to map. |
|-------------------------|--------------------------------------------------------------|

### Value Returned

|                  |                                                                                                             |
|------------------|-------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | The window is redisplayed.                                                                                  |
| <code>nil</code> | <code>w_windowId</code> is invalid or the window has never been displayed. An error message is also issued. |

### Related Topics

[Window Functions](#)

[hiOpenWindow](#)

[hiUndockWindow](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

hiUnmapWindow

## hiUnmapWindow

```
hiUnmapWindow(
 w_windowId
)
=> t / nil
```

### Description

Removes a window, dockable window, or session window from the screen without closing or destroying it.

If the window is a tabbed window in a session window, the tab is moved to the back. If the window is the only window in a session window, then the session window is also unmapped.

The window must have been displayed at one time with `hiDisplayWindow`, `hiOpenWindow`, or `hiDockWindow`. This function is usually used in conjunction with `himapWindow`.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Argument

|                         |                                                                |
|-------------------------|----------------------------------------------------------------|
| <code>w_windowId</code> | Window, dock window, or session window that you want to unmap. |
|-------------------------|----------------------------------------------------------------|

### Value Returned

|                  |                                                                                                             |
|------------------|-------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | The window is unmapped.                                                                                     |
| <code>nil</code> | <code>w_windowId</code> is invalid or the window has never been displayed. An error message is also issued. |

### ***Related Topics***

[Window Functions](#)

[hiOpenWindow](#)

[hiUndockWindow](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

hiMapWindow

## hiMoveWindow

```
hiMoveWindow(
 w_windowId
 l_point
)
=> t / nil
```

### Description

Moves the lower left corner of a window to a new screen location. If *w\_windowId* is a window contained in a session window, the session window is moved.

The lower-left corner of the window itself (without the window manager border) is placed at the new location. The window, therefore, might not appear to be placed exactly at the specified point.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                   |                                                                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window, session window, or floating dockable window that you want to move. <i>w_windowId</i> cannot be a dockable window that is currently docked. |
| <i>l_point</i>    | The new lower-left position of the window specified in screen coordinates. This assumes that 0:0 is the lower-left corner of the screen.           |

### Values Returned

|            |                                                           |
|------------|-----------------------------------------------------------|
| <i>t</i>   | The window is moved.                                      |
| <i>nil</i> | The window is not moved. An error message is also issued. |

### Examples

Moves the lower left corner of `hiGetCurrentWindow` to 50:100 coordinate.

```
hiMoveWindow(hiGetCurrentWindow() 50:100)
=> t
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

Window Functions

## hiResizeWindow

```
hiResizeWindow(
 w_windowId
 l_bBox
)
=> t / nil
```

### Description

Resizes a window to the size of a bounding box. If *w\_windowId* is a window within a session window, the session window is resized.

The window manager banner is not considered part of the bounding box. The window, therefore, might not appear to be placed exactly at the specified points.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                   |                                                                                                                                                                                                                                                  |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window, session window, or floating dockable window that you want to resize. <i>w_windowId</i> cannot be a dockable window that is currently docked.                                                                                             |
| <i>l_bBox</i>     | The new coordinates for the bounding box of the window. Specify the lower-left and upper-right screen coordinates, assuming 0:0 to be the lower-left corner of the screen:<br><br><i>((lowerLeft_X lowerLeft_Y) (upperRight_X upperRight_Y))</i> |

### Value Returned

|            |                                                             |
|------------|-------------------------------------------------------------|
| <i>t</i>   | The window is resized.                                      |
| <i>nil</i> | The window is not resized. An error message is also issued. |

### Examples

Resizes window (4) to 0:0 500:500 coordinate.

```
hiResizeWindow(window(4) list(0:0 500:500))
=> t
```



## Cadence User Interface SKILL Reference

### Window Functions

---

Resizes `window(2)` to match the size and position of `window(1)` by using the `hiGetAbsWindowScreenBBox` and `hiResizeWindow` functions:

```
win1BBox = hiGetAbsWindowScreenBBox(window(1) t)
hiResizeWindow(window(2) win1BBox)
```

#### ***Related Topics***

[Window Functions](#)

[hiGetAbsWindowScreenBBox](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiSetWindowName

```
hiSetWindowName(
 w_windowId
 t_label
)
=> t / nil
```

#### Description

Sets the window name, which is displayed in the window manager frame of a window. A session window displays the window name of the window that is currently active within it.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Arguments

|                   |                                                                                      |
|-------------------|--------------------------------------------------------------------------------------|
| <i>w_windowId</i> | The ID of the window, session window, or dockable window whose name you want to set. |
| <i>t_label</i>    | Text to be displayed in the window manager frame.                                    |

#### Value Returned

|            |                               |
|------------|-------------------------------|
| <i>t</i>   | The text is displayed.        |
| <i>nil</i> | <i>w_windowId</i> is invalid. |

#### ***Related Topics***

[Window Functions](#)

[hiSetWindowName](#)

## hiSetIconName

```
hiSetIconName (
 w_windowId
 t_iconName
)
=> t / nil
```

### Description

Sets the name that is displayed on a window's icon when the window is minimized. A session window, when minimized, displays the icon name of the window that is currently active within it; if the window does not have an icon name set, then the session window's icon name is displayed.

This function is not available in Concept SKILL. Additionally, the function has no effect for KDE or GNOME version 3. For more information, see [Design Entry HDL–SKILL Routines](#).

### Arguments

|                   |                                                                                                                                                      |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | The ID of the window, session window, or dockable window.                                                                                            |
| <i>t_iconName</i> | The name that is to be displayed on the window's icon. Only the first six or seven characters of the icon name are visible because of the icon size. |

### Value Returned

|            |                               |
|------------|-------------------------------|
| <i>t</i>   | The icon name is set.         |
| <i>nil</i> | <i>w_windowId</i> is invalid. |

### Related Topics

[Window Functions](#)

[hiGetIconName](#)

[hiSetWindowIcon](#)

## hiSetWindowIcon

```
hiSetWindowIcon(
 w_windowId
 g_icon
)
=> t / nil
```

### Description

Sets the icon of a window, which is displayed when the window is minimized. A session window, when minimized, displays the icon of the window that is currently active within it; if the window does not have an icon set, then the session window's icon is displayed.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                   |                                                           |
|-------------------|-----------------------------------------------------------|
| <i>w_windowId</i> | The ID of the window, session window, or dockable window. |
| <i>g_icon</i>     | A list representing the icon.                             |

### Value Returned

|            |                                                    |
|------------|----------------------------------------------------|
| <i>t</i>   | The icon is set.                                   |
| <i>nil</i> | <i>w_windowId</i> is invalid or there is an error. |

### Related Topics

[Window Functions](#)

[hiGetIconName](#)

[hiSetIconName](#)

## hiGetWidgetType

```
hiGetWidgetType(
 w_windowId
)
=> t_widgetType / nil
```

### Description

Returns the widget type of a window. Valid widget types are `session`, `graphics`, `text`, `hypertext`, `html`, `encap`, `form`, and `none`.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Window ID of the window. |
|-------------------|--------------------------|

### Value Returned

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <i>t_widgetType</i> | The widget type of the window specified by <i>w_windowId</i> . |
| <i>nil</i>          | <i>w_windowId</i> is invalid.                                  |

### ***Related Topics***

[Window Functions](#)

[hlsWidgetType](#)

## hiIsWidgetType

```
hiIsWidgetType(
 t_widgetType
)
=> t / nil
```

### Description

Determines if a widget type is valid or not.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Argument

|                     |                                         |
|---------------------|-----------------------------------------|
| <i>t_widgetType</i> | The widget type, specified as a string. |
|---------------------|-----------------------------------------|

### Value Returned

|            |                                                                                                                       |
|------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The specified string is a valid widget type: graphics, text, hypertext, html, encap, browser, form, session, or none. |
| <i>nil</i> | <i>w_widgetType</i> is invalid. Also returns <i>nil</i> if <i>w_widgetType</i> is <i>ciw</i> .                        |

### Related Topics

[Window Functions](#)

[hiIsWidgetType](#)

## hiSetCurrentWindow

```
hiSetCurrentWindow(
 w_windowId
)
=> t / nil
```

### Description

Sets the window you specify as the current window.

Session windows and dockable windows can never become the current window. Only windows contained in the main canvas area of a session window can become the current window. If *w\_windowId* is a session window, the last active window contained in the session window becomes the current window. Similarly, if *w\_windowId* is a dockable window, the last active window in the session window that contains the dockable window becomes the current window.

The Command Interpreter Window can also never become the current window. Also, any window with the property `neverCurrentWindow` set to `t` (for example `window->neverCurrentWindow = t`) cannot become the current window.

### Argument

*w\_windowId*

Window you want to make current. The following windows can never be the current window: session windows; dockable windows; the Command Interpreter Window; any window with the property `neverCurrentWindow` set to `t`.

If you have a command for a specific window, make it the current window before calling the `enter` function by using the *windowId* argument.

### Value Returned

*t*

The window is made current.

*nil*

*w\_windowId* is invalid, is the Command Interpreter Window, or has the property `neverCurrentWindow` set to `t`.

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Window Functions](#)

[hiGetCurrentWindow](#)

[hiRegCurWindowTrigger](#)

[hiUnregCurWindowTrigger](#)



## hiRegCurWindowTrigger

```
hiRegCurWindowTrigger(
 S_functionName
)
=> t / nil
```

### Description

Registers the SKILL function to be called whenever the current window changes. The SKILL function is called with one argument: an integer representing the window number of the new current window. If no window is current (for example, if all windows except the CIW are closed or if any open windows have the property `neverCurrentWindow` set to `t`), then the argument is 0.

You can register more than one SKILL function. The functions are called in the order in which they are registered.

### Argument

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <i>S_functionName</i> | A string or symbol representing the function name that you want to register. |
|-----------------------|------------------------------------------------------------------------------|

### Values Returned

|                  |                                                                     |
|------------------|---------------------------------------------------------------------|
| <code>t</code>   | The function is registered.                                         |
| <code>nil</code> | The function is already registered or the function name is invalid. |

### ***Related Topics***

[Window Functions](#)

[hiUnregCurWindowTrigger](#)

[hiGetCurrentWindow](#)

[hiSetCurrentWindow](#)

## hiUnregCurWindowTrigger

```
hiUnregCurWindowTrigger(
 S_functionName
)
=> t / nil
```

### Description

Unregisters a SKILL function that was registered to be called whenever the current window changes.

### Argument

|                       |                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------|
| <i>S_functionName</i> | A string or symbol representing the name of the function that you want to unregister. |
|-----------------------|---------------------------------------------------------------------------------------|

### Values Returned

|     |                                            |
|-----|--------------------------------------------|
| t   | The function is unregistered.              |
| nil | The function is not a registered function. |

### ***Related Topics***

[Window Functions](#)

[hiRegCurWindowTrigger](#)

[hiGetCurrentWindow](#)

[hiSetCurrentWindow](#)

## hiGetMaxScreenCoords

```
hiGetMaxScreenCoords()
=> l_coord_pair
```

### Description

Retrieves the screen coordinates of the upper-right corner of the display screen. The lower-left corner of the screen is assumed to be 0:0. This is the maximum size that you can specify for a window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

None.

### Value Returned

|                     |                                                                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_coord_pair</i> | The screen coordinates of the upper-right corner of the display screen. These coordinates take into account the window manager banner so that a window including a window manager banner can be located at the upper-right corner of the screen. |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Examples

Returns the maximum window initial size. This is the maximum size that can be used to create a new window.

```
hiGetMaxScreenCoords()
```

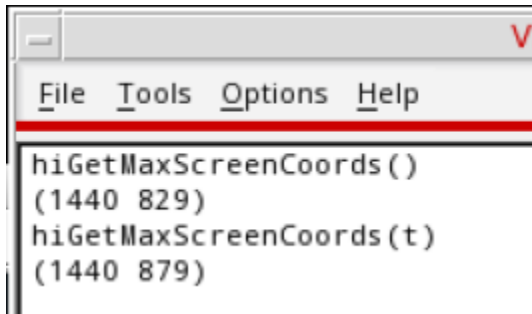
Returns the maximum size a window can grow to fill all screens of the display.

## Cadence User Interface SKILL Reference

### Window Functions

---

`hiGetMaxScreenCoords (t)`



### ***Related Topics***

[Window Functions](#)

## hiGetWMOffsets

```
hiGetWMOffsets()
=> l_offsets
```

### Description

Returns a list of offsets (dimensions) of the window manager border placed around each window.

This function can be used to specify arguments to various window functions, like `hiGetAbsWindowScreenBBox`, when you want to take the window manager border into account.

### Arguments

None.

### Value Returned

*l\_offsets*

A four-element list of window manager border dimensions, in the following order:

*x\_xOffset*: Width of the left window border in pixels.

*x\_yOffset*: Height of the top window border in pixels.

*x\_widthOffset*: Number of extra pixels added to the width of a window due to the window manager border (two times *x\_xOffset* in most cases).

*x\_heightOffset*: Number of extra pixels added to the height of a window due to the window manager border.

### Examples

Returns the dimensions of the window manager border.

```
hiGetWMOffsets()
=> (6 23 12 29)
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

Window Functions

## hiRemoveFixedMenu

```
hiRemoveFixedMenu(
 [w_window]
 [r_fixedMenu]
)
=> t / nil
```

### Description

Removes a fixed menu from a window. If you do not specify a menu, all fixed menus are removed. To delete a fixed menu, first use `hiRemoveFixedMenu` to remove the fixed menu from the window and then use [hiDeleteMenu](#) to destroy the fixed menu.

Fixed menus are no longer supported in windows. Use the new toolbar functions to create toolbars instead for windows. Existing fixed menus that have been added to windows are no longer displayed. Replace these menus with toolbars.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                    |                                                                                                                                                                                                     |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i>    | Window from which the fixed menu is to be removed. It cannot be a session window or a dockable window. If you do not specify <i>w_window</i> , the current window is used.                          |
| <i>r_fixedMenu</i> | The fixed menu to remove. If you do not specify a menu, all fixed menus are removed from the window.<br><br><i>r_fixedMenu</i> is returned by a call to <a href="#">hiCreateVerticalFixedMenu</a> . |

### Value Returned

|     |                                              |
|-----|----------------------------------------------|
| t   | The fixed menu is removed.                   |
| nil | No fixed menu is associated with the window. |

### Examples

```
hiRemoveFixedMenu(window(2))
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Window Functions](#)

[hiDeleteMenu](#)

[hiCreateVerticalFixedMenu](#)



## hiMoveFixedMenu

```
hiMoveFixedMenu(
 [?window w_window]
 [?menuSide s_menuSide]
 [?fixedMenu g_fixedMenu]
)
=> t / nil
```

### Description

Moves the fixed menu to the opposite side of the window.

**Note:** Fixed menus are no longer supported in windows. Use the new toolbar functions to create toolbars instead for windows. Existing fixed menus that have been added to windows are no longer get displayed. Replace these menus with toolbars.

### Arguments

- |                                     |                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>?window w_window</code>       | Window with the fixed menu to be moved. If you do not specify <code>w_window</code> , the current window is used.                                                                                                                                                                                                                                      |
| <code>?menuSide s_menuSide</code>   | Side of the window where the fixed menu is to appear. This value must be either <code>'left</code> , <code>'right</code> or <code>'top</code> . If you do not specify a side, the menu is moved to the opposite of the current side. (If the fixed menu is currently on the top, it is moved to the left because fixed menus cannot be on the bottom.) |
| <code>?fixedMenu g_fixedMenu</code> | The fixed menu to move, specified as a symbol. If you don't specify a fixed menu, moves all fixed menus.                                                                                                                                                                                                                                               |

### Value Returned

- |                  |                                               |
|------------------|-----------------------------------------------|
| <code>t</code>   | The fixed menu is moved.                      |
| <code>nil</code> | No fixed menu is associated with this window. |

### Examples

```
hiMoveFixedMenu(?menuSide 'right)
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Window Functions](#)

## hiGetWindowFixedMenu

```
hiGetWindowFixedMenu(
 [w_window]
)
=> r_fixMenu / nil
```

### Description

Identifies the fixed menu on a window.

**Note:** Fixed menus are no longer supported in windows. Use the new toolbar functions to create toolbars instead for windows. Existing fixed menus that have been added to windows are no longer displayed. Replace these menus with toolbars.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

### Argument

|                 |                                                                                                                  |
|-----------------|------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | Window with the fixed menu to be identified. If you do not specify <i>w_window</i> , the current window is used. |
|-----------------|------------------------------------------------------------------------------------------------------------------|

### Value Returned

|                  |                                       |
|------------------|---------------------------------------|
| <i>r_fixMenu</i> | The fixed menu ID.                    |
| <i>nil</i>       | There is no fixed menu on the window. |

### Examples

```
menu = hiGetWindowFixedMenu(myWin)
```

### Related Topics

[Window Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiSetCursor

```
hiSetCursor(
 w_windowId
 x_cursor
)
=> t / nil
```

### Description

Sets the cursor you specify for the window.

### Arguments

|                   |                                                                                                                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | The window ID.                                                                                                                                                                             |
| <i>x_cursor</i>   | The integer ID associated with the cursor you want to set. The following table lists the constants that are legal values for this argument, as well as their equivalent X Windows cursors: |

| <b>x_cursor Constant</b>                  | <b>Description</b>                                                 | <b>X Windows Equivalent</b>    |
|-------------------------------------------|--------------------------------------------------------------------|--------------------------------|
| <code>hicArrow</code>                     | The default pointer cursor                                         | <code>XC_left_ptr</code>       |
| <code>hicArrowPadlock</code>              | A pointer with a lock                                              |                                |
| <code>hicCreateRuler</code>               | A pointer with a ruler                                             | –                              |
| <code>hicCreateRulerDi-<br/>stance</code> | A two-sided arrow                                                  | –                              |
| <code>hicCreateRulerEd-<br/>ge</code>     | A two-sided arrow between edges                                    | –                              |
| <code>hicCross</code>                     | A multi-lined cross                                                | <code>XC_cross</code>          |
| <code>hicFastEdit</code>                  | A pencil with a via pointing to the top-left corner of the window  |                                |
| <code>hicFastEditPadlo-<br/>ck</code>     | A pencil with a lock pointing to the top-left corner of the window |                                |
| <code>hicHand</code>                      | A hand with index finger extended                                  | <code>XC_hand2</code>          |
| <code>hicHelp</code>                      | A question mark with an arrow                                      | <code>XC_question_arrow</code> |
| <code>hicIbeam</code>                     | An "I-beam" text insertion cursor                                  | <code>XC_xterm</code>          |

## Cadence User Interface SKILL Reference

### Window Functions

---

| <b>x_cursor Constant</b>       | <b>Description</b>                                               | <b>X Windows Equivalent</b>       |
|--------------------------------|------------------------------------------------------------------|-----------------------------------|
| <code>hicNo</code>             | A skull and crossbones                                           | <code>XC_pirate</code>            |
| <code>hicSizeAll</code>        | Four connected arrows pointing up/down/left/right                | <code>XC_fleur</code>             |
| <code>hicSizeNESW</code>       | Two connected arrows pointing top-right and bottom-left          | —                                 |
| <code>hicSizeNS</code>         | Two arrows pointing top and bottom                               | <code>XC_sb_v_double_arrow</code> |
| <code>hicSizeNWSE</code>       | Two connected arrows pointing top-left and bottom-right          | —                                 |
| <code>hicSizeWE</code>         | Two arrows pointing left and right                               | <code>XC_sb_h_double_arrow</code> |
| <code>hicUpArrow</code>        | An arrow pointing straight up                                    | <code>XC_sb_up_arrow</code>       |
| <code>hicWait</code>           | An analog wristwatch                                             | <code>XC_watch</code>             |
| <code>hicArrowHourglass</code> | An hourglass shape                                               | —                                 |
| <code>hicHourglass</code>      | As above                                                         | —                                 |
| <code>hicNoCursor</code>       | The default cursor for the widget (typically the pointer cursor) | —                                 |
| <code>hicInvertedArrow</code>  | A pointer cursor with white and black inverted                   | —                                 |
| <code>hicTumblingDice1</code>  | A box with a dot in the upper left corner                        | —                                 |
| <code>hicTumblingDice2</code>  | A box with a dot in the upper right corner                       | —                                 |
| <code>hicTumblingDice3</code>  | A box with a dot in the lower left corner                        | —                                 |

## Cadence User Interface SKILL Reference

### Window Functions

---

| <b>x_cursor Constant</b>                                                    | <b>Description</b>                                                                                                                                                                                                                                            | <b>X Windows Equivalent</b> |
|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| <code>hicTumblingDice4</code>                                               | A box with a dot in the lower right corner.<br><br>The above tumbling dice cursors are used, in turn, to indicate that DFII is busy. They are displayed when <code>hiGetUserAbort()</code> or <code>voGetUserAbort()</code> is called during long operations. | —                           |
| <code>hicBusyRead</code>                                                    | An open book (used for SKILL context loading)                                                                                                                                                                                                                 | —                           |
| <code>hicHourglass1</code>                                                  | An hourglass just turned over                                                                                                                                                                                                                                 | —                           |
| <code>hicHourglass2</code>                                                  | An hourglass with some sand in the bottom                                                                                                                                                                                                                     | —                           |
| <code>hicHourglass3</code>                                                  | An hourglass with more sand in the bottom                                                                                                                                                                                                                     | —                           |
| <code>hicHourglass4</code>                                                  | An hourglass with most of the sand in the bottom                                                                                                                                                                                                              | —                           |
| <code>hicHourglass5</code>                                                  | An hourglass with all the sand in the bottom                                                                                                                                                                                                                  | —                           |
| <code>hicHourglass6</code>                                                  | An hourglass on its side<br><br>The numbered hourglass cursors are displayed during SKILL garbage collection.                                                                                                                                                 | —                           |
| <code>hicTCross</code>                                                      | A single line cross                                                                                                                                                                                                                                           | <code>XC_tcross</code>      |
| The following cursor definitions specifically use Qt defined cursor shapes. |                                                                                                                                                                                                                                                               |                             |
| <code>hicQtArrow</code>                                                     | Same as <code>hicArrow</code>                                                                                                                                                                                                                                 | <code>XC_left_ptr</code>    |
| <code>hicQtUpArrow</code>                                                   | An arrow pointing up                                                                                                                                                                                                                                          | <code>XC_center_ptr</code>  |
| <code>hicQtCross</code>                                                     | A single-line crosshair                                                                                                                                                                                                                                       | <code>XC_crosshair</code>   |
| <code>hicQtIBeam</code>                                                     | An I-Beam text cursor                                                                                                                                                                                                                                         | <code>XC_xterm</code>       |
| <code>hicQtWait</code>                                                      | An analog wristwatch                                                                                                                                                                                                                                          | <code>XC_watch</code>       |
| <code>hicQtBusy</code>                                                      | An arrow with an hourglass                                                                                                                                                                                                                                    | —                           |

## Cadence User Interface SKILL Reference

### Window Functions

---

| <b>x_cursor Constant</b>                                                                                                                                                                | <b>Description</b>                                                | <b>X Windows Equivalent</b> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|-----------------------------|
| <code>hicQtForbidden</code>                                                                                                                                                             | A circle with a diagonal line                                     | —                           |
| <code>hicQtPointingHand</code>                                                                                                                                                          | A hand with index finger extended                                 | <code>XC_hand2</code>       |
| <code>hicQtWhatsThis</code>                                                                                                                                                             | An arrow with a question mark                                     | —                           |
| <code>hicQtDragCopy</code>                                                                                                                                                              | A closed hand with a plus sign                                    |                             |
| <code>hicQtDragLink</code>                                                                                                                                                              | A closed hand with a link icon (looks like a USB connector cable) |                             |
| <code>hicQtDragMove</code>                                                                                                                                                              | A closed hand with an arrow                                       |                             |
| <code>hicQtOpenHand</code>                                                                                                                                                              | A hand with fingers extended                                      | —                           |
| <code>hicQtSizeAll</code>                                                                                                                                                               | Four connected arrows pointing up/down/left/right                 | <code>XC_fleur</code>       |
| <code>hicQtSizeBDiag</code>                                                                                                                                                             | A solid diagonal arrow pointing down-left and up-right            | —                           |
| <code>hicQtSizeFDiag</code>                                                                                                                                                             | A solid diagonal arrow pointing down-right and up-left            | —                           |
| <code>hicQtSizeHor</code>                                                                                                                                                               | A solid horizontal arrow pointing left and right                  | —                           |
| <code>hicQtSizeVer</code>                                                                                                                                                               | A solid vertical arrow pointing up and down                       | —                           |
| <code>hicQtSplitH</code>                                                                                                                                                                | Two vertical lines with left/right arrows                         | —                           |
| <code>hicQtSplitV</code>                                                                                                                                                                | Two horizontal lines with up/down arrows                          | —                           |
| <code>hiQtClosedHand</code>                                                                                                                                                             | A hand with fingers curled up                                     | —                           |
| The following cursors are not part of the standard cursor set but can still be desirable to depict certain operations, including enterfunctions (see <a href="#">hiSetEFunCursor</a> ). |                                                                   |                             |
| <code>hicAlign</code>                                                                                                                                                                   | A cursor that represents alignment                                |                             |
| <code>hicAreaSelect</code>                                                                                                                                                              | A cursor for area selection                                       |                             |
| <code>hicArrowA</code>                                                                                                                                                                  | An arrow with an "A" for a label or other text                    | —                           |
| <code>hicArrowArc</code>                                                                                                                                                                | An arrow with an arc                                              | —                           |

## Cadence User Interface SKILL Reference

### Window Functions

---

| <b>x_cursor Constant</b>         | <b>Description</b>                                            | <b>X Windows Equivalent</b> |
|----------------------------------|---------------------------------------------------------------|-----------------------------|
| <code>hicArrowBox</code>         | An arrow with a box that represents moving or stretching      | —                           |
| <code>hicArrowCircle</code>      | An arrow with a circle                                        | —                           |
| <code>hicArrowCircleI</code>     | An arrow with an "i" in a circle                              | —                           |
| <code>hicArrowCopy</code>        | An arrow with a glyph that represents copying.                | —                           |
| <code>hicArrowDot</code>         | An arrow with a dot that represents a solder dot              | —                           |
| <code>hicArrowEllipse</code>     | An arrow with an ellipse                                      | —                           |
| <code>hicArrowEllipsis</code>    | An arrow with three dots for the default enterfunction cursor | —                           |
| <code>hicArrowIBeam</code>       | An arrow with an I-Beam for direct text edit                  | —                           |
| <code>hicArrowInstance</code>    | An arrow with a glyph for "create instance"                   | —                           |
| <code>hicArrowMagnifier</code>   | An arrow with a magnifier for the Zoom command                | —                           |
| <code>hicArrowMove</code>        | An arrow with a four-way arrow                                | —                           |
| <code>hicArrowPin</code>         | An arrow with a "pin" glyph                                   | —                           |
| <code>hicArrowPolygon</code>     | An arrow with a polygon                                       | —                           |
| <code>hicArrowRectangle</code>   | An arrow with a rectangle                                     | —                           |
| <code>hicArrowRotate</code>      | An arrow with a "rotation" glyph                              | —                           |
| <code>hicArrowSlashCircle</code> | An arrow with a "forbidden" glyph                             | —                           |
| <code>hicArrowT</code>           | An arrow with a "T" for direct text edit                      | —                           |
| <code>hicArrowText</code>        | An arrow with a glyph used for "create label"                 | —                           |
| <code>hicArrowTextBox</code>     | An arrow with text in a box                                   | —                           |
| <code>hicArrowTextWire</code>    | An arrow with a glyph used for "create wire name"             | —                           |



## Cadence User Interface SKILL Reference

### Window Functions

---

| <b>x_cursor Constant</b>       | <b>Description</b>                                                                                               | <b>X Windows Equivalent</b> |
|--------------------------------|------------------------------------------------------------------------------------------------------------------|-----------------------------|
| <code>hicArrowWire</code>      | An arrow with a zig-zag line that represents drawing a wire or a path                                            | —                           |
| <code>hicArrowX</code>         | An arrow with a X for the Delete command                                                                         | —                           |
| <code>hicAttachDetach</code>   | A cursor for attach and detach operations                                                                        |                             |
| <code>hicChaining</code>       | A cursor for chaining operations                                                                                 |                             |
| <code>hicChop</code>           | A cursor for chopping a shape                                                                                    |                             |
| <code>hicCreateRow</code>      | A cursor for row creation                                                                                        |                             |
| <code>hicCreateVia</code>      | A cursor for via creation                                                                                        |                             |
| <code>hicDescend</code>        | A down arrow for descending into an instance                                                                     |                             |
| <code>hicMagnifier</code>      | A magnifying glass that can be used with the Zoom command (the hotspot being the center of the magnifying glass) | —                           |
| <code>hicMagnifierMinus</code> | A magnifying glass that can be used with the Zoom-Out command (the hotspot being the center of the "-")          | —                           |
| <code>hicMagnifierPlus</code>  | A magnifying glass that can be used with the Zoom-In command (the hotspot being the center of the "+")           | —                           |
| <code>hicMerge</code>          | A cursor for merging shapes                                                                                      |                             |
| <code>hicMove</code>           | A cursor that represents moving.                                                                                 | —                           |
| <code>hicMPP</code>            | An arrow with "MPP" attached                                                                                     |                             |
| <code>hicProbe</code>          | A "probe" cursor                                                                                                 | —                           |
| <code>hicReshape</code>        | A cursor for modifying shapes                                                                                    |                             |
| <code>hicSpinCircle</code>     | A cursor to indicate that a command is running (the hotspot being the center of the circle)                      | —                           |
| <code>hicStretchE</code>       | A cursor that represents stretching East.                                                                        | —                           |

## Cadence User Interface SKILL Reference

### Window Functions

---

| <b>x_cursor Constant</b>     | <b>Description</b>                                      | <b>X Windows Equivalent</b> |
|------------------------------|---------------------------------------------------------|-----------------------------|
| <code>hicStretchN</code>     | A cursor that represents stretching North.              | —                           |
| <code>hicStretchNE</code>    | A cursor that represents stretching North-East.         | —                           |
| <code>hicStretchNW</code>    | A cursor that represents stretching North-West.         | —                           |
| <code>hicStretchS</code>     | A cursor that represents stretching South.              | —                           |
| <code>hicStretchSE</code>    | A cursor that represents stretching South-East.         | —                           |
| <code>hicStretchSW</code>    | A cursor that represents stretching South-West.         | —                           |
| <code>hicStretchW</code>     | A cursor that represents stretching West.               | —                           |
| <code>hicSlot</code>         | A cursor for slots                                      |                             |
| <code>hicSplit</code>        | A cursor for splitting a shape                          |                             |
| <code>hicUnmarkNet</code>    | A cursor for removing the mark from a net               |                             |
| <code>hicIBeam</code>        | A narrow I-Beam (as compared to <code>hiclbeam</code> ) |                             |
| <code>hicRotatedIBeam</code> | An I-Beam for vertical text.                            |                             |
| <code>hicMoveSmall</code>    | A smaller 4-way arrow than <code>hicMove</code>         |                             |
| <code>hicArrowFilter</code>  | An arrow with a funnel shape                            |                             |

#### Value Returned

|                  |                         |
|------------------|-------------------------|
| <code>t</code>   | The cursor was set.     |
| <code>nil</code> | The cursor was not set. |

#### ***Related Topics***

#### Window Functions

## Cadence User Interface SKILL Reference

### Window Functions

---

hiGetCursorName

## hiGetCursorName

```
hiGetCursorName(
 x_cursorID
)
=> t_cursorName
```

### Description

Returns the name of the cursor associated with the specified cursor ID.

### Arguments

|                   |                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>x_cursorID</i> | An integer specifying the ID associated with the given cursor name.<br><br>For example, "hicArrow" or "hicIbeam". |
|-------------------|-------------------------------------------------------------------------------------------------------------------|

### Value Returned

|                     |                                                                                                                                |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <i>t_cursorName</i> | Name of a cursor.<br><br>For example, "arrow" or "ibeam". The value "unknown" indicates that the <i>x_cursorID</i> is invalid. |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------|

### Examples

Returns the name of the cursor *ibeam* associated with *hicIbeam*.

```
hiGetCursorName(hicIbeam)
=> "ibeam"
```

Returns the name of the cursor *arrow* associated with *hicArrow*.

```
hiGetCursorName(hicArrow)
=> "arrow"
```

Returns the name of the cursor *unknown* associated with *-1*.

```
hiGetCursorName(-1)
=> "unknown"
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Window Functions](#)

## hiSetShadowMode

```
hiSetShadowMode(
 g_enabled
)
=> t / nil
```

### Description

Controls the display of unselected objects in a graphics window. When you enable shadow mode with this function, all unselected objects are dimmed and all selected objects are displayed at full brightness in their original layer color instead of the selection color. The shadow mode setting applies to all graphics in all windows of the application, not just to the current window.

### Argument

|                  |                                                                                      |
|------------------|--------------------------------------------------------------------------------------|
| <i>g_enabled</i> | Enables shadow mode for all unselected objects, which means their display is dimmed. |
|------------------|--------------------------------------------------------------------------------------|

### Value Returned

|     |                          |
|-----|--------------------------|
| t   | Shadow mode was enabled. |
| nil | There was an error.      |

### ***Related Topics***

[Window Functions](#)

## hiRemovePromptLine

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

### Description

Removes the prompt line from the graphics window you specify or, if you do not specify a window, from the current window.

### Argument

|                 |                                                                                                                                                                     |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window ID of the window from which you want to remove the prompt line. If you do not specify this argument, the prompt line is removed from the current window. |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|            |                              |
|------------|------------------------------|
| <i>t</i>   | The prompt line was removed. |
| <i>nil</i> | There was an error.          |

### ***Related Topics***

[Window Functions](#)

## hiSetEFunCursor

```
hiSetEFunCursor(
 w_window
 x_cursor
)
=> t / nil
```

### Description

Sets the cursor for an active enterfunction.

### Arguments

|                 |                                                                                |
|-----------------|--------------------------------------------------------------------------------|
| <i>w_window</i> | The window ID of the window for which you want to set an active enterfunction. |
| <i>x_cursor</i> | The cursor to be set for the active enterfunction.                             |

### Value Returned

|            |                                                  |
|------------|--------------------------------------------------|
| <i>t</i>   | Enterfunction cursor successfully set.           |
| <i>nil</i> | There was no active enterfunction on the window. |

### ***Related Topics***

Window Functions

hiSetCursor



## hiSetEFDrawThruDelta

```
hiSetEFDrawThruDelta(
 x_EFdelta
 [x_button]
)
=> t
```

### Description

Sets the Enter Function draw-thru delta.

**Note:** An EnterFunction draw-thru delta is measured as being the minimum distance that the mouse must move, with the mouse button held down, to be interpreted as a draw-thru (if the button is released before the double-click time has expired and an enterfunction is currently active in that window). This can be set for each of the three mouse buttons.

### Arguments

|                     |                                                                                                                                                                                                                                                            |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_EFdelta</i>    | The EnterFunction draw-thru delta setting.<br><br>The value of <i>x_EFdelta</i> cannot be lower than the regular draw-thru delta. If the value is lower, it sets to the regular draw-thru delta                                                            |
| [ <i>x_button</i> ] | The mouse button that the EnterFunction delta is to be set for.<br><br>If the argument <i>x_button</i> is not specified, it is assumed to be for button 1. If a button value other than 1, 2 or 3 is specified no action is taken and a warning is issued. |

### Value Returned

|          |                                                                                    |
|----------|------------------------------------------------------------------------------------|
| <i>t</i> | EnterFunction draw-thru delta successfully set. Function always returns <i>t</i> . |
|----------|------------------------------------------------------------------------------------|

### Examples

```
hiSetEFDrawThruDelta(x_EFdelta [x_button])
=> t
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

Window Functions

hiSetEFDDrawThruDelta

## **setCurrentWindow**

The `setCurrentWindow` function is an alias for the `hiSetCurrentWindow` function.

## Cadence User Interface SKILL Reference

### Window Functions

---

## window

```
window(
 x_windowNumber
)
=> w_windowId / nil
```

### Description

Given a window number, gets the window ID.

This function cannot get the *windowID* of session windows. It does, however, get the *windowID* of windows contained in session windows.

The window number is the number displayed in the lower-left corner of the window. The Command Interpreter Window is window 1. As windows are created (either interactively or with `hiCreateWindow` or `hiOpenWindow`), they are given higher window numbers. Windows that are in session windows have their number in parentheses—for example, if the session window displays 3(5), the window number of the window contained in the session window is 5 while the session window number is 3.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Argument

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_windowNumber</i> | The window number, which is displayed in the lower-left corner of the window.<br><br><i>x_windowNumber</i> cannot be a session window number. |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|                   |                                                                         |
|-------------------|-------------------------------------------------------------------------|
| <i>w_windowId</i> | The <i>w_windowId</i> associated with the given <i>x_windowNumber</i> . |
| <i>nil</i>        | There is no window associated with the specified window number.         |

### Examples

```
hiSetCurrentWindow(window (5))
=> t
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

Window Functions

swindow

windowp

## Cadence User Interface SKILL Reference

### Window Functions

---

#### **swindow**

```
swindow(
 x_windowNumber
)
=> w_sessionWindowID / nil
```

#### **Description**

Gets the window ID of a session window.

#### **Argument**

|                       |                                                                                                                                                                 |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_windowNumber</i> | The session window number, displayed in the lower-left corner of the session window. For example, if the window displays 3(12), the session window number is 3. |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### **Value Returned**

|                          |                                                                      |
|--------------------------|----------------------------------------------------------------------|
| <i>w_sessionWindowID</i> | The session window ID.                                               |
| <i>nil</i>               | There is no session window that matches the window number specified. |

#### **Example**

```
swindow(3)
```

#### ***Related Topics***

[Window Functions](#)

[hilsWindowSpecifier](#)

[windowp](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### **swindowp**

```
swindowp(
 g_object
)
=> t / nil
```

#### **Description**

Checks whether the specified object is a valid session window.

#### **Argument**

|                 |                                    |
|-----------------|------------------------------------|
| <i>g_object</i> | The object that you want to check. |
|-----------------|------------------------------------|

#### **Value Returned**

|     |                                                |
|-----|------------------------------------------------|
| t   | <i>g_object</i> is a valid session window.     |
| nil | <i>g_object</i> is not a valid session window. |

#### **Examples**

```
swindowp(myWindowID)
```

#### ***Related Topics***

[Window Functions](#)

[windowp](#)

[hlsWindowSpecifier](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### windowp

```
windowp(
 g_object
)
=> t / nil
```

#### Description

Checks whether the specified object is a valid window.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

#### Argument

|                 |                           |
|-----------------|---------------------------|
| <i>g_object</i> | Object you want to check. |
|-----------------|---------------------------|

#### Values Returned

|     |                                        |
|-----|----------------------------------------|
| t   | <i>g_object</i> is a valid window.     |
| nil | <i>g_object</i> is not a valid window. |

#### Example

```
windowp(myWindowID)
```

#### ***Related Topics***

[Window Functions](#)

[Viewing Functions](#)

[hlsWindowSpecifier](#)

[windowp](#)



## Cadence User Interface SKILL Reference

### Window Functions

---

#### **wtypep**

```
wtypep(
 g_object
)
=> t / nil
```

#### **Description**

A predicate function that returns *t* if *g\_object* is of type window, and *nil* otherwise.

#### **Argument**

|                 |                         |
|-----------------|-------------------------|
| <i>g_object</i> | Argument to be checked. |
|-----------------|-------------------------|

#### **Value Returned**

|            |                                     |
|------------|-------------------------------------|
| <i>t</i>   | The argument is of type window.     |
| <i>nil</i> | The argument is not of type window. |

#### ***Related Topics***

[Window Functions](#)

[windowp](#)

[Viewing Functions](#)

[hlsWindowSpecifier](#)

## Viewing Functions

This topic provides a list of Cadence® SKILL functions associated with zooming, panning, and redrawing to ensure that all basic editing functions are done in an identical manner across all applications. Application functions created from this common core behave similarly across products. Most of these functions require specific window interface routines to be registered on the windows.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

**Note:** These functions can only be called with windows that are not session windows or dockable windows.

- [hiZoomIn](#)
- [hiZoomOut](#)
- [hiZoomRelativeScale](#)
- [hiZoomWindowAtPoint](#)
- [hiZoomWindowAtMouse](#)
- [hiZoomInAtMouse](#)
- [hiZoomOutAtMouse](#)
- [hiZoomExpandAtPoint](#)
- [hiZoomExpandAtMouse](#)
- [hiGetViewBBox](#)
- [hiPan](#)
- [hiVectorPan](#)
- [hiDeltaPan](#)
- [hiAbsolutePan](#)
- [hiDynamicPanAbsolute](#)
- [hiDynamicPanRelative](#)
- [hiDynamicPanInteractive](#)
- [hiDynamicPanGrabbing](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiZoomIn

```
hiZoomIn(
 [w_window]
 [l_bBox]
)
=> t / nil
```

#### Description

Zooms in to a design.

This function also decreases the portion of the design you view.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

## Cadence User Interface SKILL Reference

### Window Functions

---

#### Arguments

*w\_window*

The window you want to zoom into. You must specify the window if you specify the *l\_bBox* argument. If you do not specify the *l\_bBox* argument, this argument is optional.

For the current window, you can use

`hiGetCurrentWindow()` as the value of *w\_window*.

**Note:** *w\_window* cannot be a session window or a dockable window.

*l\_bBox*

Zoom bounding box specified in user units. If you do not specify this argument, the function prompts you for a rubberbanding box.

After the view rectangle is specified, the command zooms in to fill the window with the contents of the view rectangle.

#### Value Returned

*t*

The magnification is increased.

*nil*

`zoomPanProc` or `uPointToDbuProc` are not registered on the window.

#### Example

```
hiZoomIn(hiGetCurrentWindow() list(0:0 10:10))
```

#### ***Related Topics***

[Viewing Functions](#)

[hiZoomOut](#)

[hiZoomRelativeScale](#)

[hiZoomAbsoluteScale](#)

[hiZoomWindowAtPoint](#)

[hiZoomWindowAtMouse](#)

[hiZoomInAtMouse](#)

## hiZoomOut

```
hiZoomOut (
 [w_window]
 [l_bBox]
)
=> t / nil
```

### Description

Zoom out of a design.

This function also decreases the magnification of the design and, consequently, increases the extent of the design viewed in the window.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                 |                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window ID. If not specified, the current window is used.<br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window.                                                                                                                                                                                                                                           |
| <i>l_bBox</i>   | Bounding box specified in user units. If this is not specified, the function prompts through a rubberbanding box.<br>This rubberbanding box displays the aspect ratio of the window. This means the box always contains what is currently viewable.<br>After the rectangular area is specified, the command zooms out to fit the contents of <i>w_window</i> into the rectangular area. |

### Value Returned

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| <i>t</i>   | The magnification is decreased.                                                |
| <i>nil</i> | <i>zoomPanProc</i> or <i>uPointToDbuProc</i> are not registered on the window. |

### Related Topics

#### [Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiZoomIn

hiZoomRelativeScale

hiZoomAbsoluteScale

hiZoomWindowAtPoint

hiZoomWindowAtMouse

hiZoomInAtMouse

hiZoomOutAtMouse

## hiZoomRelativeScale

```
hiZoomRelativeScale(
 [w_window]
 [n_scale]
)
=> t / nil
```

### Description

Zooms in or out with a scale relative to the current window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

*w\_window*

The window ID. If not specified, the current window is used.

**Note:** *w\_window* cannot be a session window or a dockable window.

*n\_scale*

Scale factor you want to use in zooming. A number greater than 1.0 specifies a zoom-in factor. A number less than 1.0 specifies a zoom-out factor. The center of the original window is the center of the new window. If *n\_scale* is not specified, a scale of 1.0 is used.

### Values Returned

*t*

You can zoom in and out.

*nil*

*zoomPanProc* or *uPointToDbuProc* are not registered on the window.

### Related Topics

[Viewing Functions](#)

[hiZoomIn](#)

[hiZoomOut](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiZoomAbsoluteScale

hiZoomWindowAtPoint

hiZoomWindowAtMouse

hiZoomInAtMouse

hiZoomOutAtMouse

hiRegZoomPanProc

hiUnregZoomPanProc



## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiZoomAbsoluteScale

```
hiZoomAbsoluteScale(
 [w_window]
 [n_scale]
 [g_ignoreEFunBBox]
)
=> t / nil
```

#### Description

Zooms in or out with an absolute scale.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Arguments

|                         |                                                                                                                                                                                                                                                                                               |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i>         | The window ID. If not specified, the current window is used.<br><br><i>w_window</i> cannot be a session window or a dockable window.                                                                                                                                                          |
| <i>n_scale</i>          | Scale factor you want to use for zooming. For example, a scale of 1.0 means the size of one <i>w_window</i> fits the entire representation. A scale of 2.0 means the size of two <i>w_windows</i> fits the entire representation. If <i>n_scale</i> is not specified, a scale of 1.0 is used. |
| <i>g_ignoreEFunBBox</i> | Ignores active enterfunction bounding box.                                                                                                                                                                                                                                                    |

#### Values Returned

|     |                                                                                            |
|-----|--------------------------------------------------------------------------------------------|
| t   | You can zoom in and out.                                                                   |
| nil | <code>zoomPanProc</code> or <code>uPointToDbuProc</code> are not registered on the window. |

#### Related Topics

[Viewing Functions](#)

[hiZoomIn](#)

[hiZoomOut](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiZoomRelativeScale

hiZoomWindowAtPoint

hiZoomWindowAtMouse

hiZoomInAtMouse

hiZoomOutAtMouse

hiRegZoomPanProc

hiUnregZoomPanProc

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiZoomWindowAtPoint

```
hiZoomWindowAtPoint (
 w_window
 f_scale
 l_point
)
=> t / nil
```

#### Description

Zooms in or out of a window, keeping the point you specify at its current location in the window.

This function supports zooming with the scrollwheel on a mouse.

#### Arguments

|                 |                                                                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window you want to zoom in to.<br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window.                                 |
| <i>f_scale</i>  | Scale factor you want to use for zooming. A number greater than 1.0 specifies a zoom-in factor. A number less than 1.0 specifies a zoom-out factor. |
| <i>l_point</i>  | The point which you want to keep in its current location while zooming in or out. Specify <i>l_point</i> in user units.                             |

#### Value Returned

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| <i>t</i>   | Zoomed in or out of the window.                                                |
| <i>nil</i> | <i>zoomPanProc</i> or <i>uPointToDbuProc</i> are not registered on the window. |

#### Example

```
hiZoomWindowAtPoint(window (3) 2.0 list(0.18 0.81))
```

or

```
hiZoomWindowAtPoint(window (3) 0.5 list(0.18 0.81))
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Viewing Functions](#)

[hiZoomWindowAtMouse](#)

[hiZoomInAtMouse](#)

[hiZoomOutAtMouse](#)

[hiZoomIn](#)

[hiZoomOut](#)

[hiZoomRelativeScale](#)

[hiZoomAbsoluteScale](#)

## hiZoomWindowAtMouse

```
hiZoomWindowAtMouse (
 w_window
 f_scale
)
=> t / nil
```

### Description

Zooms in or out of a window, keeping the point beneath the mouse at its current location in the window.

This function supports zooming with the scrollwheel on a mouse.

### Arguments

|                 |                                                                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window you want to zoom in to.<br><i>w_window</i> cannot be a session window or a dockable window.                                              |
| <i>f_scale</i>  | Scale factor you want to use for zooming. A number greater than 1.0 specifies a zoom-in factor. A number less than 1.0 specifies a zoom-out factor. |

### Values Returned

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| <i>t</i>   | Zoomed in or out of the window.                                                |
| <i>nil</i> | <i>zoomPanProc</i> or <i>uPointToDbuProc</i> are not registered on the window. |

### ***Related Topics***

### ***Related Topics***

[Viewing Functions](#)

[hiZoomWindowAtPoint](#)

[hiZoomInAtMouse](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiZoomOutAtMouse

hiZoomIn

hiZoomOut

hiZoomRelativeScale

hiZoomAbsoluteScale

hiRegZoomPanProc

hiUnregZoomPanProc

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiZoomInAtMouse

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

### Description

Zooms into a window by a scale factor of 2, keeping the point beneath the mouse at its current location in the window.

This function supports zooming with the scrollwheel on a mouse.

### Argument

*w\_window*

The window you want to zoom in to. If not specified, the current window is used.

*w\_window* cannot be a session window or a dockable window.

### Value Returned

*t*

Zoomed into the window.

*nil*

`zoomPanProc` or `uPointToDbuProc` are not registered on the window.

### ***Related Topics***

[Viewing Functions](#)

[hiZoomWindowAtPoint](#)

[hiZoomWindowAtMouse](#)

[hiZoomOutAtMouse](#)

[hiZoomIn](#)

[hiZoomOut](#)

[hiZoomRelativeScale](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

hiZoomAbsoluteScale



## hiZoomOutAtMouse

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

### Description

Zooms out of a window by a scale factor of 2, keeping the point beneath the mouse at its current location in the window.

This function supports zooming with the scrollwheel on a mouse.

### Argument

*w\_window*

The window you want to zoom in to. If not specified, the current window is used.

*w\_window* cannot be a session window or a dockable window.

### Value Returned

*t*

Zoomed out of the window.

*nil*

`zoomPanProc` or `uPointToDbuProc` are not registered on the window.

### ***Related Topics***

[Viewing Functions](#)

[hiZoomWindowAtPoint](#)

[hiZoomWindowAtMouse](#)

[hiZoomInAtMouse](#)

[hiZoomIn](#)

[hiZoomOut](#)

[hiZoomRelativeScale](#)

## hiZoomExpandAtPoint

```
hiZoomExpandAtPoint(
 w_window
 f_zoomFactor
 l_point
)
=> t / nil
```

### Description

Zooms out the image by keeping the mirror image point of `l_point` at the same location

### Arguments

|                           |                                                                                           |
|---------------------------|-------------------------------------------------------------------------------------------|
| <code>w_window</code>     | Specifies the window that contains the image to be zoomed out, usually the current window |
| <code>f_zoomFactor</code> | Specifies the zoom factor.                                                                |
| <code>l_point</code>      | Specifies the point with respect to which you want to zoom the image                      |

### Value Returned

|                  |                                         |
|------------------|-----------------------------------------|
| <code>t</code>   | If the image is successfully zoomed out |
| <code>nil</code> | If the image is not zoomed out          |

### Examples

This example can be used as a bindkey definition to zoom out an image by two times by expanding the window in the direction of the mouse relative to the center of the window.

```
hiZoomExpandAtPoint(hiGetCurrentWindow() 0.5 hiGetPoint(hiGetCurrentWindow()))
```

### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiZoomExpandAtMouse

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

#### Description

Zooms out by two in the specified window (or current window) expanding the zoom area in the direction of the mouse relative to the center of the window

This function is used for hard-coded zoom-out during drag operations.

#### Argument

|                 |                                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | Optional. Specifies the window to be zoomed out. If not specified, the function uses the current window. |
|-----------------|----------------------------------------------------------------------------------------------------------|

#### Value Returned

|            |                                                                                           |
|------------|-------------------------------------------------------------------------------------------|
| <i>t</i>   | Zoomed out of the specified (or current) window                                           |
| <i>nil</i> | <code>zoomPanProc</code> or <code>uPointToDbuProc</code> are not registered on the window |

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiGetViewBBox

```
hiGetViewBBox(
 [w_window]
)
=> l_bBox
```

#### Description

Returns the bounding box displayed in a window.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Argument

|                 |                                                                                                                                               |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window ID. If not specified, the current window is used.<br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window. |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

#### Value Returned

|               |                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_bBox</i> | The coordinates of the bounding box. The bounding box is a list of two points in user units describing the lower left and upper right corners of the window's view area. |
| <i>nil</i>    | <code>dbuToUPProc</code> is not registered.                                                                                                                              |

#### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiPan

```
hiPan(
 [w_window]
 [l_point]
)
=> t / nil
```

#### Description

Scrolls a window so that the display is centered on a point.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Arguments

|                 |                                                                                                                                                                     |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window that you want to pan. If not specified, the current window is used.<br><br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window. |
| <i>l_point</i>  | Point you wish to be at the center of the window. If not specified, the user is prompted.                                                                           |

#### Value Returned

|            |                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The display is centered.                                                                              |
| <i>nil</i> | <i>zoomPanProc</i> , <i>dbuToUPProc</i> , or <i>uPointToDbuProc</i> are not registered on the window. |

#### ***Related Topics***

[hiVectorPan](#)

[hiDeltaPan](#)

[hiAbsolutePan](#)

[hiDynamicPanAbsolute](#)

[hiDynamicPanRelative](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiDynamicPanInteractive

hiUnregZoomPanProc

## hiVectorPan

```
hiVectorPan(
 [w_window]
 [l_refPt]
 [l_destPt]
)
=> t / nil
```

### Description

Scrolls the view of the design in any direction by specifying two points in user units.

The design is panned so that *l\_refPt* on the design is displayed on the screen at *l\_destPt*. After the command is invoked and the first reference point is specified, a rubberbanding line connects the first point with the cursor. Click the mouse button at the desired location to specify the destination point. The design then pans to that location.

By placing the same design in two different windows and specifying the reference point in one window and the destination point in the other, the specified window can be panned relative to the area in the second window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                 |                                                                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window that you want to pan. If not specified, the current window is used.<br><br><i>w_window</i> cannot be a session window or a dockable window. |
| <i>l_refPt</i>  | Reference point you want to pan from. If not specified, you are prompted to enter the value.                                                           |
| <i>l_destPt</i> | Destination point you want to pan to. If not specified, you are prompted to enter the value.                                                           |

### Value Returned

|            |                                                                               |
|------------|-------------------------------------------------------------------------------|
| <i>t</i>   | The view can be scrolled as specified.                                        |
| <i>nil</i> | <i>zoomPanProc</i> or <i>uPointToDbuProc</i> is not registered on the window. |

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[hiPan](#)

[hiDeltaPan](#)

[hiAbsolutePan](#)

[hiDynamicPanAbsolute](#)

[hiDynamicPanRelative](#)

[hiDynamicPanInteractive](#)

[hiRegZoomPanProc](#)

[hiUnregZoomPanProc](#)



## hiDeltaPan

```
hiDeltaPan(
 [w_window]
 [n_deltax]
 [n_deltay]
)
=> t / nil
```

### Description

Pans the view of a design in any direction by specifying the displacements in user units.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                 |                                                                                                                                                                     |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window that you want to pan. If not specified, the current window is used.<br><br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window. |
| <i>n_deltax</i> | Number of units for panning the design in <i>x</i> direction. If not specified, the default of 0 is used.                                                           |
| <i>n_deltay</i> | Number of units for panning the design in <i>y</i> direction. If not specified, the default of 0 is used.                                                           |

### Value Returned

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| <i>t</i>   | The view can be panned in any direction.                                       |
| <i>nil</i> | <i>zoomPanProc</i> or <i>uPointToDbuProc</i> are not registered on the window. |

### Related Topics

[hiPan](#)

[hiVectorPan](#)

[hiAbsolutePan](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

hiDynamicPanAbsolute

hiDynamicPanRelative

hiDynamicPanInteractive

hiRegZoomPanProc

hiUnregZoomPanProc

## hiAbsolutePan

```
hiAbsolutePan(
 [w_window]
 [n_%x]
 [n_%y]
)
=> t / nil
```

### Description

Scrolls a window in any direction by specifying the amount to pan as a percentage of the window size.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

|                 |                                                                                                                                                                     |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window that you want to pan. If not specified, the current window is used.<br><br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window. |
| <i>n_%x</i>     | Percentage of panning in the design in the x-direction. If not specified, the default of 0 is used.                                                                 |
| <i>n_%y</i>     | Percentage of panning in the design in the y-direction. If not specified, the default of 0 is used.                                                                 |

### Value Returned

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| <i>t</i>   | The window can be scrolled in any direction.                                   |
| <i>nil</i> | <i>zoomPanProc</i> or <i>uPointToDbuProc</i> are not registered on the window. |

### Related Topics

[hiPan](#)

[hiVectorPan](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiDeltaPan

hiDynamicPanAbsolute

hiDynamicPanRelative

hiDynamicPanInteractive

hiRegZoomPanProc

hiUnregZoomPanProc

## hiDynamicPanAbsolute

```
hiDynamicPanAbsolute(
 [w_window]
 [l_point]
)
=> t / nil
```

### Description

Pans a design to display the specified point at the center of the window. This function is similar to `hiPan`, except that it pans dynamically, that is, it displays the panning movement.

You can either provide a point and window ID as arguments to this function or specify the function without any arguments and then click a point in a window.

### Arguments

|                 |                                                                                                                                                                     |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window that you want to pan. If not specified, the current window is used.<br><br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window. |
| <i>l_point</i>  | The point that you want in the center of the window. Specify <i>l_point</i> in user units.                                                                          |

### Value Returned

|            |                           |
|------------|---------------------------|
| <i>t</i>   | The design can be panned. |
| <i>nil</i> | There was an error.       |

### Related Topics

[hiDynamicPanRelative](#)

[hiDynamicPanInteractive](#)

[hiPan](#)

[hiVectorPan](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiDeltaPan

hiAbsolutePan

hiRegZoomPanProc

hiUnregZoomPanProc

## hiDynamicPanRelative

```
hiDynamicPanRelative(
 [w_window]
 [n_percentX]
 [n_percentY]
 [g_noAnimation { t | nil }]
)
=> t / nil
```

### Description

Pans a design in the direction and distance specified in *n\_percentX* and *n\_percentY*. The distance is a percentage of the window size.

For example, if you want to pan to the right for a distance that is equal to 10% of the window width, you would specify:

```
hiDynamicPanRelative(window(2) 10)
```

The image shifts to the left by 10% of the width of the window, displaying more of the design on the right.

To pan vertically, set the *n\_percentX* argument to 0; to pan horizontally, either set the *n\_percentY* argument to 0 or do not specify it.

### Arguments

|                   |                                                                                                                                                                                                                                                                                                                                              |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i>   | The window that you want to pan. If not specified, the current window is used.<br><br><b>Note:</b> <i>w_window</i> cannot be a session window or a dockable window.                                                                                                                                                                          |
| <i>n_percentX</i> | The distance that you want to pan to the right or left, specified as a percentage of the window's width. To pan to the right, specify <i>n_percentX</i> as a positive number. To pan to the left, specify <i>n_percentX</i> as a negative number.                                                                                            |
| <i>n_percentY</i> | The distance that you want to pan up or down, specified as a percentage of the window's height. To pan down, specify <i>n_percentY</i> as a positive number. To pan up, specify <i>n_percentY</i> as a negative number.<br><br>If you want to use this argument and do not want to use <i>n_percentX</i> , specify 0 for <i>n_percentX</i> . |

## Cadence User Interface SKILL Reference

### Window Functions

---

*g\_noAnimation* Specifies whether to skip the animation affect during dynamic panning. The default is `nil`, which means that the animation affect is not skipped.

#### Value Returned

`t` The design can be panned.  
`nil` There was an error.

#### Example

```
hiDynamicPanRelative(window(2) 10 20)
```

#### ***Related Topics***

[hiDynamicPanAbsolute](#)

[hiDynamicPanInteractive](#)

[hiPan](#)

[hiVectorPan](#)

[hiDeltaPan](#)

[hiAbsolutePan](#)

[hiRegZoomPanProc](#)

[hiUnregZoomPanProc](#)



## hiDynamicPanInteractive

```
hiDynamicPanInteractive(
 [w_window]
 [n_scope]
 [g_noAnimation { t | nil }]
)
=> t / nil
```

### Description

Pans a design continuously when you place the cursor near the edge of a graphics window. You can control the speed at which the design is panned by varying the distance between the cursor and the edge of the window—the closer the cursor is to the edge of the window, the faster the speed.

You can also use an anchor box to pan the design instead of moving the cursor on the window edges. Click the left mouse button on the window to display the anchor box. You can then move the cursor along the edges of the anchor box to pan the design. The distance between the cursor and the anchor box determines the speed.

To exit the panning mode, press the `ESC` key or any other key, or select any command in your graphics application.

If you create your own graphics windows, you need to register a `displayProc` callback for the window for this function to work.

On UNIX platforms, your window setting should be *"Point in Window to Make Active"* instead of *"Click in Window to Make Active"* for this function to work correctly. You can set this from the style manager on your machine. This setting is not required when you use this function with a bindkey.

### Arguments

*w\_window*

The window that you want to pan. If not specified, the current window is used.

**Note:** *w\_window* cannot be a session window or a dockable window.

## Cadence User Interface SKILL Reference

### Window Functions

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>n_scope</i>       | <p>Defines the panning range.</p> <p>For example, if a full bounding box in a design is set at (-1000:-1000 1000:1000) and <i>n_scope</i>=1.2, then you get a panning range of (-1100:-1100 1100:1100) which is 20% greater than the bounding box setting.</p> <p>If the <i>n_scope</i> argument is not specified, the default value of the argument is 0. This means there are no limits on the range making it possible to pan well beyond the design bounding box.</p> |
| <i>g_noAnimation</i> | <p>Specifies whether to skip the animation affect during dynamic panning. The default is <i>nil</i>, which means that the animation affect is not skipped.</p>                                                                                                                                                                                                                                                                                                            |

### Values Returned

|            |                                                  |
|------------|--------------------------------------------------|
| <i>t</i>   | Returns <i>t</i> when you exit the panning mode. |
| <i>nil</i> | There was an error.                              |

### Examples

```
hiDynamicPanInteractive(hiGetCurrentWindow() 1.2)
```

### ***Related Topics***

[hiDynamicPanAbsolute](#)

[hiDynamicPanRelative](#)

[hiPan](#)

[hiVectorPan](#)

[hiDeltaPan](#)

[hiAbsolutePan](#)

[hiUnregZoomPanProc](#)

## hiDynamicPanGrabbing

```
hiDynamicPanGrabbing(
 [w_window]
 [l_vector]

)
=> t / nil
```

### Description

Pans (grabs and moves) a design in the specified (or current) window from one point to another. You can even pan the design when editing it. The panning operation ends when you release holding the mouse button or DrawThru bindkey.

You can set up a DrawThru bindkey with this function. To enter the vector values by using a bindkey, you have to press, hold, drag, and release the bindkey.

To exit the panning mode, press the Esc key. Pressing Esc stops panning the image and brings it back to its original position. This is especially useful when you do not want to pan the design but just take a "peek" at what is beyond the visible area.

### Arguments

|                 |                                                                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | Optional. If not specified, the function uses the current window.                                                                                                 |
| <i>l_vector</i> | Optional.<br><br>If not specified, the function lets you interactively input the values for a vector by pressing two points in the specified (or current) window. |

### Value Returned

|            |                                                              |
|------------|--------------------------------------------------------------|
| <i>t</i>   | Returns <i>t</i> when you successfully exit the panning mode |
| <i>nil</i> | Returns <i>nil</i> if you encounter an error                 |

### Example

```
hiSetBindKey("Layout" "<DrawThru1>" "hiDynamicPanGrabbing()")
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiRedraw

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

#### Description

Redraws the contents of a window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Argument

*w\_window*

The ID of the window whose contents you want to redraw.  
*w\_window* cannot be a session window or a dockable window.  
If you do not specify this argument, the current window is used.

#### Values Returned

t

The contents of the window are redrawn.

nil

zoomPanProc is not registered on the window.

#### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiRegZoomPanProc

```
hiRegZoomPanProc (
 t_procedureName
 [w_window]
)
=> t / nil
```

### Description

Registers a SKILL procedure for a graphics window that gets called whenever a user uses a zoom, pan, fit, or redraw command in that window. If you do not specify a window while registering the procedure, the procedure gets called whenever the user uses these commands in any graphics window. The SKILL procedure is passed the window ID.

To unregister the procedure, use the `hiUnregZoomPanProc` function.

### Arguments

|                        |                                                                                                                                                                                                                                                                                                     |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_procedureName</i> | The name of the SKILL procedure to be called when the user uses a zoom, pan, fit, or redraw command in a graphics window.<br><br>This SKILL procedure passes two arguments: the first argument is the window ID and the second argument is the bounding box of the display specified in user units. |
| <i>w_window</i>        | The ID of the window for which you want to register the SKILL procedure.<br><br>If you do not specify this argument, the procedure is called whenever the user uses a zoom, pan, fit, or redraw command in any graphics window.                                                                     |

### Value Returned

|            |                               |
|------------|-------------------------------|
| <i>t</i>   | The procedure was registered. |
| <i>nil</i> | There was an error.           |

### Related Topics

[hiUnregZoomPanProc](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

#### Viewing Functions

## **hiUnregZoomPanProc**

```
hiUnregZoomPanProc(
 t_procedure
)
=> t / nil
```

### **Description**

Unregisters a SKILL procedure that was registered with the `hiRegZoomPanProc` function and that is called whenever the user uses a zoom, pan, fit, or redraw command in a graphics window.

### **Argument**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <i>t_procedure</i> | The name of the SKILL procedure to be unregistered. |
|--------------------|-----------------------------------------------------|

### **Value Returned**

|            |                                 |
|------------|---------------------------------|
| <i>t</i>   | The procedure was unregistered. |
| <i>nil</i> | There was an error.             |

### ***Related Topics***

[hiRegZoomPanProc](#)



## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiSaveView

```
hiSaveView(
 t_viewName
 [w_window]
)
=> t / nil
```

#### Description

Saves the current viewing parameters of a graphics window.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Arguments

|                   |                                                                                   |
|-------------------|-----------------------------------------------------------------------------------|
| <i>t_viewName</i> | String representing the name of the viewing parameter file.                       |
| <i>w_window</i>   | The window ID of a graphics window. If not specified, the current window is used. |

#### Value Returned

|            |                                                           |
|------------|-----------------------------------------------------------|
| <i>t</i>   | The parameters are saved.                                 |
| <i>nil</i> | <i>w_window</i> is invalid or there is no current window. |

#### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiListView

```
hiListView(
 [t_viewName]
 [w_window]
)
=> l_viewList / nil
```

#### Description

Lists the saved viewing parameters for a given view, or for all views if *viewName* is not specified. This function applies to graphics windows only

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_viewName</i> | Name of the view. If a <i>viewName</i> is specified, the <i>viewList</i> returned is the bounding box of that view (equivalent to <code>hiGetViewBBox</code> ). If no <i>viewName</i> is specified, the function lists the bounding boxes for each saved view. This list of lists contains the view name and the bounding box associated with each saved view. |
| <i>w_window</i>   | The ID of a graphics window. If not specified, the current window is used.                                                                                                                                                                                                                                                                                     |

#### Value Returned

|                   |                                                                                                         |
|-------------------|---------------------------------------------------------------------------------------------------------|
| <i>l_viewList</i> | A list of the view name and bounding box for each view specified.                                       |
| <i>nil</i>        | <i>w_window</i> is invalid, there is no current window, or the specified <i>viewName</i> was not found. |

#### Related Topics

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiRestoreView

```
hiRestoreView(
 t_viewName
 [w_window]
)
=> t / nil
```

#### Description

Restores a graphics window to the view specified by *viewName*.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

#### Arguments

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <i>t_viewName</i> | Name of the view.                                                          |
| <i>w_window</i>   | The ID of a graphics window. If not specified, the current window is used. |

#### Value Returned

|            |                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The view is restored.                                                                                   |
| <i>nil</i> | <i>w_window</i> is invalid, there is no current window, or the specified <i>viewName</i> was not found. |

#### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiPrevWinView

```
hiPrevWinView(
 w_window
)
=> t / nil
```

#### Description

Restores a graphics window to the view displayed before the last zoom or pan command.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Argument

|                 |                              |
|-----------------|------------------------------|
| <i>w_window</i> | The ID of a graphics window. |
|-----------------|------------------------------|

#### Value Returned

|            |                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The view is restored. If there was no previous zoom or pan in this window, the display is not changed and this function returns <i>t</i> . |
| <i>nil</i> | <i>w_window</i> is invalid.                                                                                                                |

#### ***Related Topics***

[Viewing Functions](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiNextWinView

```
hiNextWinView(
 w_window
)
=> t / nil
```

#### Description

Undoes `hiPrevWinView` on a graphics window.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL–SKILL Routines](#).

#### Argument

|                       |                              |
|-----------------------|------------------------------|
| <code>w_window</code> | The ID of a graphics window. |
|-----------------------|------------------------------|

#### Value Returned

|                  |                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | The view is restored. If there was no previous zoom or pan in this window, the display is not changed and this function returns <code>t</code> . |
| <code>nil</code> | <code>w_window</code> is invalid.                                                                                                                |

#### ***Related Topics***

[Viewing Functions](#)

[hiPrevWinView](#)

## hiGetUndoLimit

```
hiGetUndoLimit (
)
 => n_undoLimit
```

### Description

Returns the maximum number of `hiUndo` commands that can be performed in sequence. This number is set with the `hiSetUndoLimit` function or with the *Undo Limit* field in the CIW User Preferences form.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

None.

### Value Returned

|                    |                                                                                    |
|--------------------|------------------------------------------------------------------------------------|
| <i>n_undoLimit</i> | The maximum number of <code>hiUndo</code> commands that it is possible to perform. |
|--------------------|------------------------------------------------------------------------------------|

### ***Related Topics***

[hiUndo](#)

[Viewing Functions](#)

## hiSetUndoLimit

```
hiSetUndoLimit(
 n_undoLimit
)
=> t / nil
```

### Description

Sets the maximum number of `hiUndo` commands that can be performed in sequence. Specify either 0 or 128; 0 disables the undo command and 128 sets the undo limit to 128. If you specify any number other than 0, the undo limit is 128.

You can also set the undo limit in the *Undo Limit* field in the CIW User Preferences form.

**Note:** This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Argument

|                    |                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>n_undoLimit</i> | The maximum number of <code>hiUndo</code> commands that can be performed. Specify either 0 or 128. A setting of 0 disables the undo command, a setting of 128 sets the maximum to 128. Any value other than 0 sets the limit to 128. |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|     |                                                                       |
|-----|-----------------------------------------------------------------------|
| t   | The undo limit was successfully set.                                  |
| nil | <i>n_undoLimit</i> is non-zero and undo processing cannot be enabled. |

### Related Topics

[hiUndo](#)

[hiSetUndoLimit](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiUndo

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

### Description

Undoes the last user command that modified the database and that has not been undone by a previous `hiUndo`.

The number of `hiUndo` functions that can be performed in sequence is controlled by the User Preferences form or by using the `hiSetUndoLimit` function.

Not all commands that modify the database can be undone. Also `hiUndo` might not succeed for large designs because of memory limitations.

`hiUndo` undoes the latest sequential database modification without regard to which window it was performed in. (In other words, if Undo is selected from the menu of window *X*, it might undo a database operation performed in window *Y*, if that was the last operation performed.)

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

### Arguments

None.

### Value Returned

|                  |                                       |
|------------------|---------------------------------------|
| <code>t</code>   | An undo was successfully performed.   |
| <code>nil</code> | The last command could not be undone. |

### ***Related Topics***

[hiSetUndoLimit](#)

[Viewing Functions](#)



## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiRedo

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

#### Description

Undoes the last `hiUndo`, assuming no other database modifications have occurred.

`hiRedo` might not succeed after a successful `hiUndo` on large designs because of memory limitations.

This function is not available in Concept SKILL. For more information, see [Design Entry HDL-SKILL Routines](#).

#### Arguments

None.

#### Value Returned

|                  |                                       |
|------------------|---------------------------------------|
| <code>t</code>   | A redo was successfully performed.    |
| <code>nil</code> | The last command could not be redone. |

#### *Related Topics*

[hiUndo](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiEnableBlink

```
hiEnableBlink(
 g_blink
)
=> t / nil
```

#### Description

Enables or disables blinking, depending on the value of the *g\_blink* argument. When blinking is disabled, objects that would normally blink in graphics windows stops blinking. In some cases, disabling blinking can improve graphics performance. You can also disable blinking by starting the application with the `-noblink` option.

#### Argument

|                |                   |
|----------------|-------------------|
| <i>g_blink</i> | Enables blinking. |
|----------------|-------------------|

#### Value Returned

|     |                                                 |
|-----|-------------------------------------------------|
| t   | Blinking was enabled or disabled, as specified. |
| nil | There was an error.                             |

#### ***Related Topics***

[hilsBlinkEnabled](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### **hiIsBlinkEnabled**

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

#### **Description**

Checks whether blinking is enabled or not. You can use `hiEnableBlink` to enable or disable blinking in your current session.

#### **Arguments**

None.

#### **Value Returned**

|                  |                       |
|------------------|-----------------------|
| <code>t</code>   | Blinking is enabled.  |
| <code>nil</code> | Blinking is disabled. |

#### ***Related Topics***

[hiEnableBlink](#)

## World View Functions

This topic provides a list of Cadence® SKILL functions associated with World View assistants.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiCreateWorldView](#)
- [hiWorldViewFit](#)
- [hiWorldViewRedraw](#)
- [hiWorldViewZoomIn](#)
- [hiWorldViewZoomOut](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiCreateWorldView

```
hiCreateWorldView(
 w_windowId
)
=> dwindowID / nil
```

#### Description

Creates a World View assistant for your graphics window, which you can then display with `hiDockWindow`. World View assistant is a small window that shows you a picture of your entire design and marks the part of the design that is currently displayed in the graphics window. You can also use it to navigate the design or zoom in or out of it.

#### Argument

|                         |                                                                                                                                               |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_windowId</code> | The graphics window or the session window that contains the graphics window. For example, <code>window(3)</code> or <code>swindow(1)</code> . |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

#### Value Returned

|                        |                                             |
|------------------------|---------------------------------------------|
| <code>dwindowID</code> | The ID of the dock window that was created. |
| <code>nil</code>       | There was an error.                         |

#### ***Related Topics***

[World View Functions](#)

[hiUndockWindow](#)

[hiRegCloseProc](#)

## hiWorldViewFit

```
hiWorldViewFit(
 [w_windowId]
)
=> t / nil
```

### Description

Fits the current canvas display area in the World View assistant of the specified graphics window.

### Argument

|                   |                                                                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | The graphics window or the session window that contains the World View assistant. If not specified, the default is the current graphics window. |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|

**Note:** If a session window is specified, the function uses the session window ID to find the current graphics window in it and then uses this graphics window as the input (*w\_windowId*).

### Value Returned

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | There was an error.          |

### Examples

Fits the current canvas display area in the World View assistant of the graphics window 2.

```
hiWorldViewFit(window(2))
hiWorldViewFit()
```

### ***Related Topics***

[World View Functions](#)

## hiWorldViewRedraw

```
hiWorldViewRedraw(
 [w_windowId]
)
=> t / nil
```

### Description

Redraws the current canvas display area in the World View assistant of the specified graphics window.

### Argument

*w\_windowId*

The graphics window or the session window that contains the World View assistant. If not specified, the default is the current graphics window.

**Note:** If a session window is specified, the function uses the session window ID to find the current graphics window in it and use this graphics window as the input (*w\_windowId*).

### Value Returned

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | There was an error.          |

### Examples

Redraws the current canvas display area in the World View assistant of the graphics window 2.

```
hiWorldViewRedraw(window(2))
 hiWorldViewRedraw()
```

### ***Related Topics***

[World View Functions](#)

## hiWorldViewZoomIn

```
hiWorldViewZoomIn(
 [w_windowId]
)
=> t / nil
```

### Description

Zooms into the current canvas display area open in the World View assistant of the specified graphics window.

### Argument

*w\_windowId*

The graphics window or the session window that contains the World View assistant. If not specified, the default is the current graphics window.

**Note:** If a session window is specified, the function uses the session window ID to find the current graphics window in it and use this graphics window as the input (*w\_windowId*).

### Value Returned

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | There was an error.          |

### Examples

Zooms into the current canvas display area open in the World View assistant of the graphics window 2.

```
hiWorldViewZoomIn(window(2))
hiWorldViewZoomIn()
```

### ***Related Topics***

[World View Functions](#)



## hiWorldViewZoomOut

```
hiWorldViewZoomOut(
 [w_windowId]
)
=> t / nil
```

### Description

Zooms out of the current canvas display area open in the World View assistant of the specified graphics window.

### Argument

|                   |                                                                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | The graphics window or the session window that contains the World View assistant. If not specified, the default is the current graphics window. |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|

**Note:** If a session window is specified, the function uses the session window ID to find the current graphics window in it and use this graphics window as the input (*w\_windowId*).

### Value Returned

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | There was an error.          |

### Examples

Zooms out of the current canvas display area open in the World View assistant of the graphics window 2.

```
hiWorldViewZoomOut(window(2))
hiWorldViewZoomOut()
```

### ***Related Topics***

[World View Functions](#)

## Thumbnail View Functions

This topic provides a list of Cadence® SKILL functions associated with thumbnail views.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiGenerateThumbnails](#)
- [hiGetDefaultThumbnail](#)
- [hiGetDefaultThumbnailResolution](#)
- [hiGetThumbnailFilename](#)

## hiGenerateThumbnails

```
hiGenerateThumbnails(
 ?lib g_lib
 [?cell g_cell]
 [?view g_view]
 [?sizes g_sizes]
 [?output g_output]
 [?verbose g_verbose]
 [?exportArgs g_exportArgs]
 [?filePath t_filePath]
)
=> thumbnails / nil
```

### Description

Generates thumbnails for given library cellviews. Function behavior depends on arguments provided. All arguments can be lists in order to batch-generate thumbnails for a series of libraries, cells, views and sizes. At present, only the default 128x128 thumbnail is used by Cadence tools.

Regular expressions and wildcards are not supported.

### Arguments

|                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>?lib <i>g_lib</i></code>       | The library that contains the cellviews to generate thumbnails for.                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>?cell <i>g_cell</i></code>     | The cell that contains the cellviews to generate thumbnails for.                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>?view <i>g_view</i></code>     | The cellview to generate a thumbnail for.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>?sizes <i>g_sizes</i></code>   | The size of the thumbnails to be generated. The <i>g_sizes</i> argument can be an integer, an integer pair, or a list of integers or integer pairs. Single integers are interpreted as both the width and height. To generate a 64x64 thumbnail, specify a size of either 64 or 64 64 as both are considered the same. If no size is supplied, then either all existing thumbnail resolutions gets updated, or a new thumbnail is created at the default resolution (currently 128x128). |
| <code>?output <i>g_output</i></code> | Enter a symbol indicating where output should display. For example, the default <code>'progressbar</code> uses the embedded CIW progress bar.                                                                                                                                                                                                                                                                                                                                            |

## Cadence User Interface SKILL Reference

### Window Functions

---

`?verbose` *g\_verbose* Set to print the name of the file which was saved to the CIW.  
Default value: `nil`

`?exportArgs` *g\_exportArgs*

Set to allow arguments to be passed to the underlying `hiExportImage` function, consequently allowing further customization of the generated thumbnails.

`exportArgs` takes a list of pairs (an association table), which are passed to `hiExportImage`. For any argument that you want to pass to the export image, via thumbnails, simply add the keyword and the argument into the list. For example, to generate grayscale thumbnails with a white background:

```
hiGenerateThumbnails(?lib mylib ?cell mycell
?view myview ?exportArgs list(list('colorType
'monochrome) list('bgColor "white")
list('fgColor "black"))
```

`?filePath` *t\_filePath*

## Cadence User Interface SKILL Reference

### Window Functions

---

Set to allow the location of the generated thumbnail to be customized.

`filePath` provides for a wide range of control over where a thumbnail file is generated:

- For a simple string like "myDir", thumbnails gets generated hierarchically under that directory, for example: `myDir/lib/cell/view/thumbnail_128x128.png`.
- If the string contains a '/', the thumbnail code creates the directory if necessary.
- `printf`-style formatting strings can be used in the string. Currently supported are: %L (lib), %C (cell), %V (view), %W (width) and %H (height).

**Note:** These are all expanded at the time the thumbnail is generated.

- The above features therefore allow the filename to be highly customized, such as:

```
?filePath "%L/%C/%V/%Wx%H/thumbnail.png",
eg. "overview/follower/layout/128x128/
thumbnail.png"
```

```
?filePath "%L_%C_%V-%Wx%H.png", eg.
"overview_follower_layout-128x128.png"
```

```
?filePath "myThumbs/lib_%L/cell_%C/
view_%V", eg. "myThumbs/lib_overview/
cell_follower/view_layout/
thumbnail_128x128.png"
```

**Note:** In the last example, the default filename is appended if no filename is presented.

### Value Returned

`thumbnails`

Number of thumbnails successfully generated.

`nil`

Command failed.

## Cadence User Interface SKILL Reference

### Window Functions

---

#### Examples

Generates a thumbnail for the `acpd/follower/layout` view.

```
hiGenerateThumbnails(?lib "acpd" ?cell "follower" ?view "layout")
```

Generates thumbnails for all supported cellview types in the `acpd` library.

```
hiGenerateThumbnails(?lib "apcd")
```

Generates thumbnails for all supported cellview types in the `acpd` and `analogLib` libraries.

```
hiGenerateThumbnails(?lib ' ("acpd" "analogLib"))
```

Generates thumbnails for all libraries that are writable.

```
hiGenerateThumbnails(?lib 'AllWritable)
```

Generates, for each schematic and symbol view in the `acpd` library, the following series of thumbnails in the view directories: 64x64, 128x128, 256x256, 128x256.

```
hiGenerateThumbnails(?lib "acpd" ?view ' ("schematic" "symbol") ?sizes list(64 128 256 128:256)
```

```
hiGenerateThumbnails(?lib "ether" ?sizes list(1024))
```

**Note:** The `sizes` argument is a list of either pairs or numbers. Pairs are interpreted as the width and height, single numbers are interpreted as being both the width and the height (hence it is sufficient to just pass 1024 above, rather than `(1024:1024)`). If you required a different height and width, you could use `?sizes list(1024:512)`.

#### ***Related Topics***

[Thumbnail View Functions](#)

[hiGetDefaultThumbnailResolution](#)

[hiGetThumbnailFilename](#)

[hiGetDefaultThumbnail](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiGetDefaultThumbnail

```
hiGetDefaultThumbnail(
 ?viewType t_viewType
 [?size g_size]
)
=> icon / nil
```

### Description

Displays a default cellview type thumbnail when a thumbnail is not available for that application. This prevents the thumbnail area of a dialog or window from being left blank.

### Arguments

*?viewType t\_viewType*

Refers to the database view type. Supported view types are maskLayout, symbol, and schematicSymbol.

*?size g\_size*

Fixed number or fixed number pair (for example 123:456) that indicates the requested size of the image.

### Value Returned

*icon*

Thumbnail icon successfully displayed.

*nil*

Cellview type not recognized or no thumbnail available at the requested resolution. If not supplied, defaults to hiGetDefaultThumbnailResolution() return value.

### Example

```
myImage = hiGetDefaultThumbnail(?viewType "maskLayout")
```

### Related Topics

[Thumbnail View Functions](#)

[hiGenerateThumbnails](#)

[hiGetDefaultThumbnailResolution](#)

## hiGetDefaultThumbnailResolution

```
hiGetDefaultThumbnailResolution()
=> x_width x_height / nil
```

### Description

Returns a list that comprises of a pair of integers that indicate the default thumbnail resolution that applications are currently using.

### Argument

None

### Value Returned

|                 |                                                |
|-----------------|------------------------------------------------|
| <i>x_width</i>  | The default thumbnail resolution width.        |
| <i>x_height</i> | The default thumbnail resolution height.       |
| <i>nil</i>      | Failed to return thumbnail resolution details. |

### Example

```
hiGetThumbnailDefaultResolution
=> (128 128)
```

### ***Related Topics***

[Thumbnail View Functions](#)

[hiGenerateThumbnails](#)

[hiGetDefaultThumbnail](#)

[hiGetThumbnailFilename](#)



## hiGetDefaultWidgetSpacing

```
hiGetDefaultWidgetSpacing(
 [?widgetsAreRelated t | nil]
)
=> x_value
```

### Description

Returns the recommended number of pixels between UI elements. The spacing between related elements, such as a group of Boolean buttons, is smaller than elements, which are not related.

### Argument

`?widgetsAreRelated`

Spacing size between UI elements.

Default value: `nil`

### Value Returned

`x_value`

Number of pixels between UI elements.

### Example

```
myForm=(hiCreateLayoutForm
 'myForm
 "Grouped Buttons"
 (hiCreateVerticalBoxLayout 'rootLayout
 ?items
 (list
 (hiCreateVerticalBoxLayout 'vlyt
 ?frame "Related items"
 ?spacing (hiGetDefaultWidgetSpacing ?widgetsAreRelated t)
 ?items (list
 (hiCreateBooleanButton ?name 'bb ?buttonText "Apples")
 (hiCreateRadioField ?name 'rf
 ?choices (list "Gala" "Honeycrisp" "Braeburn")
 ?defValue "Braeburn"
 ?prompt "Type:")
)))
 ?buttonLayout 'Close
 ?initialSize (300:100))
 hiDisplayForm(myForm)
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Thumbnail View Functions](#)

## hiGetThumbnailFilename

```
hiGetThumbnailFilename(
 [?size g_size]
)
=> s_filename / nil
```

### Description

Returns the name of a thumbnail file for a given resolution. If no specific size is requested, the filename for the default thumbnail resolution is used (as set by `hiGetThumbnailDefaultResolution`).

The `hiGetThumbnailFilename` function does not check for the existence of the thumbnail, rather it instructs applications which file to look for in the view directory for a given resolution.

### Argument

|                           |                                                                                                                                                                                                                                                                                                              |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>?size g_size</code> | Optional argument that can be either a single integer (indicating a thumbnail with the same width and height) or a list of two integers specifying the thumbnail width and height. If the size argument is not supplied, it defaults to the return value from <code>hiGetDefaultThumbnailResolution</code> . |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|                         |                                  |
|-------------------------|----------------------------------|
| <code>s_filename</code> | Name of thumbnail file returned. |
| <code>nil</code>        | No thumbnail file found.         |

### Example

```
hiGetThumbnailFilename()
=> "thumbnail_128x128.png"
hiGetThumbnailFilename(?size 256)
=> "thumbnail_256x256.png"
hiGetThumbnailFilename(?size 128:256)
=> "thumbnail_128x256.png"
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Thumbnail View Functions](#)

[hiGenerateThumbnails](#)

[hiGetDefaultThumbnail](#)

[hiGetDefaultThumbnailResolution](#)

## Workspace Functions

This topic provides a list of Cadence® SKILL functions associated with windows.

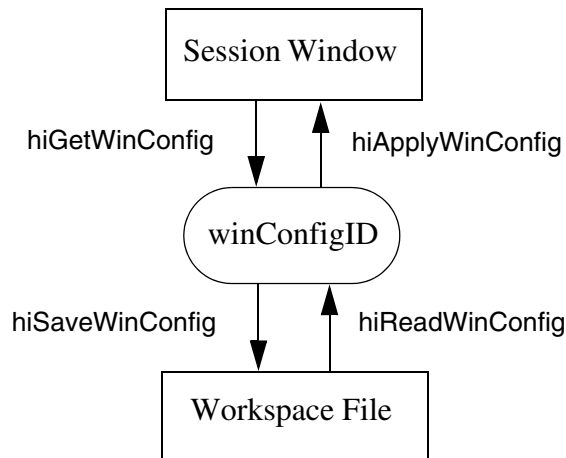
A workspace, also known as a window configuration, contains session window layout state information. Specifically, it stores the following information:

- Which dock assistants are displayed and whether they are docked, floating, or hidden
- Which toolbars are displayed and whether they are docked, floating, or hidden
- The size and position of the dock assistants
- The size and position of the toolbars

Workspaces do not include any information about menus.

You can create workspaces interactively with your Cadence application and save them to files for later use.

HI window configuration functions let you read and write workspace files programmatically as well as retrieve and apply in-memory workspaces to session windows.



You can use the `hiGetWinConfig` function to retrieve a workspace from a session window programmatically and either apply it to another session window with `hiApplyWinConfig` or save it to a file with the `hiSaveWinConfig` function, or both. To apply a saved workspace, you read it from the file with the `hiReadWinConfig` function, which returns the window configuration ID, and then use `hiApplyWinConfig` to apply it to a session window.

Every toolbar and dock assistant must have a unique name that does not change between invocations or from session window to session window.

## Cadence User Interface SKILL Reference

### Window Functions

---

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- hiGetWinConfig
- hiSaveWinConfig
- hiReadWinConfig
- hiApplyWinConfig
- hiWinConfigModified
- hiGetWinConfigInfo
- hiSetWinObjName
- hiGetWinObjName

## hiGetWinConfig

```
hiGetWinConfig(
 w_swindowID
)
=> n_configID / nil
```

### Description

Creates a new workspace based on the session window layout and returns its ID.

If a workspace has not been modified since the last use of `hiApplyWinConfig` or `hiGetWinConfig`, then the same session window ID gets returned, rather than a new window configuration being created.

### Argument

|                          |                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------|
| <code>w_swindowID</code> | The session window ID. You can get a session window ID with the <code>swindow</code> function. |
|--------------------------|------------------------------------------------------------------------------------------------|

### Value Returned

|                         |                                                                                                                 |
|-------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>n_configID</code> | The new workspace ID.                                                                                           |
| <code>nil</code>        | An invalid window (for example, a window that is not a session window) was specified. A warning is also issued. |

### Example

```
hiGetWinConfig(swindow (1))
```

### ***Related Topics***

[Workspace Functions](#)

[hiSaveWinConfig](#)

[hiApplyWinConfig](#)

[hiWinConfigModified](#)

[hiGetWinConfigInfo](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiSetWinObjName

hiGetWinObjName



## Cadence User Interface SKILL Reference

### Window Functions

---

## hiSaveWinConfig

```
hiSaveWinConfig(
 n_configID
 t_pathToFile
 [g_overwrite]
)
=> t / nil
```

### Description

Saves a workspace to a file. The *g\_overwrite* argument lets you specify whether or not to overwrite an existing file.

### Arguments

|                     |                                                                                                                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>n_configID</i>   | The ID of the workspace that you want to save to a file. You get this ID with the <code>hiGetWinConfig</code> function.                                                                           |
| <i>t_pathToFile</i> | The file to which you want to save the workspace.                                                                                                                                                 |
| <i>g_overwrite</i>  | <i>nil</i> or non- <i>nil</i> . Specify <i>t</i> or any non- <i>nil</i> value to overwrite an existing file. Specify <i>nil</i> if you do not want to overwrite <i>t_pathToFile</i> if it exists. |

### Value Returned

|            |                                               |
|------------|-----------------------------------------------|
| <i>t</i>   | The workspace was saved to the file.          |
| <i>nil</i> | There was an error. A warning is also issued. |

### Example

```
hiSaveWinConfig(myConfigID "./myConfigFile" t)
```

### Related Topics

[Workspace Functions](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiApplyWinConfig

hiWinConfigModified

hiGetWinConfigInfo

hiSetWinObjName

hiGetWinObjName

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiReadWinConfig

```
hiReadWinConfig(
 t_pathToFile
)
=> n_configID / nil
```

### Description

Reads a workspace file previously saved by `hiSaveWinConfig` and returns the workspace ID.

### Argument

|                           |                                             |
|---------------------------|---------------------------------------------|
| <code>t_pathToFile</code> | The file that contains the saved workspace. |
|---------------------------|---------------------------------------------|

### Value Returned

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>n_configID</code> | The ID of the workspace from the file.        |
| <code>nil</code>        | There was an error. A warning is also issued. |

### Example

```
hiReadWinConfig("./myConfigFile")
```

### ***Related Topics***

[Workspace Functions](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

[hiApplyWinConfig](#)

[hiWinConfigModified](#)

[hiGetWinConfigInfo](#)

[hiSetWinObjName](#)

[hiGetWinObjName](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiApplyWinConfig

```
hiApplyWinConfig(
 w_swindowID
 n_configID
)
=> t / nil
```

### Description

Applies a workspace to a session window. If the session window has toolbars and dock assistants that were not present when the workspace was created, they get hidden.

### Arguments

|                    |                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_swindowID</i> | The session window to which you want to apply a workspace. You can get a session window ID with the <code>swindow</code> function. |
| <i>n_configID</i>  | The workspace ID. You get this ID with the <code>hiGetWinConfig</code> or <code>hiReadWinConfig</code> functions.                  |

### Value Returned

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The workspace was applied to the session window.                                                             |
| <i>nil</i> | There was an error. For example, <i>n_configID</i> is invalid or <i>w_swindowID</i> is not a session window. |

### Example

```
myConfigID=hiReadWinConfig("./myConfigFile")
hiApplyWinConfig(swindow (2) myConfigID)
```

### ***Related Topics***

[Workspace Functions](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

[hiReadWinConfig](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiWinConfigModified

hiGetWinConfigInfo

hiSetWinObjName

hiGetWinObjName

## hiWinConfigModified

```
hiWinConfigModified(
 w_swindowID
 [x_winConfigID]
)
=> t / nil
```

### Description

Checks whether a workspace has been modified since it was last applied to the session window.

### Arguments

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_swindowID</i>   | The session window ID. You can get a session window ID with the <code>swindow</code> function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <i>x_winConfigID</i> | <p>The window configuration ID.</p> <p>If this is specified, <code>hiWinConfigModified</code> only returns <code>t</code> if the workspace has been modified since <code>x_winConfigID</code> was last applied to the <code>w_swindowID</code>, or if <code>x_winConfigID</code> has never been applied to the <code>w_swindowID</code>.</p> <p>If this is not specified, and the window has been modified such that the <code>saveState()</code> data has changed since the last use of <code>hiGetWinConfig</code> or <code>hiApplyWinConfig</code>, then <code>hiWinConfigModified</code> returns <code>t</code>.</p> |

### Value Returned

|                  |                                                                                    |
|------------------|------------------------------------------------------------------------------------|
| <code>t</code>   | Workspace has been modified. See <code>x_winConfigID</code> for circumstances.     |
| <code>nil</code> | Workspace has not been modified. See <code>x_winConfigID</code> for circumstances. |

### Example

```
hiWinConfig(w_swindowID @optional x_winConfigID
=> t
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Workspace Functions](#)

[hiWinConfigModified](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

[hiReadWinConfig](#)

[hiApplyWinConfig](#)

[hiGetWinConfigInfo](#)

[hiSetWinObjName](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiGetWinConfigInfo

```
hiGetWinConfigInfo(
 n_configID
)
=> l_infoList / nil
```

#### Description

Returns information about a workspace (window configuration): the list of dock assistants and toolbars that are part of the workspace and whether they are visible, hidden, or floating.

#### Argument

|                   |                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>n_configID</i> | The workspace ID. You get this ID with the <code>hiGetWinConfig</code> or <code>hiReadWinConfig</code> functions. |
|-------------------|-------------------------------------------------------------------------------------------------------------------|

#### Value Returned

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_infoList</i> | <p>A list of two lists. The first list contains information about the workspace's dock assistants; the second list contains information about the workspace's toolbars.</p> <p><i>l_infoList</i> has the following format:</p> <pre>(([(t_objectName s_state) [(t_objectName s_state)<br/>    [...] ]])<br/>  [(t_objectName s_state) [(t_objectName s_state)<br/>    [...] ]]) )</pre> <p>where:</p> <p><i>t_objectName</i> is, in the first list, the object name of a dock window and, in the second list, the object name of a toolbar. (Object names are the names that were set with <code>hiSetWinObjName</code> and <code>hiSetToolbarObjName</code>.)</p> <p><i>s_state</i> is one of the following symbols: 'floating, 'hidden, or 'visible. For toolbars, <i>s_state</i> can only be 'hidden or 'visible.</p> <p><i>nil</i> <i>n_configID</i> is not valid.</p> |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Workspace Functions](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

[hiReadWinConfig](#)

[hiApplyWinConfig](#)

[hiWinConfigModified](#)

[hiGetWinConfigInfo](#)

[hiSetWinObjName](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiSetWinObjName

```
hiSetWinObjName(
 w_dockWindowID
 t_name
)
=> t / nil
```

#### Description

Sets the window name which is used for storing workspace information about a dock assistant. Every dock assistant must have a unique name that does not change. Do not use this function with dock assistants provided by Cadence.

This function can also be used for any other type of window except a session window.

#### Arguments

|                       |                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------|
| <i>w_dockWindowID</i> | The ID of the dock window. You can get a dock window ID with the <code>dwindow</code> function. |
| <i>t_name</i>         | The window name you want to set.                                                                |

#### Value Returned

|            |                          |
|------------|--------------------------|
| <i>t</i>   | The window name was set. |
| <i>nil</i> | There was an error.      |

#### Example

```
hiSetWinObjName(dwindow(1) "hierAssistant")
```

#### ***Related Topics***

[Workspace Functions](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

[hiReadWinConfig](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

hiApplyWinConfig

hiWinConfigModified

hiGetWinConfigInfo

hiSetWinObjName

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiGetWinObjName

```
hiGetWinObjName (
 w_dockWindowID
)
=> t_name / nil
```

#### Description

Gets the window name that is used to store the dock assistant's workspace information. This name is set with the `hiSetWinObjName` function.

#### Argument

|                             |                                                                                          |
|-----------------------------|------------------------------------------------------------------------------------------|
| <code>w_dockWindowID</code> | The dock window ID. You can get a dock window ID with the <code>dwindow</code> function. |
|-----------------------------|------------------------------------------------------------------------------------------|

#### Value Returned

|                     |                                 |
|---------------------|---------------------------------|
| <code>t_name</code> | The window name.                |
| <code>nil</code>    | The window is a session window. |

#### Examples

```
hiGetWinObjName (dwindow (1))
```

#### ***Related Topics***

[Workspace Functions](#)

[hiGetWinConfig](#)

[hiSaveWinConfig](#)

[hiReadWinConfig](#)

[hiApplyWinConfig](#)

[hiWinConfigModified](#)

[hiGetWinConfigInfo](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

hiSetWinObjName

## Screen Capture Functions

This topic provides a list of Cadence® SKILL functions associated with screen captures.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- hiWindowSaveImage
- hiExportImage
- hiExportImageDialog

## hiWindowSaveImage

```
hiWindowSaveImage(
 [?target g_target]
 [?path t_path]
 [?format t_format]
 [?toplevel g_toplevel]
 [?quality n_quality]
 [?delay n_delay]
 [?timeoutBeep g_timeoutBeep]
 [?grabFromScreen g_grabFromScreen]
 [?centralWidget g_centralWidget]
)
=> t / nil
```

### Description

Takes a screen capture of a window, a region, or the entire screen. You can specify the object to capture either through the arguments or, in interactive mode, by selecting it with the mouse.

If you do not specify the `?target` argument, this function enters the interactive mode. You can then left-click to select a window or right-click and drag to select a region. If the `?toplevel` argument is set to `nil`, only the widget that is clicked, such as a toolbar, is captured. When you release the mouse, a Save Image As dialog box appears that lets you specify the file in which to save the image.

You can capture a region only in interactive mode.

## Cadence User Interface SKILL Reference

### Window Functions

---

#### Arguments

|                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>?target g_target</code>     | <p>The window, form, dialog box, or screen that you want to capture, or <code>nil</code>.</p> <ul style="list-style-type: none"><li>■ For windows, specify <code>window(n)</code> or <code>w_windowID</code>.</li><li>■ For dockable windows, specify <code>w_dwindowID</code> or <code>dwindow(n)</code>.</li><li>■ For session windows, specify <code>swindow(n)</code>.</li><li>■ For forms, specify the form symbol or <code>r_form</code>.</li><li>■ For a dialog box, specify the symbol that is set as the handle to the dialog box.</li><li>■ For the entire screen, specify the following symbol:<br/>'screen#, where # is the number of the display screen.<br/>For example, 'screen0 is the first screen, and if you have multiple screens, 'screen1 is the second one, and so forth.</li></ul> <p>To interactively select the window or region that you want to capture, specify <code>nil</code>. The function then switches to interactive mode. You can then left-click the window that you want to capture, or right-click and drag the mouse over the region you want to capture.</p> <p>Default value: <code>nil</code></p> |
| <code>?path t_path</code>         | <p>Path to the file in which to save the image. If you do not specify this argument, a Save Image As dialog box appears that lets you select the file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>?format t_format</code>     | <p>The image format in which to save the file. You can save the file in any of the formats supported by Qt; currently these are: BMP, JPEG, PNG, PPM, XBM, and XPM. However, PNG is the recommended format.</p> <p>Default value: <code>"png"</code></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>?toplevel g_toplevel</code> | <p>Displays the top of the window hierarchy and also captures the top-level window that includes the window manager frame. Otherwise, it only captures the widget that is specified.</p> <p>Default value: <code>t</code></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |



## Cadence User Interface SKILL Reference

### Window Functions

---

`?quality n_quality`

Number specifying the image quality for a JPEG image, if you are saving the image in JPEG format.

`?delay n_delay`

Time delay in seconds before the image is captured.

Default value: 0

`?timeoutBeep g_timeoutBeep`

Eliminates a beep when the screen capture is done if a delay is set.

Default value: t

`?grabFromScreen g_grabFromScreen`

Captures the image directly from the screen rather than drawing an image in the memory.

`?centralWidget g_centralWidget`

For session windows only grab the canvas, and for other windows and forms just grab the main section and not the window frame.

### Value Returned

t

The screen capture was successful.

nil

There was an error.

### Example

```
hiWindowSaveImage(?target window(1) ?path "mainWindow.png" ?format "png" ?delay 2)
hiWindowSaveImage(?target nil ?path "mySelectedRegion.png" ?format "png")
hiWindowSaveImage()
hiWindowSaveImage(?target 'screen0)
```

### Related Topics

[Screen Capture Functions](#)

[hilsWindowSpecifier](#)

[swindow](#)

## **Cadence User Interface SKILL Reference**

### Window Functions

---

windowp

hiCreateAppForm

hiGetCurrentForm

hiDisplayAppDBox

hiDisplayUserDBox

## hiExportImage

```
hiExportImage (
 ?fileName t_filename
 [?window w_window]
 [?exportRegion s_exportRegion]
 [?bBox l_bBox]
 [?width n_width]
 [?height n_height]
 [?scaleFactor n_scaleFactor]
 [?haloType s_haloType]
 [?scaleHalos g_scaleHalos]
 [?fileType g_fileType]
 [?keepAspect g_keepAspect]
 [?transparentBG g_transparentBG]
 [?colorType s_colorType]
 [?fgColor t_fgColor]
 [?bgColor t_bgColor]
 [?verbose g_verbose]
 [?windowProps g_windowProps]
 [?decorations g_decorations]
)
=> t / nil
```

### Description

Takes a screen capture of a design in a graphics window and exports it to a standard image format such as PNG. This function can only be used with graphics windows.

With this function, you can take a screen capture of the part of the design that is currently visible in the window, the entire design, or a specific region. You can specify the size of the resulting image, either by specifying a scaling factor or a custom size. This function maintains the aspect ratio of the original image.

You can also choose a background color for the image or specify a transparent background.

Screen captures taken with this function will also capture halos. You can adjust halos with the `?haloType` and `?scaleHalos` arguments.

You can save images in any of the formats supported by Qt: BMP, JPG, PNG, PPM, XBM, and XPM. To specify the format, use the appropriate suffix with the filename. Note, however, that PNG is the recommended format; not all features are supported by the other formats.

The shadow mode—in which unselected objects are dimmed and selected objects are displayed at full brightness in their original layer color instead of the selection color—is ignored by the export image functions. Objects are displayed in their normal colors and brightness in the exported image.

## Cadence User Interface SKILL Reference

### Window Functions

---

#### Arguments

?fileName *t\_filename*

The name of the file (or path) in which to save the exported image. The file name can contain any filesystem character.

The suffix of the file determines the image format. For example, if you specify `myfile.png`, the image get saved as a PNG file. All formats supported by Qt are supported; currently, these are: BMP, JPEG, PNG, PPM, XBM, and XPM. Note, however, that PNG is the recommended format; not all features are supported by the other formats.

?window *w\_window*

The window ID of the graphics window containing the design to export. You can get the window ID with the [hlsWindowSpecifier](#) function. You can also specify `window(n)` where *n* is the window number.

Default value: the current window

?exportRegion *s\_exportRegion*

The area of the design to capture. Specify one of the following symbols:

'visibleArea: The area of the design that is currently visible in the specified window

'entireDesign: The entire design, not just the part that is currently visible in the window

'selectedArea: The area selected by the user; see the description of the ?bBox argument.

If the *l\_bBox* argument is specified (and it is a valid bounding box), *s\_exportRegion* is automatically set to 'selectedArea.

Default value: 'visibleArea

## Cadence User Interface SKILL Reference

### Window Functions

---

?bBox *l\_bBox*

The screen coordinates of the area to capture, specified as a list of lower-left and upper-right coordinates (assuming 0:0 to be the lower-left corner of the screen):

```
list(x1:y1 x2:y2)
```

where *x1* is the lower-left x coordinate, *y1* is the lower-left y coordinate, *x2* is the upper-right x coordinate and *y2* is the upper-right y coordinate.

To provide the user the ability to interactively select the region to capture, you can specify `enterBox()` as the value of this argument. The user can then select the area to capture by clicking-and-dragging the mouse.

If a valid bounding box is specified for this argument, the *s\_exportRegion* argument is automatically set to 'selectedArea.

?width *n\_width*

The desired width of the image, specified in pixels.

Default value: 0

You can specify either the width or the height, or both. If you specify only the width or only the height (and the other value is 0 or unspecified), then the other value is calculated automatically according to the aspect ratio of the region being captured.

If you specify both the width and height, then they are taken to be the maximum values. In order to maintain the aspect ratio of the original region, either the width or height might be reduced; the resulting image is never larger than the width and height specified. For example, if you specify a width of 400 and a height of 400, and the area being captured is not as high as it is wide, then the resulting image is of 400 pixels wide but less than 400 pixels high to maintain the aspect ratio.

If both width and height are 0, the image remains the same size as the region being captured.

If you specify both *n\_scaleFactor* and either *n\_width* or *n\_height*, *n\_scaleFactor* is used.

## Cadence User Interface SKILL Reference

### Window Functions

---

`?height n_height` The desired height of the image, specified in pixels.  
Default value: 0

You can specify either the width or the height, or both. If you specify only the width or only the height (and the other value is 0 or unspecified), then the other value is calculated automatically according to the aspect ratio of the region being captured.

If you specify both the width and height, then they are taken to be the maximum values. In order to maintain the aspect ratio of the original region, either the width or height might be reduced; the resulting image is never larger than the width and height specified. For example, if you specify a width of 400 and a height of 400, and the area being captured is not as high as it is wide, then the resulting image is of 400 pixels wide but less than 400 pixels high to maintain the aspect ratio.

If both width and height are 0, the image remains the same size as the region being captured.

If you specify both *n\_scaleFactor* and either *n\_width* or *n\_height*, *n\_scaleFactor* is used.

`?scaleFactor n_scaleFactor`

The factor by which to scale the image. A number greater than 1.0 scales the image up; a number less than 1.0 scales it down. For example, a factor of 2.0 doubles the size of the image while a factor of 0.5 reduces its size by half.

Default value: 1.0

An image requires  $\text{<width>*<height>*<scaleFactor>*32}$  bits of memory, therefore scaling an image to a high factor can cause an out of memory condition. If insufficient memory to save an image is an issue, a warning message is displayed advising that you scale down the image size before attempting to save it again.

If you specify both *n\_scaleFactor* and either *n\_width* or *n\_height*, *n\_scaleFactor* is used.

`?haloType s_haloType`

## Cadence User Interface SKILL Reference

### Window Functions

---

One of the following symbols:

`'default`

`'simple`

specifying whether halos should be rendered as they appear on the screen (`'default`) or in a single translucent color (`'simple`). Simple halos typically look better than the fuzzy default halos when the image is printed.

Default value: `'default`

`?scaleHalos g_scaleHalos`

`t` or `nil`. Specifies whether to scale halos relative to the size of the image you capture. By default, halos are a constant size, regardless of whether you zoom in or out of the design, and can look too small or too large in the resulting image.

Default value: `nil`

`?fileType g_fileType`

Can be set as one of the supported formats mentioned in the *Description* section above. This option is not required unless you are using a non-standard file suffix, as it should be deduced automatically.

Default value: `" "`

`?keepAspect g_keepAspect`

Set to keep the aspect ratio of the original export area, adjusting the size constraints specified by `?width` and `?height` accordingly. By default, if you specify width and height, this generates the exact image size that you specify, with padding if the area being exported does not match the aspect ratio of the image (for example, padding on the sides if you export to a square image from a tall rectangular canvas). If set to `'t'`, it adjusts the size so that no padding is used.

Default value: `nil`

`?transparentBG g_transparentBG`

## Cadence User Interface SKILL Reference

### Window Functions

---

The color to use for the background color (for a transparent background, use the *s\_bgType* argument). You can specify the color as color name strings (for example, "red") or RGB values (for example, "#ff0000").

Default: Black

If you specify this argument, *s\_bgType* is automatically set to 'solid'.

*?colorType s\_colorType*

Set the type of image to export. This can be one of the following symbols:

- 'fullColor(default): generate image using the colors visible on-screen.
- 'monochrome: generate a monochrome image, with colors being interpolated between *?fgColor* and *?bgColor*. For example, if *?fgColor* is black and *?bgColor* is white, this generates a grayscale image with a white background and black-gray shapes. Conversely, *?bgColor* black and *?fgColor* black generates a grayscale image with a black background and white-gray shapes. There are no restrictions on color.
- 'biColor: generate a bicolor image, with *?bgColor* as the background color, and all other shapes rendered in *?fgColor*. For example, *?bgColor* white *?fgColor* black produces a white background with black shapes.

*?fgColor t\_fgColor*

The color to use as the foreground color for bicolor and monochrome exports.

Default value: " "

*?bgColor t\_bgColor*

The color to use as the background color for bicolor and monochrome exports.

Default value: " "

*?verbose g\_verbose*

Set to print the name of the file which was saved to the CIW.

Default value: nil

*?windowProps g\_windowProps*



## Cadence User Interface SKILL Reference

### Window Functions

---

A list of list of key-value pairs which represent window properties, and their values, to be set before export, and restored afterwards. It is used by the thumbnailing process to perform actions such as hiding the grid, but any window property is valid. For example:

```
?windowProps list(list("drawAxesOn" nil)
list("drawGridOn" nil)
```

Set the 'drawAxesOn' and 'drawGridOn' properties on the window to 'nil', and restore the original values immediately after export.

Default value: nil

`?decorations g_decorations`

A list of "decorations" to be rendered on a resulting image. Can be used to generate 'PCell' text on pcell thumbnails. It is a DPL that consists of key-value pairs indicating what to draw and where to draw it. Currently, only text is supported. For example:

```
?decorations ncons('(nil text "Pcell" corner
"bottomLeft"))
```

Will render 'PCell' in the bottom left. `corner` can be one of "bottomLeft", "bottomRight", "topLeft", "topRight".

Default value: nil

## Values Returned

|                  |                                                                       |
|------------------|-----------------------------------------------------------------------|
| <code>t</code>   | Returns <code>t</code> when the image file is saved.                  |
| <code>nil</code> | The window specified is not a valid window or there is another error. |

## Examples

Prompts the user to enter a bounding box, and when done, scales the selected region by 5, scales halos, and saves the image in PNG format in a file named `a.png`.

```
hiExportImage(?fileName "a.png" ?bBox enterBox() ?scaleFactor 5 ?scaleHalos t)
```

Creates an image of the visible area of the design, with a transparent background and a width of 10,000 pixels wide (and the height required to maintain the aspect ratio), and saves it to a file named `a.png`.

## Cadence User Interface SKILL Reference

### Window Functions

---

```
hiExportImage(?fileName "a.png" ?bgType 'transparent ?width 10000)
```

Creates an image of the entire design, regardless of the current zoom level, with a red background and a width of 10,000 pixels (and the height required to maintain the aspect ratio), and saves it to a file named `a.png`.

```
hiExportImage(?fileName "a.png" ?exportRegion 'entireDesign ?width 10000 ?bgColor "red")
```

Creates a 10,000-pixel-wide image of the entire design, with a transparent background and simple halos, and saves it in PNG format in a file named `a.png`.

```
hiExportImage(?fileName "a.png" ?bgType 'transparent ?exportRegion 'entireDesign ?haloType 'simple ?width 10000)
```

### ***Related Topics***

[Screen Capture Functions](#)

[hiExportImageDialog](#)

[hiWindowSaveImage](#)

## hiExportImageDialog

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

### Description

Displays the Export Image dialog box, which lets you take a screen capture of the design contained in the specified window and save it to a standard image format. This function can only be called for graphics windows.

Screen captures taken with this dialog box also capture halos.

Here, the shadow mode—in which unselected objects are dimmed and selected objects are displayed at full brightness in their original layer color instead of the selection color—is ignored. Objects are displayed in their normal colors and brightness in the exported image.

You can save images in any of the formats supported by Qt: BMP, JPG, PNG, PPM, XBM, and XPM. To specify the format, use the appropriate suffix with the filename. Note, however, that PNG is the recommended format; not all features are supported by the other formats.

### Argument

|                 |                                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | The window containing the design that you want to capture.<br><i>w_window</i> must be a graphics window. |
|-----------------|----------------------------------------------------------------------------------------------------------|

### Value Returned

|            |                                                                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | Returns <i>t</i> when the dialog is displayed.                                                                                                                   |
| <i>nil</i> | Returns <i>nil</i> if the dialog cannot be displayed. For example, if you specify an invalid window, an error is displayed and the function returns <i>nil</i> . |

### Example

```
hiExportImageDialog(window(3))
```

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[Screen Capture Functions](#)

[hiExportImage](#)

[hiWindowSaveImage](#)

## Splash Screen Functions

This topic provides a list of Cadence® SKILL functions associated with splash screens modification and windows. You can change the background, icon, and family and product names on a splash screen or About window.

These functions only apply to the Virtuoso Studio design environment.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiAbout](#)
- [hiSetSplashBackground](#)
- [hiSetSplashDefaultBackground](#)
- [hiSetSplashIcon](#)
- [hiSetSplashFamily](#)
- [hiSetSplashProduct](#)
- [hiSetSplashLicense](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

## hiAbout

```
hiAbout (
 w_windowId
)
=> t / nil
```

## Description

Displays an application's product information window, which includes the release version and copyright information. If *w\_windowId* is a session window, the product information is about the window that is displayed in the top tab.

## Argument

|                   |                                   |
|-------------------|-----------------------------------|
| <i>w_windowId</i> | The window ID of the application. |
|-------------------|-----------------------------------|

## Value Returned

|            |                                                  |
|------------|--------------------------------------------------|
| <i>t</i>   | The product information window is displayed.     |
| <i>nil</i> | The product information window is not displayed. |

## ***Related Topics***

[Splash Screen Functions](#)

[hiSetSplashBackground](#)

[hiSetSplashDefaultBackground](#)

[hiSetSplashIcon](#)

[hiSetSplashFamily](#)

[hiSetSplashProduct](#)

[hiSetSplashLicense](#)

## hiSetSplashBackground

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

### Description

Sets the background for an application's splash screen. If the graphic you specify has too many colors, the background is not set.

### Argument

|                   |                                                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>t_fileName</i> | The name of the file containing the background graphic (in any format supported by Qt). Specify the full path to the file. |
|-------------------|----------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|            |                             |
|------------|-----------------------------|
| <i>t</i>   | The background was set.     |
| <i>nil</i> | The background was not set. |

### ***Related Topics***

[Splash Screen Functions](#)

[hiAbout](#)

[hiSetSplashDefaultBackground](#)

[hiSetSplashIcon](#)

[hiSetSplashFamily](#)

[hiSetSplashProduct](#)

[hiSetSplashLicense](#)

## **hiSetSplashDefaultBackground**

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

### **Description**

Sets the splash screen background to the default background.

### **Value Returned**

|     |                                                       |
|-----|-------------------------------------------------------|
| t   | The background was set to the default background.     |
| nil | The background was not set to the default background. |

### ***Related Topics***

[Splash Screen Functions](#)

[hiAbout](#)

[hiSetSplashBackground](#)

[hiSetSplashIcon](#)

[hiSetSplashFamily](#)

[hiSetSplashProduct](#)

[hiSetSplashLicense](#)



## Cadence User Interface SKILL Reference

### Window Functions

---

## hiSetSplashIcon

```
hiSetSplashIcon(
 g_icon
)
=> t / nil
```

### Description

Sets a new icon for an application splash screen. The icon is displayed between the product name and the license information and always be set to a 48x48 size.

### Argument

|               |                                 |
|---------------|---------------------------------|
| <i>g_icon</i> | The icon for the splash screen. |
|---------------|---------------------------------|

### Value Returned

|     |                       |
|-----|-----------------------|
| t   | The icon was set.     |
| nil | The icon was not set. |

### ***Related Topics***

[Splash Screen Functions](#)

[hiAbout](#)

[hiSetSplashBackground](#)

[hiSetSplashDefaultBackground](#)

[hiSetSplashFamily](#)

[hiSetSplashProduct](#)

[hiSetSplashLicense](#)

## hiSetSplashFamily

```
hiSetSplashFamily(
 t_string
)
=> t / nil
```

### Description

Sets the product family name on an application's splash screen.

### Argument

|                 |                          |
|-----------------|--------------------------|
| <i>t_string</i> | The product family name. |
|-----------------|--------------------------|

### Value Returned

|     |                                      |
|-----|--------------------------------------|
| t   | The product family name was set.     |
| nil | The product family name was not set. |

### ***Related Topics***

[Splash Screen Functions](#)

[hiAbout](#)

[hiSetSplashBackground](#)

[hiSetSplashDefaultBackground](#)

[hiSetSplashIcon](#)

[hiSetSplashProduct](#)

[hiSetSplashLicense](#)

## hiSetSplashProduct

```
hiSetSplashProduct(
 t_string
)
=> t / nil
```

### Description

Sets the product name on an application's splash screen.

### Argument

|                 |                   |
|-----------------|-------------------|
| <i>t_string</i> | The product name. |
|-----------------|-------------------|

### Value Returned

|     |                               |
|-----|-------------------------------|
| t   | The product name was set.     |
| nil | The product name was not set. |

### ***Related Topics***

[Splash Screen Functions](#)

[hiAbout](#)

[hiSetSplashBackground](#)

[hiSetSplashDefaultBackground](#)

[hiSetSplashIcon](#)

[hiSetSplashFamily](#)

[hiSetSplashLicense](#)

## hiSetSplashLicense

```
hiSetSplashLicense(
 t_string
)
=> t / nil
```

### Description

Sets the license information in an application's splash screen. You can currently set only Cadence license information.

### Argument

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| <i>t_string</i> | The license information you want to display in the splash screen. |
|-----------------|-------------------------------------------------------------------|

### Value Returned

|            |                                        |
|------------|----------------------------------------|
| <i>t</i>   | The licensing information was set.     |
| <i>nil</i> | The licensing information was not set. |

### ***Related Topics***

[Splash Screen Functions](#)

[hiAbout](#)

[hiSetSplashBackground](#)

[hiSetSplashDefaultBackground](#)

[hiSetSplashIcon](#)

[hiSetSplashFamily](#)

[hiSetSplashProduct](#)

## Magnifier Functions

This section describes those functions that can be used for canvas magnification procedures at any time during a Virtuoso session.

These functions are:

- [hiGetMagnifierSize](#)
- [hiIsMagnifierAnchored](#)
- [hiIsMagnifierOpened](#)
- [hiResizeMagnifier](#)
- [hiScaleMagnifier](#)
- [hiToggleAnchorMagnifier](#)
- [hiToggleMagnifier](#)
- [hiToggleMagnifierResolution](#)
- [hiUpdateMagOptions](#)
- [hiUseMagnifierResolution](#)
- [hiUsingMagnifierResolution](#)
- [hiZoomInMagnifier](#)
- [hiZoomOutMagnifier](#)

See [Using the Magnifier](#) in the *Virtuoso Schematic Editor L User Guide*.

## hiGetMagnifierSize

```
hiGetMagnifierSize(
 w_window
)
=> size / nil
```

### Description

Returns a two element list for the currently opened magnifier for the specified graphics window.

### Argument

|                 |                      |
|-----------------|----------------------|
| <i>w_window</i> | The graphics window. |
|-----------------|----------------------|

### Values Returned

|             |                                            |
|-------------|--------------------------------------------|
| <i>size</i> | Two-element list detailing magnifier size. |
| <i>nil</i>  | Operation unsuccessful.                    |

### ***Related Topics***

[Magnifier Functions](#)

[hiIsMagnifierAnchored](#)

[hiIsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

[hiToggleMagnifierResolution](#)

## **hIsMagnifierAnchored**

```
hIsMagnifierAnchored(
 w_window
)
=> t / nil
```

### **Description**

Checks if the magnifier has been anchored inside the specified graphics window. If the magnifier has been anchored it and does not move with the mouse cursor.

### **Argument**

|                 |                      |
|-----------------|----------------------|
| <i>w_window</i> | The graphics window. |
|-----------------|----------------------|

### **Value Returned**

|     |                            |
|-----|----------------------------|
| t   | Magnifier is anchored.     |
| nil | Magnifier is not anchored. |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hIsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

[hiToggleMagnifierResolution](#)

## **hiIsMagnifierOpened**

```
hiIsMagnifierOpened(
 w_window
)
=> t / nil
```

### **Description**

Indicates whether a magnifier is currently opened in the specified window.

### **Argument**

|                 |                      |
|-----------------|----------------------|
| <i>w_window</i> | The graphics window. |
|-----------------|----------------------|

### **Value Returned**

|     |                                  |
|-----|----------------------------------|
| t   | Magnifier is currently open.     |
| nil | Magnifier is not currently open. |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hiIsMagnifierAnchored](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

[hiToggleMagnifierResolution](#)



## hiResizeMagnifier

```
hiResizeMagnifier(
 l_resizeBy
)
=> t / nil
```

### Description

Changes the magnifier size dynamically.

### Argument

|                   |                                                                                                                                                                                                                                                                         |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_resizeBy</i> | Resizes the frame of the magnifier by multiplying its current width and height by the setting given. For example, if <code>resizeBy</code> is set to 0.5, the area of the magnifier is quarter of its current size, as both sides of the magnifier are reduced by half. |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>t</code>   | Magnifier size has been successfully changed. |
| <code>nil</code> | Magnifier size not changed.                   |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

[hiToggleMagnifierResolution](#)

## hiScaleMagnifier

```
hiScaleMagnifier(
 n_scale
)
=> t / nil
```

### Description

Changes the zooming factor of the magnifier dynamically.

### Argument

|                |                                                                                                                                                                                                                                                                                       |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>n_scale</i> | A float argument that modifies the scale factor of the contents of the magnifier by the given scale argument. For example, a scale factor of 1.0 does not changes what is shown in the magnifier, while a scale factor of 0.5 modifies the geometries so that they are half the size. |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Value Returned

|     |                                              |
|-----|----------------------------------------------|
| t   | Magnifier size has been successfully scaled. |
| nil | Magnifier scale not changed.                 |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

[hiToggleMagnifierResolution](#)

## **hiToggleAnchorMagnifier**

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

### **Description**

Toggles the anchored state of the magnifier to anchored/un-anchored depending upon the current anchored state of the magnifier.

This function is enabled for windows where the root application type returned by `deGetRootType` is either "schematic", "maskLayout" or "schematicSymbol".

### **Argument**

None.

### **Value Returned**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>t</code>   | Magnifier anchored state successfully changed. |
| <code>nil</code> | Action failed.                                 |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hiIsMagnifierAnchored](#)

[hiIsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleMagnifier](#)

[hiToggleMagnifierResolution](#)

## hiToggleMagnifier

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

### Description

Changes the visibility of the magnifier to on/off depending on the current visibility setting of the magnifier.

This is enabled for windows where the root application type returned via `deGetRootType` is either "schematic", "maskLayout" or "schematicSymbol".

### Argument

None.

### Value Returned

|     |                                                      |
|-----|------------------------------------------------------|
| t   | Magnifier visibility successfully changed to on/off. |
| nil | Action failed.                                       |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifierResolution](#)

## **hiToggleMagnifierResolution**

```
hiToggleMagnifierResolution()
=> t
```

### **Description**

Turns on magnifier resolution in the canvas if the canvas is not currently using magnifier resolution. Otherwise turns off magnifier resolution.

### **Arguments**

None.

### **Value Returned**

|   |                                                   |
|---|---------------------------------------------------|
| t | Magnifier resolution successfully toggled on/off. |
|---|---------------------------------------------------|

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

## hiUpdateMagOptions

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

### Description

Update and display the Magnifier Options form with the magnifier values that are being used in the current window.

### Argument

None (uses current window).

### Value Returned

|     |                                                                                                          |
|-----|----------------------------------------------------------------------------------------------------------|
| t   | Magnifier Options form successfully displayed and updated to use magnifier values in the current window. |
| nil | Magnifier options form not updated.                                                                      |

### Example

```
hiUpdateMagOptions ()
```

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

## hiUseMagnifierResolution

```
hiUseMagnifierResolution(
 w_window
 g_use
)
=> t / nil
```

### Description

Alters the movement of the cursor within the canvas.

As a use model, it takes a larger mouse motion to move between objects under the magnifier, compared to when those same objects are not currently magnified in the canvas.

Therefore, when turned on, `hiUseMagnifierResolution` makes the mouse movement in the canvas behave as if the mouse is in the magnifier, thereby enabling fine resolution of mouse motion between geometries in the canvas.

Turning this feature on slows down the motion of the mouse in the canvas, as determined by the current scale factor in the magnifier.

This feature does not work correctly if you are using VNC, or any other thin client, due to bad, or non-existent, mouse-warping. The result of which leads to a drifting apart of the mouse and tracking cursors.

### Arguments

|                       |                                                                                                                                                                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_window</code> | The graphics window.                                                                                                                                                                                                                                       |
| <code>g_use</code>    | Use to set higher mouse movement precision.<br><br>To move the mouse cursor precisely by a single pixel, in the magnifier space of resolution, set <code>g_use</code> to <code>t</code> . To cancel this mode set <code>g_use</code> to <code>nil</code> . |

### Values Returned

|                  |                                                                             |
|------------------|-----------------------------------------------------------------------------|
| <code>t</code>   | Magnifier size has been successfully set use magnifier resolution settings. |
| <code>nil</code> | Action failed.                                                              |

## Cadence User Interface SKILL Reference

### Window Functions

---

#### ***Related Topics***

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)



## **hiUsingMagnifierResolution**

```
hiUsingMagnifierResolution(
 w_window
)
=> t / nil
```

### **Description**

Returns whether mouse movement in the canvas is set to behave as if the mouse is located within the magnifier.

If it is, it requires more movement of the mouse between objects and areas in the canvas.

### **Argument**

|                 |                                                  |
|-----------------|--------------------------------------------------|
| <i>w_window</i> | The graphics window that contains the magnifier. |
|-----------------|--------------------------------------------------|

### **Values Returned**

|            |                                            |
|------------|--------------------------------------------|
| <i>t</i>   | Mouse movement set to be within magnifier. |
| <i>nil</i> | Mouse movement not set within magnifier.   |

### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier,](#)

[hiToggleMagnifier](#)

## Cadence User Interface SKILL Reference

### Window Functions

---

#### hiZoomInMagnifier

```
hiZoomInMagnifier(
 w_window
)
=> t / nil
```

#### Description

Zooms into the picture in canvas, as opposed to the picture in the magnifier, with the current magnifier zooming factor.

#### Argument

|                 |                      |
|-----------------|----------------------|
| <i>w_window</i> | The graphics window. |
|-----------------|----------------------|

#### Values Returned

|            |                                                    |
|------------|----------------------------------------------------|
| <i>t</i>   | Picture in canvas has been successfully zoomed in. |
| <i>nil</i> | Picture in canvas has not been zoomed in on.       |

#### ***Related Topics***

[Magnifier Functions](#)

[hiGetMagnifierSize](#)

[hilsMagnifierAnchored](#)

[hilsMagnifierOpened](#)

[hiResizeMagnifier](#)

[hiScaleMagnifier](#)

[hiToggleAnchorMagnifier](#)

[hiToggleMagnifier](#)

## hiZoomOutMagnifier

```
hiZoomOutMagnifier(
 w_window
)
=> t / nil
```

### Description

Backs out one call to the `hiZoomInMagnifier` command from the stack, similar to an undo action.

### Argument

|                       |                      |
|-----------------------|----------------------|
| <code>w_window</code> | The graphics window. |
|-----------------------|----------------------|

### Values Returned

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <code>t</code>   | Picture in canvas has been successfully zoomed (backed) out. |
| <code>nil</code> | Picture in canvas has not been zoomed (backed) out from.     |

### ***Related Topics***

Magnifier Functions

hiGetMagnifierSize

hilsMagnifierAnchored

hilsMagnifierOpened

hiResizeMagnifier

hiScaleMagnifier

hiToggleAnchorMagnifier

## **Cadence User Interface SKILL Reference**

### Window Functions

---

---

## Window Banner Functions

---

This topic provides a list of Cadence® SKILL functions associated with window banner. Menus are added to the menu bar and labels are added to the banner bar of a window.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiChangeBannerLabel](#)
- [hiDeleteBannerLabel](#)
- [hiDeleteBannerMenu](#)
- [hiDeleteBannerMenus](#)
- [hiGetBannerLabels](#)
- [hiGetBannerMenus](#)
- [hiGetNumMenus](#)
- [hiInsertBannerMenu](#)
- [hiIsMenuSlotFilled](#)
- [hiReplaceAllBannerMenus](#)

## hiChangeBannerLabel

```
hiChangeBannerLabel(
 w_windowId
 t_bannerLabel
 x_labelPosition
)
=> t / nil
```

### Description

Overwrites an existing label or adds a new label in the banner bar of a window.

### Arguments

|                        |                                                                                                                                                                                                                                                                                                                                                          |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>      | Window to which you want to add the label.                                                                                                                                                                                                                                                                                                               |
| <i>t_bannerLabel</i>   | Specifies the label.                                                                                                                                                                                                                                                                                                                                     |
| <i>x_labelPosition</i> | Integer specifying the position for the label. 0 represents the leftmost user-defined label, 1 the one to its right, and so on. If a label already exists at the position you specify, it is overwritten. Otherwise, if <i>x_labelPosition</i> is greater than the current number of labels, the new label is added to the right of the existing labels. |

### Values Returned

|            |                                                           |
|------------|-----------------------------------------------------------|
| <i>t</i>   | The label was added.                                      |
| <i>nil</i> | The label was not added. An error message is also issued. |

### Examples

Adds Label 1 in the banner bar of hiGetCurrentWindow.

```
hiChangeBannerLabel(hiGetCurrentWindow() "Label 1" 0)
=> t
```

### Related Topics

[Window Banner Functions](#)

## **Cadence User Interface SKILL Reference**

### **Window Banner Functions**

---

hiDeleteBannerLabel

hiGetBannerLabels

## hiDeleteBannerLabel

```
hiDeleteBannerLabel(
 w_windowId
 x_labelPosition
)
=> t / nil
```

### Description

Deletes a label from the banner bar of a window.

### Arguments

|                        |                                                                                                                                                                                                                                                                                                                     |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>      | Window from which you want to delete the label.                                                                                                                                                                                                                                                                     |
| <i>x_labelPosition</i> | Integer specifying the position of the label that you want to delete. 0 represents the leftmost user-defined label, 1 the one to its right, and so on. If the label being deleted was the only label on the banner bar and there is no application data associated with the window, the banner bar is also removed. |

### Values Returned

|     |                                                             |
|-----|-------------------------------------------------------------|
| t   | The label was deleted.                                      |
| nil | The label was not deleted. An error message is also issued. |

### Examples

Deletes a label from banner bar of `hiGetCurrentWindow`.

```
hiDeleteBannerLabel(hiGetCurrentWindow() 0)
=> t
```

### ***Related Topics***

[Window Banner Functions](#)

[hiChangeBannerLabel](#)

[hiGetBannerLabels](#)



## hiDeleteBannerMenu

```
hiDeleteBannerMenu(
 w_windowId
 x_menuPosition
)
=> t / nil
```

### Description

Deletes a menu from the menu bar of a window.

### Arguments

|                       |                                                                                                                                                                      |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>     | Window from which to delete the menu.                                                                                                                                |
| <i>x_menuPosition</i> | Integer specifying the position of the menu that you want to delete. 0 represents the leftmost user-defined menu on the menu bar, 1 the one to its right, and so on. |

### Values Returned

|     |                                                           |
|-----|-----------------------------------------------------------|
| t   | The menu is deleted.                                      |
| nil | The menu is not deleted. An error message is also issued. |

### Examples

Deletes the left-most menu from the menu bar of `hiGetCurrentWindow`.

```
hiDeleteBannerMenu(hiGetCurrentWindow() 0)
=> t
```

### Related Topics

[Window Banner Functions](#)

## Cadence User Interface SKILL Reference

### Window Banner Functions

---

#### hiDeleteBannerMenus

```
hiDeleteBannerMenus(
 w_windowId
)
=> t / nil
```

#### Description

Deletes all menus from the menu bar of a window.

#### Argument

|                         |                                        |
|-------------------------|----------------------------------------|
| <code>w_windowId</code> | Window from which to delete all menus. |
|-------------------------|----------------------------------------|

#### Values Returned

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>t</code>   | The menus are deleted.                                      |
| <code>nil</code> | The menus are not deleted. An error message is also issued. |

#### Examples

Deletes all menu bar from window (3).

```
hiDeleteBannerMenus(window(3))
=> t
```

#### ***Related Topics***

[Window Banner Functions](#)

## Cadence User Interface SKILL Reference

### Window Banner Functions

---

#### hiGetBannerLabels

```
hiGetBannerLabels(
 w_windowID
)
=> l_labels / nil
```

#### Description

Returns the list of labels that are in the banner bar of a window. (These labels were added with the `?labels` argument to the `hiOpenWindow` function when the window was created or with the `hiChangeBannerLabel` function.)

#### Argument

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>w_windowID</code> | Window from which you want to retrieve labels. |
|-------------------------|------------------------------------------------|

#### Value Returned

|                       |                                                               |
|-----------------------|---------------------------------------------------------------|
| <code>l_labels</code> | List of labels in the banner bar of the window you specified. |
| <code>nil</code>      | There are no labels in the banner bar.                        |

#### ***Related Topics***

[Window Banner Functions](#)

[hiChangeBannerLabel](#)

[hiDeleteBannerLabel](#)

## hiGetBannerMenus

```
hiGetBannerMenus (
 w_windowId
)
=> l_menuSymbols / nil
```

### Description

Returns a list of menus available in the menu bar of a window.

### Argument

|                   |                                            |
|-------------------|--------------------------------------------|
| <i>w_windowId</i> | Window from which the menus are retrieved. |
|-------------------|--------------------------------------------|

### Values Returned

|                      |                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_menuSymbols</i> | A list of all menu symbols contained in the menu bar. The menu symbols are returned in the order in which they appear on the menu bar, from left to right. |
| <i>nil</i>           | There are no menus in the menu bar.                                                                                                                        |

### Examples

Returns a list of menus available in hiGetCIWindow.

```
hiGetBannerMenus(hiGetCIWindow ())
=> (hiOpenMenu hiDMMMenu tcMenu hiPlotMenu hiFrameworkMenu transMenu)
```

### Related Topics

[Window Banner Functions](#)

## hiGetNumMenus

```
hiGetNumMenus (
 w_windowId
)
=> x_numMenus
```

### Description

Returns the number of menus in the menu bar of a window.

### Argument

|                   |                |
|-------------------|----------------|
| <i>w_windowId</i> | The window ID. |
|-------------------|----------------|

### Values Returned

|                   |                                    |
|-------------------|------------------------------------|
| <i>x_numMenus</i> | The number of menus in the window. |
|-------------------|------------------------------------|

### Examples

Returns the number of menus in the menu bar of `hiGetWindow`.

```
hiGetNumMenus (hiGetCIWindow ())
=> 6
```

### ***Related Topics***

[Window Banner Functions](#)

## Cadence User Interface SKILL Reference

### Window Banner Functions

---

#### hiInsertBannerMenu

```
hiInsertBannerMenu(
 w_windowId
 g_menu
 x_menuPosition
)
=> t / nil
```

#### Description

Inserts a menu into the menu bar of a window. If the menu symbol that you specify already exists on the window, a warning message is issued and the menu is not placed on the window.

There is a maximum number of 40 window banner menus that can be added.

#### Arguments

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>     | Specifies the window to add a menu.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <i>g_menu</i>         | Menu that you want to insert. The menu must be the value returned from <code>hiCreatePulldownMenu</code> , or a symbol representing that value. The title of this menu is displayed as text on the menu bar.                                                                                                                                                                                                                                                                                                                                                                                          |
| <i>x_menuPosition</i> | <p>Location in the banner where the menu is to be positioned. The menu positions are numbered from 0, starting with the leftmost user-defined menu contained in the menu bar.</p> <p>This function does not overwrite an existing menu. If a menu already exists at the position you specify, the function inserts the new menu into the specified position and moves the existing menus to the right. If <i>x_menuPosition</i> is greater than the current number of user-defined menus, the function adds the new menu to the right of all other menus (except the <i>Help</i> button or menu).</p> |

## Cadence User Interface SKILL Reference

### Window Banner Functions

---

#### Values Returned

|     |                                                            |
|-----|------------------------------------------------------------|
| t   | The menu is inserted in the menu bar.                      |
| nil | The menu is not inserted. An error message is also issued. |

#### Examples

Inserts `hiCreatePulldownMenu` into the menu bar of the window.

```
menu=hiCreatePulldownMenu(...)
hiInsertBannerMenu(window menu 0)
=> t

hiInsertBannerMenu(window hiCreatePulldownMenu(...) 0)
=> t
```

The following example inserts a pull-down menu in the leftmost position of the CIW.

```
hiInsertBannerMenu(window(1) trPulldownMenu 0)
```

#### ***Related Topics***

[Window Banner Functions](#)

[hiCreatePulldownMenu](#)

## hiIsMenuSlotFilled

```
hiIsMenuSlotFilled(
 w_windowId
 x_position
)
=> t / nil
```

### Description

Checks whether a menu already exists in the specified position in the menu bar of a window.

### Arguments

|                   |                                                                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | The window ID.                                                                                                                                    |
| <i>x_position</i> | Integer specifying the position in the menu bar that you want to check. Positions on the menu bar are numbered from left to right, starting at 0. |

### Value Returned

|            |                                                 |
|------------|-------------------------------------------------|
| <i>t</i>   | The specified position contains a menu.         |
| <i>nil</i> | The specified position does not contain a menu. |

### Examples

Checks whether a menu already exists in the 5th position in the menu bar of `hiGetCIWindow`.

```
hiIsMenuSlotFilled(hiGetCIWindow() 5)
=> t
```

### Related Topics

[Window Banner Functions](#)



## hiReplaceAllBannerMenus

```
hiReplaceAllBannerMenus(
 w_windowId
 l_menulist
)
=> t / nil
```

### Description

Replaces all menus in a window with new menus.

There is a maximum number of 40 window banner menus that can be added.

### Arguments

|                   |                                                                                                                                                                                                            |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window in which you want to replace menus.                                                                                                                                                                 |
| <i>l_menulist</i> | List of menus that will replace the existing menus. These menus must be listed in the order in which you want them to be placed. Each menu must have been created with <code>hiCreatePulldownMenu</code> . |

### Values Returned

|            |                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The menus are replaced.                                                                                            |
| <i>nil</i> | <i>w_windowId</i> or any of the menus in the list is invalid, or the same menu appears more than once in the list. |

### Related Topics

[Window Banner Functions](#)

[hiCreatePulldownMenu](#)

## **Cadence User Interface SKILL Reference**

### Window Banner Functions

---

---

## Bindkey Functions

---

This topic provides a list of Cadence® SKILL functions associated with Bindkeys.

You can use SKILL functions to automate repetitive tasks by tying commands to short key strokes or key stroke combinations.

You can tie commands to single keys, key combinations, or mouse sequences. After you have defined a key or key combination to execute a certain command or a list of commands, you can use this bindkey to speed up your work. Binding keys to a task or a whole series of tasks also reduces typing mistakes.

The application name is used as the bindkey prefix and should be registered with the Design Editor via the `deRegApp` function. If you need to define the bindkeys for a non-Design Editor application (one that is not registered through `deRegApp`), register the application name with the `hiRegisterBindKeyPrefix` function.

For performance reasons, a group of application names can share the same set of bindkeys through bindkey inheritance. One application name can be an alias and inherit bindkeys using the `hiInheritBindKey` call from another application that has bindkeys defined.

Bindkeys may be defined for alias applications, also known as sub-applications or sub-apps that will override those bindkeys inherited from the root application. Only one level of inheritance is permitted however, and an alias application cannot inherit from another alias application. In addition, bindkeys can also be defined for plugins, which will override the bindkeys for the root application and/or alias application. If multiple plugins are installed on the same window, the bindkeys for the most recently added plugin will override the bindkeys of the previously added plugin.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiSetBindKey](#)
- [hiSetBindKeys](#)
- [hiRegSetBindKeyProc](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

- [hiGetBindKey](#)
- [hiGetBindKeys](#)
- [hiShowBindKeys](#)
- [hiShowBindKeysByAppType](#)
- [hiShowBindKeysByWindow](#)
- [hiRegisterBindKeyPrefix](#)
- [hiInheritBindKey](#)
- [hiGetBindKeyInheritRoot](#)
- [hiGetBindKeyInheritAlias](#)
- [hiGetBindKeyPrefixList](#)
- [hiConfigureBindKeys](#)
- [hiEnableAsstBindKeyPrefix](#)
- [hiDisableAsstBindKeyPrefix](#)
- [hiGetDisabledAsstBindKeyPrefixes](#)
- [hiPrintBindKeyUsage](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiBindKeyModifiers

```
hiBindKeyModifiers(
 [s_type]
)
=> l_modifiers
```

#### Description

Returns the list of modifiers that can be used with bindkeys in the current session. You can specify the type of keys for which you want the list of legal modifiers: printable, mouse, or function.

By default `Alt` cannot be used as a modifier with any printable characters (alphanumeric and special characters) because it is used as a modifier for menu access keys. `Alt` can be used with function keys and for mouse bindings. However, if menu access keys are disabled, `Alt` is a legal modifier for all keys, including printable characters.

To enable `Alt` for printable characters in bindkeys, you need to disable menu access keys.

If a window manager or desktop, such as CDE, predefines an `Alt+key` or `Alt+mouseButton` binding, then that combination will not work as a menu access key or bindkey on that system.

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### Argument

*s\_type*

The type of keys for which to return legal modifiers. One of the following symbols:

- `'printable`: For printable ASCII characters: alphanumeric and special characters
- `'mouse`: For mouse buttons (click, drag, double-click)
- `'function`: For function keys. The following keys are considered function keys:

Any key whose name begins with `F#` or `KP_` (for keypad keys) or `osf` (for OSF virtual key binding names)

Support of Xt/Motif will be removed from a future Virtuoso release. Resultantly, the use of OSF virtual key names will no longer be possible, as that facility is specific to Xt/Motif. They should therefore be avoided.

Any of the following keys: `Activate`, `AddMode`, `Again`, `Back`, `BackSpace`, `BackTab`, `Begin`, `BeginLine`, `Break`, `Cancel`, `Clear`, `Copy`, `Cut`, `Del`, `Delete`, `Down`, `End`, `EndLine`, `Enter`, `Esc`, `Escape`, `Find`, `Forward`, `Front`, `Help`, `Home`, `Ins`, `Insert`, `Left`, `Menu`, `MenuBar`, `Next`, `Open`, `PageDown`, `PageLeft`, `PageRight`, `PageUp`, `Paste`, `Pause`, `PgDn`, `PgUp`, `PrimaryPaste`, `Print`, `PrintScreen`, `Prior`, `Props`, `QuickPaste`, `Redo`, `Refresh`, `Return`, `Right`, `ScrollDown`, `ScrollUp`, `Select`, `Space`, `Stop`, `SysReq`, `Tab`, `Undo`, `Up`

#### Values Returned

*l\_modifiers*

List of modifiers that can be used with the type of keys specified.

#### Examples

Returns the list of modifiers for the `'printable` keys.

```
hiBindKeyModifiers('printable)
```

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

```
=> ("Shift" "Ctrl" "Meta")
```

Returns the list of modifiers for the 'function keys.

```
hiBindKeyModifiers('function)
```

```
=> ("Shift" "Ctrl" "Alt" "Meta")
```

### ***Related Topics***

[Bindkey Functions](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiSetBindKey

```
hiSetBindKey(
 [t_applicationType]
 [t_key]
 [t_skill_cmd]
)
=> t / nil
```

#### Description

Binds a SKILL command string to a keyboard key or a mouse sequence for an application.

Before setting bindkeys, the application should be registered either through `deRegApp` or `hiRegisterBindKeyPrefix`. Cadence applications are already registered. You can get a list of registered bindkey prefixes for applications that you are running with the `hiGetBindKeyPrefixList` function or the Key or Mouse Binding form (choose *Options – Bindkeys* in the CIW to get the form).

If the *t\_applicationType* or *t\_key* arguments are not specified, a bindkey form is displayed.

#### Arguments

|                          |                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_applicationType</i> | Any user-definable string that has been registered through <code>hiRegisterBindKeyPrefix</code> or a Cadence application type. You can get a list of registered bindkey prefixes for applications that you are running with the <code>hiGetBindKeyPrefixList</code> function or the Key or Mouse Binding form.<br><br>If this argument is not specified, a bindkey form is displayed. |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## Cadence User Interface SKILL Reference

### Bindkey Functions

---

*t\_key*

String specifying the key sequence, the operator, detail, and occurrences. If not specified, a bindkey form is displayed. Items in this list are:

- *modifier\_list* (optional)
- *operator*
- *detail* (optional)
- *occurrences* (optional)

If *t\_key* ends with EF, the SKILL command is used in `enterfunction` mode. Otherwise, it is a `non-enterfunction` mode command. If there is no `enterfunction` mode command defined when a key or mouse event happens in `enterfunction` mode, the `non-enterfunction` mode command for this key is used.

Defining an empty string for a command is the equivalent to not defining the command in the first place. A bindkey defined for an alias application that has an empty string for the command, will inherit from the root application. If an `enterfunction` command is defined as an empty string, then the `non-enterfunction` command will be used.

When using an `enterfunction` in an alias application, the `enterfunction` binding for the alias application will be checked first, then, if that is not defined, the `enterfunction` binding for the root application will be checked. If that is not defined, then the `non-enterfunction` binding for the alias application is checked, finally followed by the `non-enterfunction` binding for the root application.

For more information, see Additional Information.

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

*t\_skill\_cmd*

SKILL function to be executed with the translation in *t\_key*. If you wish to specify more than one function, separate each one with a space. If a function is not specified, a bindkey form is displayed.

The maximum string length for the SKILL command is 511 characters. If multiple commands are to be executed, it is recommended to define a SKILL procedure and call the SKILL procedure rather than coding the entire procedure into the bindkey command string.

To unset a previously set bindkey, you can set the *t\_skill\_cmd* to ". This is the equivalent to not defining a command in the first place.

### Examples

The following are examples of various key bindings:

```
"Shift<Key>F2" (Shift key + F2 key)
"Ctrl Shift<Key>Escape" (Control key + Shift key + Esc key)
"<Key>3" (3 key)
```

When setting a bindkey to the comma key, spell out the word comma rather than using the comma character (,).

```
"<Key>comma"
"Shift<Key>comma"Value Returned"
```

Binds the Command Interpreter application with `hiQuit` using the bindkey `None<Key>F6`:

```
hiSetBindKey("Command Interpreter" "None<Key>F6"
"hiQuit()")
=> t
```

Binds the Schematics application with `hiZoomIn` using the bindkey `Shift<Key>z`:

```
hiSetBindKey("Schematics" "Shift<Key>z"
"hiZoomIn(hiGetCurrentWindow())")
=> t
```

Binds the Schematics application with `hiEditPropList` using the bindkey `Shift<Key>s`:

```
hiSetBindKey("Schematics" "Shift Ctrl<Key>s"
"{println(\"Editing selected object \")
hiEditPropList(car(getSelectedSet()))}")
=> t
```

Binds the Schematics application with `mouseAddPt` using the bindkey `None<Btn1Down>EF`:

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

```
hiSetBindKey("Schematics" "None <Btn1Down> EF"
 "mouseAddPt()")
=> t
```

Binds the Schematics application with `hiZoomIn` using the bindkey `<DrawThru3>`:

```
hiSetBindKey("Schematics" "<DrawThru3>" "hiZoomIn()")
=> t
```

Binds the Schematics application with `hiUnselectTextAll` using the bindkey `<Key>F8`:

```
hiSetBindKey("Show File" "<Key>F8"
 "hiUnselectTextAll(hiGetCurrentWindow())")
=> t
```

### Additional Information

To define no action for a binding and stop inheritance, specify the string `nil` for `t_key`.

Use `t_key` in the following syntax:

```
"modifier_list<operator>detail(occurrences) "
```

Use the `hiBindKeyModifiers` functions to get a list of legal modifiers for your session. The following modifiers could be in that list:

- `Ctrl`: Control modifier bit
- `Shift`: Shift modifier bit
- `Meta`: Meta key modifier
- `Hyper`: Hyper key modifier
- `Super`: Super key modifier
- `Alt`: Alt key modifier (legal only for mouse bindings and function keys, not for printable characters)
- `None`: No modifiers

All modifiers except `Ctrl` and `Shift` will be checked to ensure they are assigned to keys on the current keyboard. A warning message will be issued, and the bindkey will not be defined for any modifier that is not assigned to a keyboard key. The `Ctrl` and `Shift` modifiers are guaranteed to be assigned. The Caps Lock key cannot be specified as a modifier. `Alt` can be specified only with function keys and mouse bindings, not with printable characters (unless menu access keys are disabled.)

An exclamation point or a tilde is illegal in a bindkey string.

If `None` is specified, it means no modifiers can be asserted. This is also the equivalent of having no modifiers.

```
None <event> detail
```

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

<event> detail

These following operators are supported by *t\_key*:

- Btn1Down: Button 1 Press
- Btn2Down: Button 2 Press
- Btn3Down: Button 3 Press
- Btn4Down: Mouse scroll wheel up
- Btn5Down: Mouse scroll wheel down
- Btn4Down(2) and Btn5Down(2) is ignored.
- Key: Key Press
- DrawThru: Button 1 Press and Motion
- DrawThru2: Button 2 Press and Motion
- DrawThru3: Button 3 Press and Motion

When performing a DrawThru operation, keyboard keys will be ignored except for the following keys which are hardcoded to the listed commands during DrawThru operations:

- Esc - which cancels the command.
- Arrow Keys - which pan by 20% in the appropriate direction.
- Z - which zooms in at the mouse.
- Shift+Z - which zooms out at the mouse.

### ***Related Topics***

[Bindkey Functions](#)

[hiBindKeyModifiers](#)

[hiGetBindKeyPrefixList](#)

[hiRegisterBindKeyPrefix](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

## hiSetBindKeys

```
hiSetBindKeys (
 t_applicationType
 l_bindKeyList
)
=> t / nil
```

### Description

Sets multiple bindkeys for an application at one time. For memory and performance efficiency, use this function instead of `hiSetBindKey` if you are setting multiple bindkeys. For information about restrictions, see `hiSetBindKey`.

### Arguments

*t\_applicationType* Any user-definable string that has been registered through `hiRegisterBindKeyPrefix` or a Cadence application type. You can get a list of registered bindkey prefixes for applications that you are running with the `hiGetBindKeyPrefixList` function or the Key or Mouse Binding form.

If this argument is not specified, a bindkey form is displayed.

*l\_bindKeyList* The list of bindkey lists. Each list is a bindkey setting with the following format:

```
list(
 t_key
 t_skill_cmd
 [t_EFskill_cmd]
)
```

*t\_EFskill\_cmd* is the same as *t\_skill\_cmd* when the *t\_key* argument ends with `enterfunction` mode. Do not add EF to the end of the *t\_key* argument.

The only way to invalidate a command is to replace it with another command or with an empty string. If you set the *t\_skill\_cmd* argument to `nil` when a non-`enterfunction` command is already defined for the key, it will not be changed. This rule also applies to *t\_EFskill\_cmd* and `enterfunction` commands if they are `nil` or if they are not passed.

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### Values Returned

|                  |                       |
|------------------|-----------------------|
| <code>t</code>   | The bindkeys are set. |
| <code>nil</code> | There was an error.   |

#### Additional Information

To set multiple bindkeys, use the `hiSetBindKeys()` function, which is much more efficient than multiple calls to `hiSetBindKey()`.

The following restrictions apply to setting bindkeys:

- When specifying a bindkey, you must ensure that it is spelt correctly and in the correct case.

Normally, the keyname should be spelt as it appears on the keyboard. One method to ensure that you are using the correct name is to look at the file: `/usr/include/X11/keysymdef.h`

Warnings due to incorrectly spelt keynames will appear in the `CDS.log` file.

- Do not assign the following printable keys to the CIW or to an encapsulation window:

- ❑ a through z, A through Z, or 0 through 9
- ❑ ! @ # \$ % ^ & \* ( ) \_ - + = | \ ~ ' { [ ] : ; " ' < , > . ? /

If you assigned any of the above keys accidentally, the key can be restored by specifying an empty string for `t_skill_cmd`. If the key was modified in the CIW, use the bindkey form to restore the behavior, since you will not be able to type the key you are trying to restore in the CIW anymore.

- Do not use the left or middle mouse buttons for setting bindings in the CIW and encapsulation windows. Setting bindings for these buttons conflicts with their normal actions.
- Do not assign anything to F3 or F4 function keys. Cadence software uses the F3 key to display options forms and the F4 key for partial selection in graphics windows.
- Do not assign anything to the following keys, depending on your window manager:

Motif and OpenLook window managers use the F1 key for help and the F10 key for highlighting menus. F1 key is usually only used for Help if no explicit Help key exists on the keyboard.

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

- Sun Solaris reserves the F11 and F12 function keys, and they cannot be overridden. This restriction might apply to other platforms as well.
- The unmodified F10 function key is used for `osfMenuBar` and cannot be overridden on Sun Solaris. This restriction might apply to other platforms as well.

In future releases, those keys with names that begin with "osf" may no longer be used. Therefore, their use should be avoided.

- By default, `Alt` cannot be used as a modifier with any printable characters (alphanumeric and special characters) because it is used as a modifier for menu access keys. `Alt` can be used with function keys and for mouse bindings. However, if menu access keys are disabled, `Alt` is a legal modifier for all keys, including printable characters.
- If a window manager or desktop, such as CDE, predefines an `Alt+key` or `Alt+mouseButton` binding, then that combination will not work as a menu access key or bindkey on that system.
- Bindkeys are not supported in form windows.

Bindkey customizations cannot be used in Layout Viewer. Any updates to the existing bindkeys or user-defined bindkeys are available only in Layout XL and Layout EXL . If you try to customize a bindkey for Layout Viewer, a one-time warning is displayed to inform you that these bindkeys are not available in Layout Viewer when Layout Viewer is opened. You can set `noWarnOnLVBindKeyCustomization` to `t` to disable this warning message.

### ***Related Topics***

[Bindkey Functions](#)

[hiBindKeyModifiers](#)

[hiGetBindKeyPrefixList](#)

[hiRegisterBindKeyPrefix](#)

[hiSetBindKey](#)

[noWarnOnLVBindKeyCustomization](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiRegSetBindKeyProc

```
hiRegSetBindKeyProc(
 t_application
 S_function
 [g_notInherited]
)
=> t / nil
```

#### Description

Registers a SKILL function that will be called after a bindkey is set for the specified application with the `hiSetBindKey` function or the CIW Key or Mouse Binding form, or after a set of bindkeys is defined with the `hiSetBindKeys` function.

If there are any errors while setting a bindkey with `hiSetBindKey`, the function will not be called. If an error prevents `hiSetBindKeys` from setting any bindkeys, the function will not be called; however, errors during the setting of individual keys will not prevent the function from being called. The function will be called even if no change occurred as a result of calling `hiSetBindKey` or `hiSetBindKeys`.

If you are registering an application that inherits bindkeys from another application, you should also register the function with the root application that it inherits them from. This will allow you to view all changes made to the bindkeys that apply to the alias application.

You can use the `hiGetBindKeyInheritRoot` function to get the base application.

#### Arguments

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_application</i> | Application prefix registered with <code>hiRegisterBindkeyPrefix</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <i>S_function</i>    | <p>Name or symbol of the SKILL function to be called whenever a bindkey is set for the application.</p> <p>The function will be passed two arguments. The first argument is a string specifying the bindkey prefix. The second argument is a list of the bindkeys that were updated (each item in the list is a list containing the bindkey and the command it was set to, in the same format as the <code>l_bindkeyList</code> argument of <code>hiSetBindKeys</code>). The return value of the function will be ignored.</p> |



## Cadence User Interface SKILL Reference

### Bindkey Functions

---

*g\_notInherited*

If this argument is specified, and it not set as `nil` and the `applicationType` specified is an alias, then inherited commands will not be returned.

If the argument is not specified, or it is `nil`, then the command returned may be inherited from the root application.

#### Values Returned

`t`

The function was registered.

`nil`

*t\_application* is not registered, *t\_application* inherits from another application prefix, or *S\_function* is not a valid name or symbol. A warning is also displayed.

#### ***Related Topics***

[Bindkey Functions](#)

[hiRegisterBindKeyPrefix](#)

[hiGetBindKeyInheritRoot](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiGetBindKey

```
hiGetBindKey(
 [t_applicationType]
 [t_key]
)
=> t_skill_cmd / nil
```

#### Description

Returns the SKILL command string bound to a key or mouse button for an application.

If *t\_applicationType* or *t\_key* is not specified, a bindkey form is displayed.

#### Arguments

*t\_applicationType* Any user-type string that has been registered using `deRegApp` or `hiRegisterBindKeyPrefix`. If not specified, a bindkey form is displayed. The application type can be one of the following strings:

- `Command Interpreter: Command Interpreter Window (CIW)`
- `Schematics: Schematics Window`
- `Show File: Show File Window`
- `wa: Waveform Window`

*t\_key* String specifying the key sequence, the operator, detail, and occurrences. If not specified, a bindkey form is displayed. Items in this list are:

- `modifier_list` (optional)
- `operator`
- `detail` (optional)
- `occurrences` (optional)

If *t\_key* ends with “EF”, the SKILL function is used in `enterfunction` mode. If not, it is a `non-enterfunction` mode command.

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### Values Returned

|                    |                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>t_skill_cmd</i> | The associated SKILL function string if one is associated with the specified application type and key sequence. |
| <i>nil</i>         | <i>t_applicationType</i> or <i>s_key</i> is invalid. A warning is also issued.                                  |

#### Examples

Display the Bindkey Editor form.

```
hiGetBindkey("Layout")
```

The following examples retrieve information about different bindkeys in layout.

```
hiGetBindKey("Layout" "<Key>z")
=> "hiZoomIn()"
```

```
hiGetBindKey("Layout" "<Key>Up")
=> "geScroll(nil \"n\" nil)"
```

```
hiGetBindKey("Layout" "Ctrl<Key>J")
=> "leHiFlip()"
```

```
hiGetBindKey("Layout" "Ctrl Shift <Key>J")
=> "drdCompactSelSet('right)'"
```

```
hiGetBindKey("Layout" "Ctrl Shift <Btn1Down>") ; Btn1Down - button of mouse
=> "hiZoomInMagnifier(hiGetCurrentWindow())"
```

#### ***Related Topics***

#### Bindkey Functions

#### hiRegisterBindKeyPrefix

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiGetBindKeys

```
hiGetBindKeys (
 t_applicationType
 [g_alsoInherited]
)
=> l_bindkeyList / nil
```

#### Description

Returns the list of bindkeys that are set for an application.

#### Arguments

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_applicationType</i> | Application types that have been registered using either <code>hiRegisterBindKeyPrefix</code> or <code>deRegApp</code> .                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>g_alsoInherited</i>   | <p>If this argument is specified, and it is not <code>nil</code> nor is the <code>applicationType</code> specified an alias, then the returned list will also include bindkeys that have been defined for the root application.</p> <p>The inherited keys are listed first. Therefore, if a key appears twice in the list, then the second listed will be the one that is defined for the alias application. Otherwise, there is no way to distinguish one from the other if <code>g_alsoInherited</code> is not <code>nil</code>.</p> |

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### Value Returned

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_bindkeyList</i> | <p>A list, each element of which is a list of the following form:</p> <pre>list(<i>t_key</i> <i>t_function</i>/nil [<i>t_EFfunction</i>])</pre> <p>where:</p> <ul style="list-style-type: none"><li>■ <i>t_key</i> is a string specifying the key sequence, such as "Ctrl&lt;Key&gt;w".</li><li>■ <i>t_function</i> is a string specifying the SKILL function to which the key is bound. If the key is not bound to any function but does have an enterfunction binding, <i>t_function</i> is nil.</li><li>■ <i>t_EFfunction</i> is an optional string specifying the enterfunction binding for the key.</li></ul> |
| nil                  | <p>There are no bindkeys set for <i>t_applicationType</i>, or <i>t_applicationType</i> is not registered.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

#### Examples

The following example retrieves all bindkeys related to VLS-EXL.

```
hiGetBindKeys("VLS-EXL")
```

The following example retrieves all bindkeys related to the CIW.

```
hiGetBindKeys("Command Interpreter")
(("<Key>F3" "hiToggleEnterForm()")
 ("<Key>Help" "hiHelp('window 'ciw window(1))")
 ("<Key>F1" "hiHelp('window 'ciw window(1))")
)
```

The following example retrieves all bindkeys related to the Graphics Browser.

```
hiGetBindKeys("Graphics Browser")
(("Shift<Btn1Down>" "mouseAddSelectPt()" "mouseAddSelectPt()")
 ("<Key>F3" nil "hiToggleEnterForm()")
 ("Shift<Key>F4" "geToggleAreaSelectOption()")
 ("None<Btn5Down>" "geScroll(nil \"s\" nil)")
 ("None<Btn1Down>(2)" nil "mouseFinishPt()")
 ("<DrawThru3>" "hiZoomIn()" "hiZoomIn()")
 ("Ctrl<Btn5Down>" "hiZoomOutAtMouse()")
 ("None<Btn3Down>(2)" nil "hiToggleEnterForm()")
)
```

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

```
("None<Btn2Down>" "hiRepeat () " "")
("<DrawThru1>" "geSingleSelectBox () " "geSingleSelectBox () ")
("Shift<Btn2Down>" "" "")
("Shift<DrawThru3>" "hiZoomOut () " "hiZoomOut () ")
("Ctrl<DrawThru1>" "geSubSelectBox () " "geSubSelectBox () ")
("<Key>BackSpace" nil "deletePoint () ")
("None<Btn4Down>" "geScroll (nil \"n\" nil)")
("Shift<Btn5Down>" "geScroll (nil \"e\" nil)")
("Shift<Btn4Down>" "geScroll (nil \"w\" nil)")
("<Key>F4" "geTogglePartialSelect () ")
("Ctrl<Btn4Down>" "hiZoomInAtMouse () ")
("Ctrl<Key>Tab" "hiLowerWindow (hiGetCurrentWindow ()) ")
("None<Btn3Down>" "mousePopUp () " "mousePopUp () ")
("<Key>Escape" nil "cancelEnterFun () ")
("None<Btn1Down>" "mouseSingleSelectPt () " "mouseAddPt (t)")
("<Key>F1" "hiHelp ('window
deGetAppInfo (deGetEditViewType (hiGetCurrentWindow ())) ->appName)")
("Shift<DrawThru1>" "geAddSelectBox () " "geAddSelectBox () ")
("Ctrl<Btn1Down>" "mouseSubSelectPt () " "mouseSubSelectPt () ")
)
```

The following example uses `hiSetBindKey` for user-defined applications:

```
hiRegisterBindKeyPrefix ("RootApp")
hiRegisterBindKeyPrefix ("SubApp")
hiInheritBindKey ("SubApp" "RootApp")
hiSetBindKey ("RootApp" "<Key>T" "printf(\"test message RootApp\")")
hiSetBindKey ("RootApp" "<Key>Q" "printf(\"RootApp - Key Q\")")
```

The following example retrieves all bindkeys for applications at one level:

```
hiGetBindKeys ("SubApp")
("<Key>Q" "printf(\"SubApp - Key Q\")")
hiGetBindKeys ("RootApp")
(("<Key>Q" "printf(\"RootApp - Key Q\")")
("<Key>T" "printf(\"test message RootApp\")")
)
```

The following example retrieves all bindkeys from all levels from sub application to root application:

```
hiGetBindKeys ("SubApp" t)
(("<Key>Q" "printf(\"RootApp - Key Q\")")
("<Key>T" "printf(\"test message RootApp\")")
("<Key>Q" "printf(\"SubApp - Key Q\")")
)
```

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

)

In case the application does not have a root application, bindkeys for only one level are returned:

```
hiGetBindKeys("RootApp" t)
 ((" <Key>Q" "printf(\"RootApp - Key Q\")")
 (" <Key>T" "printf(\"test message RootApp\")")
)
```

### ***Related Topics***

[Bindkey Functions](#)

[hiRegisterBindKeyPrefix](#)

[hiSetBindKey](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

## hiShowBindKeys

```
hiShowBindKeys (
 [g_appTypeOrWindow]
 [g_showInheritBindings]
)
=> t / nil
```

### Description

Displays the bindkeys for an application type or window in a Viewfile window.

You can save the window contents to a file by selecting *Save* or *Save As* from the *File* pulldown menu or using the `hiSaveViewfile()` or `hiSaveAsViewfile()` functions. After saving the file, you can load it into another Opus session using the `load()` or `loadi()` SKILL functions.

If there is any bindkey inheritance for this bindkey table, the application aliases are displayed at the top of the window. If `g_appTypeOrWindow` is missing, a bindkey form is displayed.

This is a wrapper function. Depending on the type of the argument, it calls `hiShowBindKeysByAppType` or `hiShowBindKeysByWindow`.

### Arguments

`g_appTypeOrWindow` String or window ID. If it is a string, the argument is interpreted as an application type and the bindkeys are displayed using the `hiShowBindKeysByAppType` function. If it is a window ID, the bindkeys are displayed using the `hiShowBindKeysByWindow` function.

If `g_appTypeOrWindow` is not specified, or is an empty string, then the Bindkey Editor is displayed.

If the application type is specified, it must have been previously registered either through `deRegApp` or `hiRegisterBindKeyPrefix`.

`g_showInheritBindings`

If the `g_appTypeOrWindow` argument is an application type, then this argument is passed to `hiShowBindKeysAppType`. Otherwise, it is ignored.



## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### Values Returned

|     |                                                                |
|-----|----------------------------------------------------------------|
| t   | The bindkey is displayed.                                      |
| nil | <i>g_appTypeOrWindow</i> is invalid. A warning is also issued. |

#### ***Related Topics***

[Bindkey Functions](#)

[hiRegisterBindKeyPrefix](#)

[hiShowBindKeysByAppType](#)

[hiShowBindKeysByWindow](#)

[Bindkey Editor](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

## hiShowBindKeysByAppType

```
hiShowBindKeysByAppType (
 w_window1
 t_applicationType
 [g_showInheritBindings]
)
=> t / nil
```

### Description

Displays the bindkeys for an application in a Viewfile window.

You can save the window contents to a file by selecting *Save* or *Save As* from the *File* pulldown menu or using the `hiSaveViewfile` or `hiSaveAsViewfile` functions. After saving the file, you can load it into another Opus session using the `load()` or `loadi()` SKILL functions.

If there is any bindkey inheritance for this bindkey table, the application aliases are displayed at the top of the window.

If `t_applicationType` is an alias application, then only the bindkeys specifically defined for the alias application (which will override any bindkeys inherited from the root application) will be displayed.

### Arguments

|                                    |                                                                                                                                                                                                                                                                                                  |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_window1</code>             | The window in which you want to display the bindkeys.                                                                                                                                                                                                                                            |
| <code>t_applicationType</code>     | Any application type previously registered with <code>deRegApp</code> or <code>hiRegisterBindKeyPrefix</code> .                                                                                                                                                                                  |
| <code>g_showInheritBindings</code> | If <code>g_showInheritBindings</code> is specified, and it is not set as <code>nil</code> , nor is <code>t_applicationType</code> set as a root application, then the window contents will also include any bindings that have been defined for any alias application that has bindkeys defined. |

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### Value Returned

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>t</code>   | The bindkey is displayed.                                            |
| <code>nil</code> | <code>t_applicationType</code> is invalid. A warning is also issued. |

#### ***Related Topics***

[Bindkey Functions](#)

[hiRegisterBindKeyPrefix](#)

## hiShowBindKeysByWindow

```
hiShowBindKeysByWindow(
 [w_window1]
 [w_window2]
)
=> t / nil
```

### Description

Displays the bindkeys for a window in a Viewfile window.

You can save the window contents to a file by selecting *Save* or *Save As* from the *File* pulldown menu or using the `hiSaveViewfile()` or `hiSaveAsViewfile()` functions. After saving the file, you can load it into another Opus session using the `load()` or `loadi()` SKILL functions.

If there is any bindkey inheritance for this bindkey table, the application aliases are displayed at the top of the window.

If the application type for `w_window2` is set to an alias application, then the bindkey text window will display bindkeys defined both for the root application that the alias application inherits from, as well as any bindkeys defined specifically for the alias application (which will override root application bindkeys).

### Arguments

|                        |                                                                                |
|------------------------|--------------------------------------------------------------------------------|
| <code>w_window1</code> | Text window in which you want to display the bindkeys.                         |
| <code>w_window2</code> | Window associated with the application for which you want to see the bindkeys. |

### Values Returned

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>t</code>   | The bindkey is displayed.                                   |
| <code>nil</code> | <code>w_window</code> is invalid. A warning is also issued. |

### Related Topics

[Bindkey Functions](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiRegisterBindKeyPrefix

```
hiRegisterBindKeyPrefix(
 t_applicationType
 [t_ignored]
)
=> t / nil
```

#### Description

Registers an application bindkey prefix. Use this function to register prefixes that you do not want to register with the Design Editor application. Cadence applications are already registered.

This function should be called only when you want to have bindkeys for the application. The Design Editor application should be registered via the `deRegApp` call that calls this routine.

#### Arguments

|                          |                                                                                                                                     |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_applicationType</i> | The name of the application set as the <i>appType</i> of the window or the name of the plugin for which bindkeys are to be defined. |
| <i>t_ignored</i>         | The specified argument is ignored.                                                                                                  |

#### Values Returned

|            |                                                                                 |
|------------|---------------------------------------------------------------------------------|
| <i>t</i>   | The bindkey prefix is registered.                                               |
| <i>nil</i> | <i>t_applicationType</i> has already been registered. A warning is also issued. |

#### Related Topics

[Bindkey Functions](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiInheritBindKey

```
hiInheritBindKey(
 t_applicationType
 t_from_app_type
)
=> t / nil
```

#### Description

Causes a registered application to inherit bindkeys from another application.

Nested inheritance is not allowed, that is, *t\_from\_app\_type* cannot inherit bindkeys from another application.

Bindkeys defined for an application that inherits bindkeys will override those that are inherited from the root application. Also, an application can only inherit from one application and multiple inheritance is not allowed. However, any number of applications can inherit from any particular application.

#### Arguments

|                          |                                              |
|--------------------------|----------------------------------------------|
| <i>t_applicationType</i> | Application type to inherit the bindkeys.    |
| <i>t_from_app_type</i>   | Application type where bindkeys are defined. |

#### Value Returned

|     |                                                                                                                                                                                                                            |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| t   | The bindkeys are inherited.                                                                                                                                                                                                |
| nil | <i>t_applicationType</i> or <i>t_from_app_type</i> is invalid, <i>t_applicationType</i> already has bindkeys defined, or <i>t_from_app_type</i> inherited its bindkeys from another application. A warning is also issued. |

#### ***Related Topics***

[Bindkey Functions](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiGetBindKeyInheritRoot

```
hiGetBindKeyInheritRoot(
 t_applicationType
)
=> t_application / nil
```

#### Description

Returns the application name from which *t\_applicationType* inherits its bindkeys.

If *t\_applicationType* does not inherit bindkeys from another application, returns *t\_applicationType*.

#### Argument

|                          |                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------|
| <i>t_applicationType</i> | Application type that does not define bindkeys but inherits them from another application. |
|--------------------------|--------------------------------------------------------------------------------------------|

#### Values Returned

|                      |                                                                     |
|----------------------|---------------------------------------------------------------------|
| <i>t_application</i> | The name of the application from which the bindkeys were inherited. |
| nil                  | <i>t_applicationType</i> is invalid. A warning is also issued.      |

#### Related Topics

[Bindkey Functions](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiGetBindKeyInheritAlias

```
hiGetBindKeyInheritAlias(
 t_applicationType
)
=> l_application / nil
```

#### Description

Returns a list of application names that inherit the bindkeys from *t\_applicationType*.

#### Argument

*t\_applicationType*    Application type that defines bindkeys to be inherited by another application.

#### Values Returned

|                      |                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_application</i> | A list of applications that inherit bindkeys from this application type.                                                                               |
| <i>nil</i>           | There are no applications that inherit bindkeys from the specified application type, or <i>t_applicationType</i> is invalid. A warning is also issued. |

#### ***Related Topics***

[Bindkey Functions](#)



## **hiGetBindKeyPrefixList**

```
hiGetBindKeyPrefixList()
=> l_application
```

### **Description**

Returns a list of registered applications. This list is used to build the menu for the Cyclic button on the Bind Keys form.

### **Argument**

None

### **Value Returned**

*l\_application*

The list that is used to build the menu for the Cyclic button on the Bind Keys form. Root applications are listed with no leading spaces. Sub-applications (applications that inherit from a root application) are listed following the root application from which they inherit and have three leading spaces in the string.

Root applications are registered using `hiRegisterBindKeyPrefix`, and sub-applications are registered using `hiInheritBindKey`.

### ***Related Topics***

[Bindkey Functions](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

## hiConfigureBindKeys

```
hiConfigureBindKeys (
 ?showOnlyApps l_apps
 ?showAllChildren g_showAllChildren
 ?selectApp t_appname
 ?selectKeyBinding t_keyBinding
 ?showEmptyApps g_showEmptyApps
 ?showInheritedBindings g_showInheritedBindings
)
=> t / nil
```

### Description

Resets and displays the Bindkey Configuration form according to the supplied arguments. If the form is currently open, and contains unsaved changes, it will prompt before resetting the data.

See also [Configuring Application Bindkeys](#).

### Arguments

*?showOnlyApps l\_apps*

Accepts a string, or list of strings, where each string specifies the name of an application which will appear in the application tree in the Bindkey Configuration form. The default is "", which means that all applications with bindkeys defined will be shown. Wildcards can also be used, for example, *?showOnlyApps "Schematics"*.

*?showAllChildren g\_showAllChildren*

Boolean argument, where, by default, if a root application is matched, then all children of that application will be shown, rather than only those children which also match one of the supplied application names. Defaults to 't', but set to 'nil' to show only children of root applications that match the strings passed to *?showOnlyApps*.

*?selectApp t\_appname*

The name of the application to select in the application tree when the Bindkey Configuration form is displayed. Wildcards are accepted and the first match will be selected.

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

?selectKeyBinding *t\_keyBinding*

The keybinding string to be selected in the key table. The string need not be an exact string match for the keybinding. That is, it enough for the representation of the keybinding to be the same. For example, passing "CtrlShift<Key>a" will select "ShiftCtrl<Key>A", if found in the tree.

?showEmptyApps *g\_showEmptyApps*

Boolean argument that is the equivalent to the *Show applications with no bindkeys defined* checkbox option in the Bindkey Configuration form. The default value is 'Default, which indicates that the Bindkey Editor form will be opened with the same settings that were made the last time when the form was closed.

?showInheritedBindings *g\_showInheritedBindings*

Boolean argument that is the equivalent to the *Show inherited bindings* checkbox option in the Bindkey Configuration form. This will determine whether bindings that are inherited from a root app are shown when a child app is selected. The default value is 'Default, which indicates that the Bindkey Editor form will be opened with the same settings that were made the last time when the form was closed.

### Values Returned

|            |                                                              |
|------------|--------------------------------------------------------------|
| <i>t</i>   | Bindkey configuration form successfully reset and displayed. |
| <i>nil</i> | Command failed.                                              |

### Examples

```
hiConfigureBindKeys()
=> t
```

### Related Topics

[Bindkey Functions](#)

## hiEnableAsstBindKeyPrefix

```
hiEnableAsstBindKeyPrefix(
 t_prefix
)
=> t / nil
```

### Description

Removes the specified bindkey prefix (also known as an appType), which is associated with a dock window (assistant), from the list of disabled bindkey prefixes.

### Argument

|                 |                     |
|-----------------|---------------------|
| <i>t_prefix</i> | The bindkey prefix. |
|-----------------|---------------------|

### Return Values

|     |                                                |
|-----|------------------------------------------------|
| t   | The bindkey prefix was removed from the list.  |
| nil | The bindkey prefix does not exist in the list. |

### Examples

Removes `Palette` from the list of disabled bindkey prefixes.

```
hiEnableAsstBindKeyPrefix("Palette")
> t
```

`Palette1` does not exist in the list of disabled bindkey prefixes.

```
hiEnableAsstBindKeyPrefix("Palette1")
> nil
```

To view the list of disabled bindkey prefixes, use `hiGetDisabledAsstBindKeyPrefixes`.

### ***Related Topics***

[Bindkey Functions](#)

[hiGetDisabledAsstBindKeyPrefixes](#)

[hiDisableAsstBindKeyPrefix](#)

## hiDisableAsstBindKeyPrefix

```
hiDisableAsstBindKeyPrefix(
 t_prefix
)
=> t
```

### Description

Adds the specified bindkey prefix also known as an appType, which is associated with a dock window (assistant), to the list of disabled bindkey prefixes. As a result, the bindkeys defined using that prefix are disabled.

If the prefix for a dock window exists in the list, the bindkeys associated with that prefix are not used for the dock window; instead it uses the bindkey definition of the main window.

### Argument

|                 |                     |
|-----------------|---------------------|
| <i>t_prefix</i> | The bindkey prefix. |
|-----------------|---------------------|

### Value Returned

|          |                                                                     |
|----------|---------------------------------------------------------------------|
| <i>t</i> | The bindkey prefix was added to the list. Always returns <i>t</i> . |
|----------|---------------------------------------------------------------------|

### Examples

The Palette assistant defines its own bindkeys with the prefix `Palette`. For example, when the input focus is in the main window, the `F11` key toggles `on` and `off` the assistants that are open, however, when the focus is in the Palette, the `F11` key toggles `on` and `off` the Palette toolbars.

```
hiDisableBindKeyPrefix("Palette")
```

This function call adds the bindkey prefix `Palette` to the list of disabled prefixes. If you now press `F11` with the focus in the Palette, it will toggle `on` and `off` the assistants that are open, that is, the bindkeys defined using the Palette bindkey prefix are no longer in effect.

### Related Topics

[Bindkey Functions](#)

## hiGetDisabledAsstBindKeyPrefixes

```
hiGetDisabledAsstBindKeyPrefixes (
)
=> l_list / nil
```

### Description

Returns the list of bindkey prefixes disabled using `hiDisableAsstBindKeyPrefix`.

### Argument

None

### Return Values

|                     |                                        |
|---------------------|----------------------------------------|
| <code>l_list</code> | The list of disabled bindkey prefixes. |
| <code>nil</code>    | No bindkey prefix is disabled.         |

### Examples

Returns the list of bindkey prefixes, "App1 ", "App2 ", and "App3 ", that have been disabled using `hiDisableAsstBindKeyPrefix`.

```
hiDisableBindKeyPrefix("App1")
> t
hiDisableBindKeyPrefix("App2")
> t
hiDisableBindKeyPrefix("App3")
> t
hiGetDisabledBindKeyPrefixes()
> ("App1" "App2" "App3")
```

### ***Related Topics***

[Bindkey Functions](#)

[hiDisableAsstBindKeyPrefix](#)

## Cadence User Interface SKILL Reference

### Bindkey Functions

---

#### hiPrintBindKeyUsage

```
hiPrintBindKeyUsage(
 [?appNames l_appNames]
 [?getData t_val]
)
=> t / list
```

#### Description

Returns bindkey usage data for the specified applications.

#### Arguments

*?appNames l\_appNames*

Specifies the applications for which statistics are to be printed. The default value is `nil`, which means information for all applications is printed.

*?getData t\_val*

Specifies whether to return the statistics data. When set to `t`, a list structure that contains the requested data is returned. The default value is `nil`, which means no statistic data is returned.

#### Value Returned

*t*

No statistic data is returned.

*list*

A list structure is returned. The list structure contains all the specified applications with their respective statistic data.

#### Examples

Prints data for all applications:

```
>hiPrintBindKeyUsage()
```

Prints data for `Layout` applications and returns the data:

```
>hiPrintBindKeyUsage(?appNames list("Layout") ?getData t)
```

#### Related Topics

#### Bindkey Functions

## **Cadence User Interface SKILL Reference**

### **Bindkey Functions**

---



---

## User Entry Functions

---

User entry functions collect data (such as the points of a shape, a string, or a number) that you provide. These functions are known as enterfunctions and this term is used throughout this section.

Enterfunctions are a special class of SKILL functions specific to a window or group of windows. They let you digitize shapes or enter a string or number. When you invoke an enterfunction, the current window is the only window where data can be entered. Some applications, such as the layout editors, have a group of windows where each of the windows is viewing the same database cell. In this case, the enterfunctions let you enter data in any of the windows. Data entered in a window that is not in the window group has no effect on an enterfunction.

The functions are nestable, meaning that you can be entering a box and then start entering an arc. When you finish digitizing the arc, you can complete the box.

Enterfunctions are specific to a window or group of windows, you can have separate enterfunctions active in unrelated windows. To illustrate this, suppose you have two unrelated windows A and B. You start an `enterBox` function in window A and then start an `enterLine` function in window B. You can enter a point for the box in window A, enter a line point on window B, and so forth. The system treats them as two separate functions.

The `enterMultiRep` enterfunction is an exception to this rule and is described later in this section.

If an enterfunction is used inside an application window, any adjustments to the list of points and/or "snapping" points is performed by the application. Refer to the documentation for the specific application for more information.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- `addPoint`
- `applyEnterFun`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

- cancelEnterFun
- changeEnterFun
- changeNextEnterFun
- clearAllEnterFunctions
- deletePoint
- drawEnterFun
- enterArc
- enterBox
- enterCircle
- enterDonut
- enterEllipse
- enterFunNestLevel
- enterLine
- enterString
- finishEnterFun
- hiGetCurrentCmd
- hiInEnterFun
- hiMarkNestable
- hiMarkNonNestable
- undrawEnterFun
- hiUpdate
- preXY
- setEFunFlightLineTarget
- undrawEnterFun

## Enterfunction Prompts

Prompts let you know that you have an enterfunction active in a window. A prompt string (supplied to the enterfunction) is displayed on the CIW prompt line whenever you enter data into a window that has an enterfunction active. Prompt strings should be informative and let users know what the state of the enterfunction is, as well as what they should enter.

If an enterfunction requires multiple points, the caller supplies a list of prompt strings, one for each point. The prompt displayed in the CIW advances one item in the list of strings as each point is entered. Suppose that an enterfunction requires two points and the prompt list provided is

```
'("Enter First Point" "Enter Second Point")
```

Before you enter a point, the prompt line would say

```
"Enter First Point"
```

After you have entered the first point, the next string is displayed

```
"Enter Second Point"
```

If you delete a point, the previous prompt message is displayed. If the list of prompt strings is exhausted before you enter all the points, the last prompt message continues to display. Extra prompt strings are ignored. If a prompt list is not supplied, prompts are not displayed. (This practice is not recommended.)

## Terminating an Enterfunction

Enterfunctions are terminated in one of several ways. All enterfunctions can be canceled, by calling `cancelEnterFun`, or finished, by calling `finishEnterFun`. When an enterfunction is canceled, all data entered is lost and `nil` is returned. When it is finished, the data entered is returned as the result of the enterfunction.



Do NOT use the return value of the enterfunction to gather data but to register a callback procedure for this purpose. Enterfunctions will no longer block in future releases, and the return value will indicate only that the enterfunction was successfully started.

Enterfunctions that take a specific number of points, such as `enterArc` or `enterBox`, are automatically terminated when the last point is entered. Functions that take a variable number of points are terminated automatically only when two identical points are entered. Otherwise, enterfunctions are terminated only with an explicit call to `finishEnterFun` or `cancelEnterFun`. If a pointlist is passed for preloading to an enterfunction that requires a

## Cadence User Interface SKILL Reference

### User Entry Functions

---

variable number of points, the enterfunction returns without further interaction if the last two points in the list are identical.

#### ***Related Topics***

[Callback Procedures](#)

[Enterfunction Flags](#)

[Nesting Enterfunctions](#)

[finishEnterFun](#)

## Callback Procedures

Enterfunctions do nothing but gather data, they are service layer functions and are used by higher level procedures, such as adding a polygon to a database cell. To allow the higher level procedures to respond to or to monitor the progress of the enterfunction, callback procedures should be supplied to the enterfunction.

These procedures are not required, but allow an application to trap a condition as soon as the condition becomes known. This prevents having to enter twenty points of a shape, only to find that the second point caused a problem. The enterfunctions can have registered callback procedures that are invoked for the following events:

- The enterfunction is starting
- A point is entered
- A point is deleted
- The enterfunction is finished or canceled

These functions are known as the `initProc`, `addPointProc`, `delPointProc`, `formProc`, and `doneProc`.

Do not call `finishEnterFun` or `cancelEnterFun` from within any of the callback procedures.

### Callback Procedure `addPointProc`

`addPointProc` is called after you enter a point and is passed the `w_windowId` and the list of points that you have entered so far. The last point in the list is the point you just entered.

```
addPointProc(w_windowId l_pointList)
```

To reject the most recently added point, `addPointProc` should call the `deletePoint` function.

### Callback Procedure `delPointProc`

`delPointProc` is called when you delete a point and is passed the `w_windowId` and the list of points you have entered so far. The last point in the list is the point that is about to be removed.

```
delPointProc(w_windowId l_pointList)
```

## Callback Procedure formProc

`formProc` is called when the options form is about to be displayed. It is passed the symbol for the options form and a flag indicating whether the form is being displayed (`g_map == t`) or erased (`g_map == nil`). You normally use this to make sure the form is initialized with current values.

```
formProc(s_form g_map)
```

## Callback Procedure doneProc

`doneProc` is called when you finish an enterfunction or cancel it. It is passed the `w_windowId`, either `t` or `nil` (`t` if the enterfunction completed, `nil` if it was canceled), and the list of points that you entered. For `enterString`, the string that you typed is passed instead of a point list. For `enterNumber`, the number you typed is passed instead of a point list. `doneProc` is important when enterfunctions are nested. `doneProc` and nesting are discussed at the end of this section.

```
doneProc(w_windowId t | nil l_pointList)
```

### ***Related Topics***

[Enterfunction Flags](#)

[Enterfunction Prompts](#)

[Nesting Enterfunctions](#)

[finishEnterFun](#)

## Enterfunction Flags

All enterfunctions can have certain flags passed that control the operation of the enterfunction. These flags control such things as

- Drawing rubber band lines
- Automatic display of the options form
- Acceptance of a string or number without termination
- The use of infix

### Enterfunction Flag `dontDraw`

If `dontDraw` is `nil` (the default), the rubber band lines are drawn between the entered points. Otherwise, these lines are not displayed.

### Enterfunction Flag `alwaysMap`

If `alwaysMap` is `nil` (the default), the options form is displayed only if you request it with `hiToggleEnterForm` or if you are in novice mode. Otherwise, the options form is displayed automatically when the enterfunction is called.

### Enterfunction Flag `acceptString`

If `acceptString` is `nil` (the default) and you enter a string into the CIW, the enterfunction is terminated immediately. Otherwise, you can enter the string without terminating the enterfunction. In the latter case, `doneProc` is called with the string instead of the entered points. `doneProc` knows that the enterfunction is not being terminated.

### Enterfunction Flag `acceptNumber`

If `acceptNumber` is `nil` (the default) and you enter a number into the CIW, the enterfunction is terminated immediately. Otherwise, you can enter the number without terminating the enterfunction. In this latter case, `doneProc` is called with the number instead of the entered points. `doneProc` knows that the enterfunction is not being terminated.

## **Enterfunction Flag noInfix**

Infix mode allows you to specify the first point for any operation implicitly. When infix is on and an enterfunction is executed through a bindkey in some window, the cursor location at which the function was executed is taken as the first point entered in the enterfunction. If infix is off, this point is ignored and you must explicitly enter it. Infix cannot be used from a menu item because there is no location implicitly specified. The `noInfix` flag, when `nil` (the default), allows infix to be used when the global infix flag (`window(1)->infixOn`) is set and when the function is called from a bindkey. If `noInfix` is not `nil`, then infix is not used no matter what the setting of the global flag.

### ***Related Topics***

[Enterfunction Prompts](#)

[Callback Procedures](#)

[Nesting Enterfunctions](#)

[undrawEnterFun](#)



## Nesting Enterfunctions

Most enterfunctions can be nested inside one another, meaning that while one is active, another can be started. When the nested enterfunction terminates, the previous one again becomes active, and its state at the time of the second enterfunction is restored.

Because you can nest enterfunctions, and because the functions block until they are canceled or completed, it helps to be aware of what happens inside the software. Enterfunctions start new SKILL top levels, in which the flow of control is passed from one `topLevel` to another. The processing of user events does not stop, but evaluation of statements after the enterfunction is blocked until the `topLevel` is completed (either by completing an enterfunction or by cancelling it).

Suppose you have two procedures A and B

```
procedure(A()
 prog((result)
 result = enterBox()
 println("The result from A is %L\n" result)
 return(result)
)
)

procedure(B()
 prog((result)
 result = enterArc()
 println("The result from B is %L\n" result)
 return(result)
)
)
```

and you have two windows, `window1` and `window2`. If `window1` is current and you type `A( )` in the CIW, procedure A executes. From within procedure A, `enterBox` is called and evaluation blocks at this statement, waiting for you to complete the enterfunction.

Now suppose you went to `window2`, made it current, and then invoked procedure B. Procedure B calls `enterArc` and evaluation blocks at this statement, waiting for you to complete the arc.

The system allows you to enter points in either `window1` or `window2`. Suppose you enter the two points of the box in `window1`. The `println` will not be executed because evaluation is blocked inside procedure B. As soon as you finish the arc in `window2`, the `println` statement in procedure B is evaluated. Procedure B then returns, allowing procedure A to return, and its `println` statement is evaluated.

Due to this blocking behavior, we strongly suggest that you use a `doneProc` whenever you need to do something immediately after you complete an enterfunction.

If we rewrote the previous example using a `doneProc`, it would behave better.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

```
procedure(resultsA(windowId doneOrCancel pointsList)
 if(doneOrCancel == t) /* it was finished */
 then println("The result from A is %L" pointsList)
)
)

procedure(A()
 enterBox(?doneProc "resultsA")
)

procedure(resultsB(windowId doneOrCancel pointsList)
 if(doneOrCancel == t) /* it was finished */
 then println("The result from B is %L" pointsList)
)
)

procedure(B()
 enterArc(?doneProc "resultsB")
)
```

The trick to using the enterfunctions in a nested environment is to organize your code in a callback style, doing what you want to do with the data in a callback procedure, rather than after the enterfunction statement.

### ***Related Topics***

[Enterfunction Flags](#)

[Callback Procedures](#)

[Enterfunction Prompts](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### addPoint

```
addPoint(
 l_point
)
=> t / nil
```

#### Description

Adds the point to the enterfunction in the current window.

`addPoint` is often used in conjunction with `hiGetCommandPoint` to add the point where you last clicked to an enterfunction.

#### Argument

|                      |                                                          |
|----------------------|----------------------------------------------------------|
| <code>l_point</code> | Point in user units to add to the current enterfunction. |
|----------------------|----------------------------------------------------------|

#### Values Returned

|                  |                         |
|------------------|-------------------------|
| <code>t</code>   | The point is added.     |
| <code>nil</code> | The point is not added. |

#### Examples

Displays the prompt Enter a point is displayed on the CIW prompt line:

```
procedure(pointDone(w done pts)
 if(done then
 printf("Point entered was %L.\n" car(pts))
 else
 println("Point entry aborted.")
)
)
enterPoint(?prompts
 list("Enter a point.")
 ?doneProc "pointDone"
)
```

Adds the point at 2 : 6 location:

```
addPoint(2:6)
```

The enterfunction terminates and `doneProc` prints the specified message:

```
Point entered was (2.000 6.000).
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

The enterfunction returns the point list:

```
(2.000 6.000)
```

#### ***Related Topics***

[User Entry Functions](#)

[hiGetCurrentCmd](#)

[hiGetCommandPoint](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

## applyEnterFun

```
applyEnterFun(
 [g_form]
)
=> t / nil
```

### Description

Calls `doneProc`, passes it the points you entered, resets the enterfunction, and calls `initProc`.

This function does not return from the active enterfunction.

`applyEnterFun` can be used in conjunction with [doneProc](#) to allow you to enter many shapes without having to start another enterfunction.

### Argument

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>g_form</i> | Form handle used to locate the enterfunction to apply. Normally the enterfunction applied is the one active in the current window. If you want to apply a specific enterfunction, whether or not it is the current window, there is no way to tell which window is the correct one. So you pass the same form that was originally supplied to the enterfunction to <code>applyEnterFun</code> , and it applies that specific enterfunction regardless of whether it is in the current window or not. |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### Values Returned

|     |                                 |
|-----|---------------------------------|
| t   | The enterfunction is begun.     |
| nil | The enterfunction is not begun. |

### Examples

Displays the prompt `First` in the CIW. Enter point *p1*. The prompt changes to "Next." Enter *p2* and *p3*.

```
procedure(donePoints(w done pts)
 if(done then
 printf("Points = %L\n" pts)
 else
 println("Point entry aborted.")
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

```
)
)
 enterPoints(
 ?prompts list("First" "Next") ?doneProc "donePoints")
 applyEnterFun()
```

The enterfunction is not terminated, the prompt changes back to "First," and "Points = ( p1 p2 p3 )" is printed out. Now enter points p4, p5, and p6, followed by

```
finishEnterFun()
```

The enterfunction is terminated, the prompt goes away, the message "Points = ( p4 p5 p6 )" is printed out, and ( p4 p5 p6 ) is returned.

### ***Related Topics***

User Entry Functions

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### cancelEnterFun

```
cancelEnterFun(
 [g_form]
)
=> t / nil
```

#### Description

Cancels the enterfunction in the current window.

**Note:** Do not call `cancelEnterFun` from within an enterfunction callback procedure.

In SKILL code, when an enterfunction is currently active, do not use `cancelEnterFun` before invoking a new enterfunction or you might get unexpected behavior; use `changeNextEnterFun` instead.

#### Argument

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>g_form</i> | Form handle used to locate the enterfunction to cancel. Normally the enterfunction that is canceled is the one active in the current window. If you want to cancel a specific enterfunction, whether or not it is in the current window, there is no way to tell which window is the correct one. So you pass the same form that was originally supplied to the enterfunction to <code>cancelEnterFun</code> , and it cancels that specific enterfunction regardless of whether it is in the current window or not. |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Values Returned

|     |                                                         |
|-----|---------------------------------------------------------|
| t   | The enterfunction is canceled.                          |
| nil | There is no active enterfunction in the current window. |

#### Examples

Displays the prompt `First` in the CIW:

```
enterSegment(?prompts list("First" "Second")
```

Terminates the enterfunction and the prompt goes away. `nil` is returned.

```
cancelEnterFun()
```

## **Cadence User Interface SKILL Reference**

### **User Entry Functions**

---

#### ***Related Topics***

User Entry Functions



## Cadence User Interface SKILL Reference

### User Entry Functions

---

## changeEnterFun

```
changeEnterFun (
 s_enterFun
 [?prompts l_promptList]
 [?points l_pointList]
 [?wantPoints x_pointLimit]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?pathStyle t_pathStyle]
 [?cursor x_cursor]
 [?pathWidth f_pathWidth]
 [?beginExtent f_beginExtent]
 [?endExtent f_endExtent]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?target l_point]
)
=> t / nil
```

## Description

Changes the current enterfunction to another one.

Since this function uses the current window to determine which enterfunction to change, make sure to properly set the current window before using this function. See [hiSetCurrentWindow](#) for information about setting the current window.

You can use `changeEnterFun` to get immediate response to any changes the user makes to the values on the option forms.

`s_enterFun` is the symbol of the enterfunction you want to change. For example, if you want to change to an *enterBox*, you specify the argument as `'enterBox`.

`changeEnterFun` should always be provided with a `?doneProc` argument.

`changeEnterFun` cancels the current enterfunction and then changes it to the new one. The call to the original enterfunction remains blocked until the changed enterfunction exits. When the new enterfunction is completed, it calls `doneProc`. Do not use `changeEnterFun`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

without specifying a `doneProc` because it will not return from the original enterfunction until the changed enterfunction returns or is canceled.

`changeEnterFun` allows a union of all the enterfunction arguments, although it uses only the arguments that apply to the enterfunction specified by `s_enterFun`.

### Arguments

|                                          |                                                                                                                                       |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <code>s_enterFun</code>                  | Symbol for the enterfunction you wish to change to.                                                                                   |
| <code>?prompts l_promptList</code>       | List of prompt strings used by the new enterfunction.                                                                                 |
| <code>?points l_pointList</code>         | List of points that can be preloaded for the new enterfunction to operate on.                                                         |
| <code>?wantPoints x_pointLimit</code>    | Number of points needed by the new enterfunction.                                                                                     |
| <code>?form s_form</code>                | Form supplied to the new enterfunction.                                                                                               |
| <code>?addPointProc t_addProcName</code> | <code>addPointProc</code> corresponding to the new enterfunction.                                                                     |
| <code>?delPointProc t_delProcName</code> | <code>delPointProc</code> corresponding to the new enterfunction.                                                                     |
| <code>?initProc t_initProcName</code>    | <code>initProc</code> corresponding to the new enterfunction.                                                                         |
| <code>?doneProc t_doneProcName</code>    | <code>doneProc</code> corresponding to the new enterfunction.                                                                         |
| <code>?formProc t_formProcName</code>    | <code>formProc</code> corresponding to the new enterfunction.                                                                         |
| <code>?pathStyle t_pathStyle</code>      | PathStyle to use if the new enterfunction is <code>enterPath</code> .                                                                 |
| <code>?cursor x_cursor</code>            | Cursor that is used when the enterfunction is active. For more information on all available cursors see <a href="#">hiSetCursor</a> . |
| <code>?pathWidth f_pathWidth</code>      |                                                                                                                                       |

## Cadence User Interface SKILL Reference

### User Entry Functions

---

Path width to use if the new enterfunction is `enterPath`.

`?beginExtent` *f\_beginExtent*

Beginning extension to use if the new enterfunction is `enterPath`.

`?endExtent` *f\_endExtent*

Ending extension to use if the new enterfunction is `enterPath`.

`?dontDraw` *g\_dontDraw*

Determines whether or not to draw rubber band lines.

`?alwaysMap` *g\_alwaysMap*

Determines whether or not to immediately display the options form.

`?acceptString` *g\_acceptString*

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber` *g\_acceptNumber*

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix` *g\_noInfix*

Determines whether or not to disable infix mode.

`?cmdName` *t\_cmdName*

The name of the new enterfunction. You can use this name as an identifier to control nesting.

`?target` *l\_point*

The target point to which a flight line is drawn from the current cursor.

This argument is valid only for the `enterLine` function.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### Values Returned

|                  |                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | The original enterfunction was successfully terminated and the new enterfunction was successfully initialized. |
| <code>nil</code> | The enterfunction was not successfully terminated or the new enterfunction was not initialized.                |

#### Examples

`circleForm` is a form containing options for drawing a circle. Here, because the `alwaysMap` flag is `t`, the form is displayed. The prompt in the CIW reads `Center`. Enter point `p1`. The prompt reads `Radius`, and a rubberbanding circle with center at `p1` follows the cursor.

```
procedure(circleDone(w done pts)
 if(done then
 printf("Circle = %L\n" pts)
 else
 println("Circle entry aborted.")
)
)
)
procedure(ellipseDone(w done pts)
 if(done then
 printf("Ellipse = %L\n" pts)
 else
 println("Ellipse entry aborted.")
)
)
)
enterCircle(?prompts list("Center" "Radius")
 ?doneProc "circleDone" ?form circleForm ?alwaysMap t)
```

The message `Circle entry aborted` is printed, and the circle options form and the rubberbanding circle disappear as explained above. The prompt in the CIW is changed to `First corner`, and the `ellipseForm` appears. Enter `p2`, and a rubberbanding ellipse is drawn.

```
changeEnterFun('enterEllipse
 ?prompts list("First Corner" "Second Corner")
 ?doneProc "ellipseDone" ?form ellipseForm ?alwaysMap t)
```

#### ***Related Topics***

#### User Entry Functions

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### changeNextEnterFun

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

#### Description

If an enterfunction is currently active, sets a flag that causes the next enterfunction invocation for the current window or a specified window to use the `changeEnterFun` logic rather than being nested.

#### Argument

|                       |                                                                                                                            |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------|
| <code>w_window</code> | Window where the <code>changeEnterFun</code> logic is to be used. If no argument is specified, the current window is used. |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------|

#### Values Returned

|                  |                      |
|------------------|----------------------|
| <code>t</code>   | The flag is set.     |
| <code>nil</code> | The flag is not set. |

#### ***Related Topics***

[User Entry Functions](#)

[hiGetCurrentCmd](#)

[hiMarkNonNestable](#)

[hiMarkNestable](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### clearAllEnterFunctions

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

#### Description

Cancels all nested enterfunctions associated with the current window.

Same as calling `cancelEnterFun` repeatedly until no more enterfunctions are running in the current window.

#### Arguments

None

#### Value Returned

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>t</code>   | The functions are canceled or there is nothing to cancel.            |
| <code>nil</code> | There are no windows open other than the Command Interpreter Window. |

#### ***Related Topics***

[User Entry Functions](#)

[finishEnterFun](#)

[hiGetCurrentCmd](#)

[hiMarkNonNestable](#)

[hiMarkNestable](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### deletePoint

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

#### Description

Deletes a point from the enterfunction in the current window.

#### Argument

None

#### Values Returned

|     |                                                                    |
|-----|--------------------------------------------------------------------|
| t   | The point was successfully deleted from the pending enterfunction. |
| nil | The current window is not in an enterfunction.                     |

#### Examples

Displays the prompt Enter the second point in the CIW prompt line.

```
procedure(segDone(w done pts)
 if(done then
 printf("Segment entered was %L.\n" pts)
 else
 println("Segment entry aborted.")
)
)

procedure(myEnterSegment()
 enterSegment(?prompts
 list("Enter the first point."
 "Enter the second point.")
 ?doneProc "segDone"
)
)
```

Sets the infix mode on and bind the s key to the procedure myEnterSegment. Type the s key in a Schematics window at location p0. Because of infix mode, p0 is taken as the first point for the enterSegment call. The rubberbanding line immediately appears between p1 and the cursor location.

```
window(1)->infixOn = t
hiSetBindKey("Schematics" "<Key>s" "myEnterSegment()")
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

The first prompt is skipped since the first point is already entered. Type:

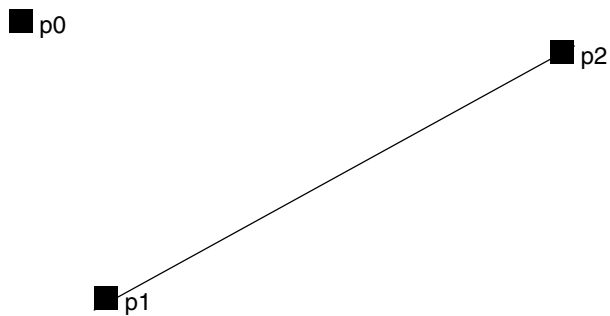
```
deletePoint ()
```

The rubberbanding line goes away, and the CIW prompt changes to “Enter the first point.” Enter point *p1*. The rubberbanding line reappears from *p1* to the cursor, and the prompt changes to Enter the second point. Enter *p2*. `doneProc` prints out the specified message:

```
Segment entered was (p1 p2).
```

The enterfunction returns the point list:

```
(p1 p2)
```



### ***Related Topics***

[User Entry Functions](#)



## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### drawEnterFun

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

#### Description

Redraws rubberbanding shapes. Used when an enterfunction is active.

#### Argument

None

#### Values Returned

|     |                                                          |
|-----|----------------------------------------------------------|
| t   | The draw is successful.                                  |
| nil | There is no enterfunction pending in the current window. |

#### Examples

Redraws the segments that were erased by `undrawEnterFun`:

```
drawEnterFun()
```

The two segments connecting the three points are redrawn.

#### ***Related Topics***

[User Entry Functions](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterArc

```
enterArc(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_point_list / nil
```

#### Description

Enters an arc into the current window.

An arc is defined as a portion of a horizontally or vertically oriented ellipse. It is specified by three points: the starting point of the arc on the ellipse, the ending point of the arc on the ellipse, and a point between them through which the arc must pass. An arc can subtend no more than 180 degrees.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to appear on the CIW prompt line. If supplied, only the first three strings are used. Each string corresponds to one of the three points needed to specify the arc.

*?points l\_pointList*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

List of no more than three points specifying the arc. If all three points are supplied, the enterfunction returns the list after adjustment. Otherwise, the supplied points are adjusted and displayed as if you had entered them explicitly.

After the first point is entered, a rubberband line is drawn between this point and the cursor's current position. After the second point is entered, a rubberbanding arc is drawn so that it passes through the first and second points and the cursor's current position. When the third point is entered, the function returns the point list.

If you attempt to enter a point through which a legal arc cannot be drawn, nothing is displayed. If the function is terminated at this point, no points are returned.

`?form s_form` Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`  
Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`  
Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`  
Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`  
Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`  
Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`  
Determines whether or not to draw rubberband lines.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name to associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

`l_point_list`

The list of points you entered in user units.

`nil`

The function is canceled or an arc could not be generated from the entered points.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt Enter the first point of your arc on the CIW prompt line. When you enter point *p1*, the prompt changes to Enter the second point of your arc and a rubber band line appears from *p1* to the cursor. Enter point *p2* and the line becomes an arc while the prompt changes to Enter the last point of your arc.

```
procedure(arcDone(w done pts)
 if(done then
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

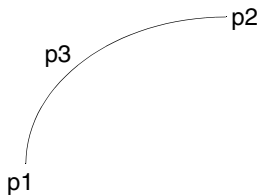
```
 printf("Arc points are %L.\n" pts)
 else
 println("Arc entry aborted.")
)
)
enterArc(?prompts list("Enter the first point of your arc."
 "Enter the second point of your arc."
 "Enter the last point of your arc.")
 ?doneProc "arcDone"
)
```

Finally, enter point *p3* and `doneProc` prints the three arc points:

Arc points are ( p1 p2 p3 ).

The `enterfunction` returns a list of the three arc points.

( p1 p2 p3 )



### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterBox

```
enterBox(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_bBox / nil
```

#### Description

Enters a box in the current window.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to appear on the CIW prompt line. If supplied, only the first two strings are used. Each string corresponds to one of the two points needed to specify the box.

*?points l\_pointList*

List of no more than two points specifying the box. If both points are specified, the box is returned after adjustment. If not, you are prompted for the point or points not specified. If one point is supplied, it is adjusted as if you had entered it. After you enter the first point, a rubberbanding box is drawn between the first point and the cursor's current position. After the second point is entered, the function returns a list of two points. The list returned is normalized so that the first point corresponds to the lower left corner of the box; the second is the upper right corner.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?form s_form` Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`  
Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`  
Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`  
Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`  
Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`  
Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`  
Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`  
Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`  
Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`  
Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`  
Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

|                               |                                                                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | Name you associate with the enterfunction. If the command name is set, it can be accessed with <code>hiGetCurrentCmd</code> .                                                                                                                                                  |
| <code>?cursor x_cursor</code> | An integer argument that will default to an appropriate cursor. For example, <code>hicArrowRectangle</code> for <code>enterBox</code> and <code>hicArrowArc</code> for <code>enterArc</code> . For more information on all available cursors see <a href="#">hiSetCursor</a> . |

### Value Returned

|                     |                         |
|---------------------|-------------------------|
| <code>l_bBox</code> | A list of two points.   |
| <code>nil</code>    | A nil list is returned. |

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

The prompt Enter the first corner of your box is displayed on the CIW prompt line. `myForm` is a form containing two string items, one for the layer and one for the purpose. The form is immediately displayed, regardless of user ability mode. The layer-purpose pair is initially `y2` and `drawing`. You change the layer to `ndiff` and the purpose to `pin`.

```
procedure(boxDone(w done pts)
 if(done then
 printf("Box entered was %L. Layer is %L %s.\n" pts
 list(myForm->layer->value
 myForm->purpose->value)
 hiGetWindowName(w))
 else
 println("Box entry aborted.")
)
)

procedure(boxForm(form map)
 if(map then
 eval(form->layer->value = "y2"
 eval(form->purpose->value = "drawing"
 println("map form")
 else
 println("unmap form")
)
)

hiCreateOptionsForm(
 'myForm
 "Layer Entry"
 list(hiCreateStringField(
 ?name 'layer
 ?prompt "purpose"
```



## Cadence User Interface SKILL Reference

### User Entry Functions

---

```
)
 hiCreateStringField(
 ?name 'purpose
 ?prompt "purpose"
)
)
nil
)
procedure(test()
 enterBox(
 ?prompts list("Enter the first corner of your box."
 "Enter the last corner of your box.")
 ?doneProc "boxDone"
 ?formProc "boxForm"
 ?alwaysMap t
 ?form eval(myForm)
)
)
hiSetBindKey("Layout" "<Key>5" "test()")
```

A rubberband box appears from  $p_1$  to the cursor. Enter point  $p_2$  and `doneProc` prints out the box points:

Box entered was (  $p_1$   $p_2$  ). Layer is ("ndiff" "pin").

The `enterfunction` returns the box points:

(  $p_1$   $p_2$  )



### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterCircle

```
enterCircle(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_bBox / nil
```

#### Description

Enters a circle in the current window.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to appear on the CIW prompt line. If supplied, only the first two strings are used. The first string corresponds to the center, which is the first point entered, and the second string corresponds to a point on the circumference.

*?points l\_pointList*

List of no more than two points. If both points are supplied, they are returned after adjustment with no further interaction. If one point is supplied, it is adjusted and you are prompted to enter the second point. The first point entered is the center of the circle. After entry, a rubberbanding circle appears about the center and passing through the cursor. When the second point is entered, the bounding box of the circle is computed and returned.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?form s_form` Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`  
Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`  
Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`  
Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`  
Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`  
Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`  
Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`  
Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`  
Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`  
Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`  
Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

|                               |                                                                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | Name you associate with the enterfunction. If the command name is set, it can be accessed with <code>hiGetCurrentCmd</code> .                                                                                                                                                  |
| <code>?cursor x_cursor</code> | An integer argument that will default to an appropriate cursor. For example, <code>hicArrowRectangle</code> for <code>enterBox</code> and <code>hicArrowArc</code> for <code>enterArc</code> . For more information on all available cursors see <a href="#">hiSetCursor</a> . |

### Value Returned

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <code>l_bBox</code> | A list containing the bounding box of the circle. |
| <code>nil</code>    | A nil list is returned.                           |

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt Enter the circle center on the CIW prompt line.

```
procedure(circleInit(w)
 leSetEntryLayer(list("pdiff" "drawing"))
)
procedure(circleDone(w done pts)
 if(done then
 printf("Circle entered was %L. Layer is %L.\n" pts
 leGetEntryLayer())
 else
 println("Circle entry aborted.")
)
)
enterCircle(?prompts list("Enter the circle center."
 "Enter a point on the circumference.")
 ?doneProc "circleDone" ?initProc "circleInit"
)
```

The `initProc` forces the window current layer and purpose pair to be ("pdiff" "drawing"). When you enter point *p1* the prompt changes to Enter a point on the circumference. A rubberband circle appears with center *p1* and passing through the cursor. Enter point *p2* and `doneProc` prints the circle points:

```
Circle entered was (p1 p2). Layer is ("pdiff" "drawing").
```

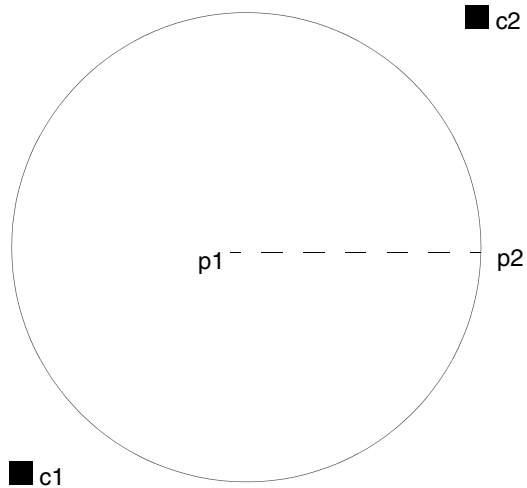
Here, (c1 c2) is the bounding box of the circle entered. The enterfunction returns a list of the circle points:

## Cadence User Interface SKILL Reference

### User Entry Functions

---

( c1 c2 )



#### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterDonut

```
enterDonut (
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_point_list / nil
```

#### Description

Enter a donut in the current window.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to appear on the CIW prompt line. If supplied, only the first three strings are used. The first string corresponds to the center, which is the first point entered. The second string corresponds to a point on the circumference of the donut hole. The third string corresponds to a point on the outer circumference of the donut.

*?points l\_pointList*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

List of no more than three points. If all points are specified, they are returned after adjustment. If not, the supplied points are adjusted and you are prompted for the remaining points.

The first point is the center of the donut. After the first point is entered, a rubberbanding circle is drawn with that center point and passing through the cursor. When the second point is entered, the circle becomes permanent, and another rubberbanding circle is drawn. After the third point is entered, the three points are returned in a list.

The second and third points determine the hole radius and the outer radius of the donut and can be specified in either order. The returned list orders the points as center, inner radius, and outer radius.

`?form s_form`

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the `enterfunction` and its caller.

`?addPointProc t_addProcName`

Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the `enterfunction` is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the `enterfunction` is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`

Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hiArrowRectangle` for `enterBox` and `hiArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

`l_point_list`

The entered points as a list containing the coordinates for the donut.

`nil`

A nil list is returned.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

```
procedure(donutAddPt(w pts)
 prog((p)
 p = car(pts)
 if(car(p) < 0 || cadr(p) < 0 then
 warn("Donut center is not in first quadrant.\n"))
)
)

procedure(donutDone(w done pts)
 if(done then
 printf("Donut entered was %L.\n" pts)
```



## Cadence User Interface SKILL Reference

### User Entry Functions

---

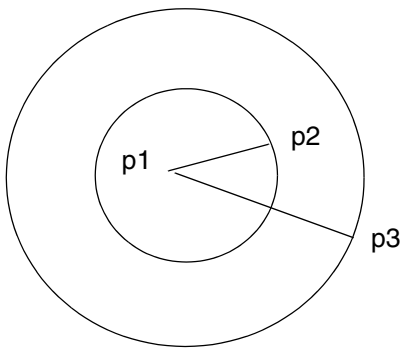
```
 else
 println("Donut entry aborted.")
)
)
enterDonut(?prompts list("Enter the donut center."
 "Enter a point on the circumference of the hole."
 "Enter a point on the circumference of the donut.")
 ?doneProc "donutDone" ?addPointProc "donutAddPt"
)
```

A rubber band circle appears with center  $p1$  and passing through the cursor. Enter point  $p2$ . The circle becomes permanent, and a new rubberbanding circle appears through the cursor. Enter point  $p3$ . `doneProc` prints the donut points:

Donut entered was (  $p1$   $p2$   $p3$  ).

The enterfunction returns a list of the donut points:

(  $p1$   $p2$   $p3$  )



### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterEllipse

```
enterEllipse(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_bBox
```

#### Description

Enters an ellipse in the current window.

An ellipse is defined by its bounding box, therefore, only horizontally and vertically oriented ellipses can be entered.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to appear on the CIW prompt line. If supplied, only the first two strings are used. Each string corresponds to one of the two points needed to specify the bounding box of the ellipse.

*?points l\_pointList*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

List of no more than two points specifying the ellipse bounding box. If both points are specified, the box is returned after adjustment. If not, you are prompted for the remaining point or points. If one point is supplied, it is adjusted as if you had entered it. After you enter the first point, a rubberbanding ellipse is drawn inside the bounding box defined by the first point and the cursor's current position. After the second point is entered the function returns a list of two points. The list returned is normalized so that the first point corresponds to the lower left corner of the box; the second is the upper right corner.

`?form s_form`

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller

`?addPointProc t_addProcName`

Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`

Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

Determines whether or not to accept string input without terminating the enterfunction.

?acceptNumber *g\_acceptNumber*

Determines whether or not to accept numeric input without terminating the enterfunction.

?noInfix *g\_noInfix*

Determines whether or not to disable infix mode.

?cmdName *t\_cmdName*

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

?cursor *x\_cursor*

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

*l\_bBox*

The two digitized points as coordinates in user units. These points represent the bounding box containing the ellipse. It is normalized so that the first point corresponds to the lower left corner of the bounding box and the second is the upper right corner.

Retrieve data through the callback procedure *t\_doneProcName* instead of using the return value for this enterfunction.

### Examples

Displays the prompt Enter the first ellipse point on the CIW prompt line. When you enter point *p1* the prompt changes to Enter the second ellipse point. A rubberband ellipse appears with a bounding box constructed from *p1* and the current cursor location.

```
procedure(ellipseDone(w done pts)
 if(done then
 if(stringp(pts) then
 pts = list(pts "drawing")
 leGetEntryLayer(pts)
 printf("Current layer changed to %L.\n" pts)
 else
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

```
 printf("Ellipse entered was %L. Layer is %L.\n" pts
 leGetEntryLayer())
 else
 println("Ellipse entry aborted.")
)
)

enterEllipse(?prompts
 list("Enter the first ellipse point."
 "Enter the second ellipse point.")
 ?doneProc "circleDone" ?acceptString t
)
```

Type "thinox" into the CIW. doneProc notes the change:

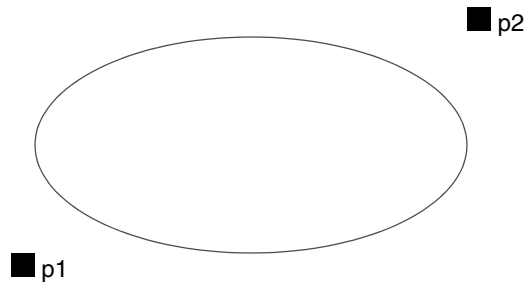
Current layer changed to ("thinox" "drawing").

The enterfunction is still active, as shown by the rubberband ellipse and the CIW prompt. Enter point p2. doneProc prints the ellipse points:

Ellipse entered was ( p1 p2 ). Layer is ("thinox" "drawing").

Here, (p1 p2) is the bounding box of the circle entered. The enterfunction returns the ellipse points:

( p1 p2 )



### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### **enterFunNestLevel**

```
enterFunNestLevel()
=> x_value
```

#### **Description**

Returns the current nesting level of the enter function.

#### **Arguments**

None

#### **Values Returned**

|                |                                              |
|----------------|----------------------------------------------|
| <i>x_value</i> | Current nesting level of the enter function. |
|----------------|----------------------------------------------|

#### **Examples**

The current nesting level of the enter function is 0.

```
enterFunNestLevel()
=> 0
```

#### ***Related Topics***

[User Entry Functions](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterLine

```
enterLine(
 [?prompts l_promptList]
 [?points l_pointList]
 [?wantPoints x_pointLimit]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
 [?target l_point]
)
=> l_point_list / nil
```

#### Description

Enters a multisegment line in the current window.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to be displayed in the CIW.

*?points l\_pointList*

List of points to start the line. If specified, all points are adjusted. If the list terminates with two identical points, the adjusted points are returned with no further interaction. Otherwise, the given points are displayed with connecting lines and a rubberbanding line is drawn from the last point to the current cursor position. Each successive point entered causes a new permanent segment between that point and the previous point.

The function terminates when two identical points are entered or when the function is explicitly terminated with [finishEnterFun](#) or [cancelEnterFun](#).

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?wantPoints x_pointLimit`

Number specifying the maximum number of input points to be used. The function returns the list of points after it reaches this number. In this case, the last two points do not need to be identical to terminate the function.

`?form s_form`

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`

Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`

Determines whether or not to draw rubberband lines.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`



## Cadence User Interface SKILL Reference

### User Entry Functions

---

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

`?target l_point`

The target point to which a flight line is drawn from the current cursor.

### Value Returned

`l_point_list`

The points you entered as a list containing the coordinates for a line.

`nil`

A nil list is returned.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt Enter the first point on the prompt line. When you enter point *p1*, the prompt changes to Enter the next point. A rubber band line appears between *p1* and the current cursor location.

```
procedure(lineDone(w done pts)
 if(done then
 printf("Line entered was %L.\n" pts)
 else
 println("Line entry aborted.")
)
)
enterLine(?prompts
 list("Enter the first point."
 "Enter the next point.")
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

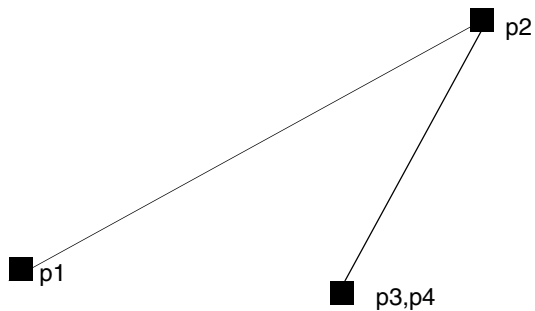
```
 ?doneProc "lineDone"
)
```

Enter point  $p2$  and the first segment becomes permanent, while the rubberbanding line now attaches the cursor to  $p2$ . Continue with  $p3$ . Enter  $p4$  at the same location as  $p3$ . `doneProc` prints the following point list:

```
Line entered was (p1 p2 p3 p4).
```

The `enterLine` function returns the point list:

```
(p1 p2 p3 p4)
```



### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

## enterNumber

```
enterNumber(
 [?prompts l_promptList]
 [?form s_form]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?alwaysMap g_alwaysMap]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> f_number
```

### Description

Prompts you to enter a number.

When you type a number and press *Enter*, any characters accumulated in the input line are returned as a number in floating-point format.

### Arguments

?prompts l\_promptList

List of prompt strings to be displayed in the CIW. Only the first prompt in the list is used.

?form s\_form

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the `enterfunction` and its caller. The form can be displayed any time before you press *Enter* to terminate the `enterfunction`.

?initProc t\_initProcName

Name of the SKILL function to be called immediately after the `enterfunction` is started and before any data entry.

?doneProc t\_doneProcName

Name of the SKILL function to be called when the `enterfunction` is terminated. The number is passed as the third argument to the done proc.

?formProc t\_formProcName

## Cadence User Interface SKILL Reference

### User Entry Functions

---

Name of the SKILL function to be called when the options form is displayed.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Values Returned

`f_number`

The number you typed in floating-point format.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

The prompt "Enter any number" is displayed on the CIW prompt line. When you type a number and press *Enter*, the enterfunction returns that number.

```
enterNumber(?prompts
 list("Enter any number.")
)
```

### Related Topics

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterPath

```
enterPath(
 [?prompts l_promptList]
 [?points l_pointList]
 [?wantPoints x_pointLimit]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?pathStyle t_pathStyle]
 [?pathWidth f_pathWidth]
 [?beginExtent f_beginExtent]
 [?endExtent f_endExtent]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_point_list / nil
```

#### Description

Enters a path in the current window.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to be displayed.

*?points l\_pointList*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

List of points to start the path. If specified, all points are adjusted. If the list terminates with two identical points, the adjusted points are returned with no further interaction. If not, the path is displayed using the given points, and a rubberbanding path segment is drawn from the last point to the current cursor position. Each successive point entered causes a new permanent segment between that point and the previous point. The function terminates when two identical points are entered, or when the function is explicitly terminated with [finishEnterFun](#) or [cancelEnterFun](#).

`?wantPoints x_pointLimit`

Number specifying the maximum number of points to be used. If the `x_pointLimit` argument is supplied, the function returns the list of points after it reaches this number. In this case, the last two points do not need to be identical to terminate the function.

`?form s_form`

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`

Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

`?pathStyle t_pathStyle`

## Cadence User Interface SKILL Reference

### User Entry Functions

---

String defining the path style. It can take one of the following values:

**T** or **Truncate**: The path is square ended with no extension.

**E** or **Extended**: The path is square ended with an extension half of the width on each end.

**R** or **Rounded**: The path is round ended with no extension.

**V** or **VariableEnd**: The path is square ended with its extension specified by the *f\_beginExtent* and *f\_endExtent* arguments.

?pathWidth *f\_pathWidth*

Width of the path in user units.

?beginExtent *f\_beginExtent*

Distance from the first point that the path extends to.

?endExtent *f\_endExtent*

Distance from the last point that the path extends to.

?dontDraw *g\_dontDraw*

Determines whether or not to draw rubber band lines.

?alwaysMap *g\_alwaysMap*

Determines whether or not to immediately display the options form.

?acceptString *g\_acceptString*

Determines whether or not to accept string input without terminating the enterfunction.

?acceptNumber *g\_acceptNumber*

Determines whether or not to accept numeric input without terminating the enterfunction.

?noInfix *g\_noInfix*

Determines whether or not to disable infix mode.

?cmdName *t\_cmdName*

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

*?cursor x\_cursor*      An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

#### Value Returned

*l\_point\_list*      The points you entered as a list containing the coordinates for a path.

*nil*      A nil list is returned.

Retrieve data through the callback procedure *t\_doneProcName* instead of using the return value for this enterfunction.

#### Examples

Displays the prompt Enter the first point on the CIW prompt line. When you enter point *p1* the prompt changes to Enter the next point. A rubber band path with width 6 appears between *p1* and the current cursor location.

```
procedure(pathDone(w done pts)
 if(done then
 printf("Path entered was %L.\n" pts)
 else
 println("Path entry aborted.")
)
)

enterPath(?prompts
 list("Enter the first point."
 "Enter the next point.")
 ?doneProc "pathDone"
 ?wantPoints 4
 ?pathWidth 6. ?pathStyle "Truncate"
)
```

Enter point *p2* and the first path segment becomes permanent, while the rubberbanding path segment now attaches the cursor to *p2*. The prompt remains the same because the second prompt in the list is used for any points after the first. Continue with *p3* and *p4*. *doneProc* prints out the list of points:

```
Path entered was (p1 p2 p3 p4).
```

The enterfunction returns the list of points:

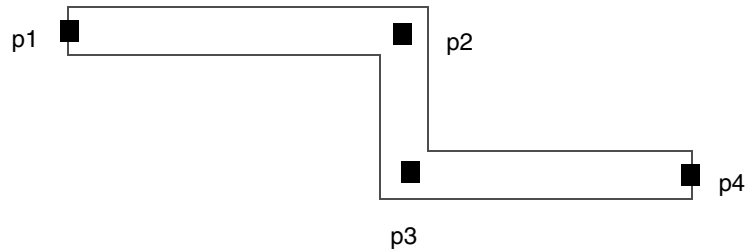


## Cadence User Interface SKILL Reference

### User Entry Functions

---

( p1 p2 p3 p4 )



#### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterPoint

```
enterPoint(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_coord_pair
```

#### Description

Prompts you to enter a single point.

#### Arguments

*?prompts l\_promptList*

List of prompt strings to be displayed. The first string in the list is displayed until you enter a point. Any other strings are ignored.

*?points l\_pointList*

List with no more than one point in it. If supplied, the point is adjusted and returned with no further interaction. If not supplied, the first string in *l\_promptList* is displayed on the CIW prompt line asking you to enter the point.

*?form s\_form*

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the `enterfunction` and its caller.

*?addPointProc t\_addProcName*

Name of the SKILL function to be called when a point is entered.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted. Since entering the first point terminates the function, this procedure is effectively never called.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### Value Returned

*l\_coord\_pair*                      The adjusted point you specified.

Retrieve data through the callback procedure *t\_doneProcName* instead of using the return value for this enterfunction.

#### Examples

Displays the prompt `Enter a point` is displayed on the prompt line.

```
procedure(pointDone(w done pts)
 if(done then
 printf("Point entered was %L.\n" car(pts))
 else
 println("Point entry aborted.")
)
)
enterPoint(?prompts
 list("Enter a point.")
 ?doneProc "pointDone"
)
```

The prompt `Enter a point` is displayed on the prompt line. Type the following into the CIW:

`2:6`

The enterfunction terminates, and *doneProc* prints out the point:

`Point entered was (2.000 6.000).`

The enterfunction returns the point:

`(2.000 6.000)`

#### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

## enterPoints

```
enterPoints(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?wantPoints x_pointLimit]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_coord_pair
```

## Description

Prompts you to enter a list of points.

## Arguments

*?prompts l\_promptList*

List of prompt strings to be displayed for successive points.

*?points l\_pointList*

List of points to be preloaded into the enterfunction. If this list is terminated by two identical points, the function returns the adjusted points with no further interaction. If not, any points in this list are added. The points are not displayed, and there is no rubber band line connecting any of the points. The function is terminated in the same manner as other multi-point enterfunctions.

*?form s\_form*

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

*?addPointProc t\_addProcName*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

`?wantPoints x_pointLimit`

Maximum number of points to be accepted by this function. Collects points until the number of points specified by `x_pointLimit` is reached or until you finish or cancel the enterfunction. Nothing is drawn by the enterfunction.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?cursor x_cursor`      An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

`l_coord_pair`      The adjusted points entered as a list of points.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt `Enter a point` is displayed on the prompt line.

```
procedure(pointsDone(w done pts)
 if(done then
 printf("Points entered were %L.\n" pts)
 else
 println("Point entry aborted.")
)
)
enterPoints(?prompts
 list("Enter the first point."
 "Enter the next point.")
 ?doneProc "pointsDone"
)
```

When you enter point *p1* the prompt changes to `Enter the next point` Unlike `enterLine`, no rubber band line appears between *p1* and the current cursor location.

Enter point *p2*. The prompt reads the same—`Enter the next point`—because the second prompt in the list is used for any points after the first. Continue with *p3*.

```
finishEnterFun()
```

`doneProc` prints the points entered:

```
Points entered were (p1 p2 p3).
```

The enterfunction returns the point list:

```
(p1 p2 p3)
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)



## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterPolygon

```
enterPolygon(
 [?prompts l_promptList]
 [?points l_pointList]
 [?wantPoints x_pointLimit]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_point_list / nil
```

#### Description

Digitizes a polygon in the current window.

The function does not allow entry of points that result in a self-intersecting polygon. It also removes any co-linear points.

#### Arguments

?prompts *l\_promptList*

List of prompt strings to be displayed.

?points *l\_pointList*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

List of points to start the polygon. If specified, all points are adjusted. If the list terminates with a point identical to the first point, the adjusted points are returned with no further interaction. If not, the given points are displayed with connecting lines, and a rubberbanding line is drawn from the last point to the current cursor position. Each successive point entered causes a new permanent segment between that point and the previous point. The function terminates when a point identical to the first point is entered or when the function is explicitly terminated with [finishEnterFun](#) or [cancelEnterFun](#).

?wantPoints *x\_pointLimit*

Maximum number of points to be specified. If the *x\_pointLimit* argument is supplied, the function returns the list of points after it reaches this number. In this case, the last two points do not need to be identical to terminate the function.

?form *s\_form*

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

?addPointProc *t\_addProcName*

Name of the SKILL function to be called when a point is entered.

?delPointProc *t\_delProcName*

Name of the SKILL function to be called when a point is deleted.

?initProc *t\_initProcName*

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

?doneProc *t\_doneProcName*

Name of the SKILL function to be called when the enterfunction is terminated.

?formProc *t\_formProcName*

Name of the SKILL function to be called when the options form is displayed.

?dontDraw *g\_dontDraw*

## Cadence User Interface SKILL Reference

### User Entry Functions

---

Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

`l_point_list`

The list of points that have been entered. If the function is terminated by returning to the initial point, that point appears at the beginning *and* at the end of the returned list.

`nil`

A nil list is returned.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt `Enter the first point` on the prompt line. When you enter point `p1` the prompt changes to `Enter the next point`. No rubber band line appears between `p1` and the current cursor location because the `dontDraw` flag is specified.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

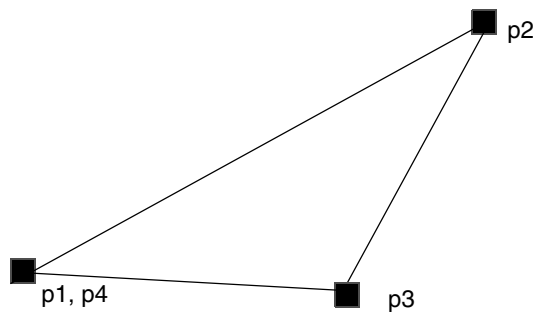
```
procedure(polyDone(w done pts)
 if(done then
 printf("Polygon entered was %L.\n" pts)
 dbCreatePolygon(w->editCellView
 list("cut" "drawing")
 pts)
 else
 println("Polygon entry aborted.")
)
)
enterPolygon(?prompts
 list("Enter the first point."
 "Enter the next point.")
 ?doneProc "polyDone"
 ?dontDraw t
)
```

Enter point  $p2$ . The prompt reads the same—"Enter the next point"—because the second prompt in the list is used for any points after the first. Continue with  $p3$ . Now enter  $p4$  at the same location as  $p1$ . `doneProc` creates the polygon on the `cut drawing` layer and prints out the points entered. Here  $p4$  is the same point as  $p1$ .

The polygon entered was (  $p1$   $p2$   $p3$   $p4$  ).

The enterfunction returns the point list:

(  $p1$   $p2$   $p3$   $p4$  )



### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterScreenBox

```
enterScreenBox ()
 => l_bBox
```

#### Description

Lets you enter a box in screen coordinates.

Most enterfunctions return coordinates in user units, so this function provides a way to specify screen coordinates when necessary. When `enterScreenBox` is invoked, it displays a small box on the root window (denoting that it is in `ScreenBox` mode). Press the mouse button at the first point and hold down. While the button is down a rubberbanding box is displayed. When you release the mouse button, the box is returned. This is different from the behavior of `enterBox`, which uses two clicks, one for each corner.

`enterScreenBox` is not nestable and suspends all other enterfunctions until it completes.

#### Argument

None

#### Value Returned

*l\_bBox*                      The bounding box in screen coordinates that you entered.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

#### Examples

Changes the cursor to a small box.

```
enterScreenBox ()
```

Press the mouse button at *p1*. While the button is down, a rubberbanding box connects *p1* to the current cursor location. Release the mouse button at the desired location.

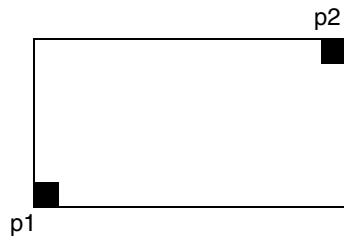
The function terminates and returns the selected points in screen coordinates:

## Cadence User Interface SKILL Reference

### User Entry Functions

---

( p1 p2 )



#### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

## enterSegment

```
enterSegment (
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_point_list / nil
```

## Description

Lets you enter a segment into the current window.

The function accepts two points. After the first point is digitized, a rubberbanding line appears between the first point and the cursor's current location.

## Arguments

*?prompts l\_promptList*

List of prompt strings to be displayed. Only the first two strings are used.

*?points l\_pointList*

List of no more than two points. If both points are supplied, they are adjusted and returned with no further interaction. If one point is supplied, it is adjusted and displayed as if the user had entered it. After the first point is entered a rubberbanding line appears between the point and the current cursor location. When the second point is entered, the function terminates and returns the list of points entered. The segment obeys the current snap mode and snap shape rule when in Graphics Editor windows.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?form s_form` Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`  
Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`  
Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`  
Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`  
Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`  
Name of the SKILL function to be called when the options form is displayed.

`?dontDraw g_dontDraw`  
Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`  
Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`  
Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`  
Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`  
Determines whether or not to disable infix mode.

`?cmdName t_cmdName`



## Cadence User Interface SKILL Reference

### User Entry Functions

---

|                               |                                                                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | Name you associate with the enterfunction. If the command name is set, it can be accessed with <code>hiGetCurrentCmd</code> .                                                                                                                                                  |
| <code>?cursor x_cursor</code> | An integer argument that will default to an appropriate cursor. For example, <code>hicArrowRectangle</code> for <code>enterBox</code> and <code>hicArrowArc</code> for <code>enterArc</code> . For more information on all available cursors see <a href="#">hiSetCursor</a> . |

### Values Returned

|                           |                                                  |
|---------------------------|--------------------------------------------------|
| <code>l_point_list</code> | A list containing the coordinates for a segment. |
| <code>nil</code>          | A nil list is returned.                          |

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt Enter the second point on the prompt line.

```
procedure(segDone(w done pts)
 if(done then
 printf("Segment entered was %L.\n" pts)
 else
 println("Segment entry aborted.")
)
)
procedure(myEnterSegment()
 enterSegment(?prompts
 list("Enter the first point."
 "Enter the second point.")
 ?doneProc "segDone"
)
)
```

Sets infix mode on and bind the `s` key to the procedure `myEnterSegment`. Type the `s` key in a Schematics window at location `p1`. Because of infix mode, `p1` is accepted as the first point for the `enterSegment` call. The rubberbanding line immediately appears between `p1` and the cursor location.

```
window(1)->infixOn = t
hiSetBindKey("Schematics" "<Key>s" "myEnterSegment()")
```

Enter point `p2`. `doneProc` prints out the points entered:

```
Segment entered was (p1 p2).
```

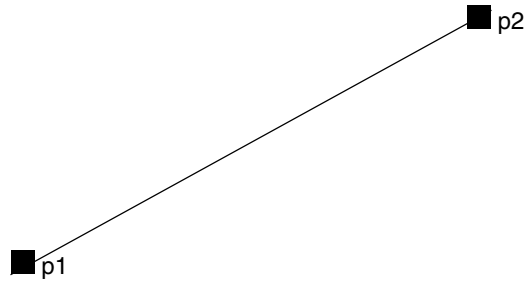
The enterfunction returns the list of points:

## Cadence User Interface SKILL Reference

### User Entry Functions

---

( p1 p2 )



#### ***Related Topics***

User Entry Functions

hiCreateOptionsForm

undrawEnterFun

hiGetCurrentCmd

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### enterMultiRep

```
enterMultiRep(
 [?prompts l_promptList]
 [?points l_pointList]
 [?form s_form]
 [?addPointProc t_addProcName]
 [?delPointProc t_delProcName]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?dontDraw g_dontDraw]
 [?alwaysMap g_alwaysMap]
 [?acceptString g_acceptString]
 [?acceptNumber g_acceptNumber]
 [?noInfix g_noInfix]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> l_point_list / nil
```

#### Description

Identical to `enterSegment` except that the two points can be in windows viewing different cellviews.

This is the only enterfunction that allows this capability. All others accept input only in windows viewing the same Cellview (known as *equivalent* windows).

#### Arguments

?prompts l\_promptList

List of prompt strings to be displayed in the CIW. Only the first two strings are used.

?points l\_pointList

## Cadence User Interface SKILL Reference

### User Entry Functions

---

List of no more than two points. If both points are supplied, they are adjusted and returned with no further interaction. In this case, the two points are both assumed to be in the current window. If one point is supplied, it is adjusted and displayed as if the user had entered it. After the first point is entered a rubberbanding line appears between the point and the current cursor location.

The rubberbanding line appears only while the cursor is in windows equivalent to the window in which the first point was entered. When the second point is entered, the function terminates and returns the list of points entered. If the second point is entered in a window that is not equivalent to the first, it is returned in coordinates corresponding to the window in which it was entered. Normally this is not a problem. If the two windows, however, have different database-to-user-unit conversion factors, the result might not be as expected. The segment obeys the current snap mode and snap shape rule when in Graphics Editor windows.

`?form s_form`

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

`?addPointProc t_addProcName`

Name of the SKILL function to be called when a point is entered.

`?delPointProc t_delProcName`

Name of the SKILL function to be called when a point is deleted.

`?initProc t_initProcName`

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

`?doneProc t_doneProcName`

Name of the SKILL function to be called when the enterfunction is terminated.

`?formProc t_formProcName`

Name of the SKILL function to be called when the options form is displayed.

## Cadence User Interface SKILL Reference

### User Entry Functions

---

`?dontDraw g_dontDraw`

Determines whether or not to draw rubber band lines.

`?alwaysMap g_alwaysMap`

Determines whether or not to immediately display the options form.

`?acceptString g_acceptString`

Determines whether or not to accept string input without terminating the enterfunction.

`?acceptNumber g_acceptNumber`

Determines whether or not to accept numeric input without terminating the enterfunction.

`?noInfix g_noInfix`

Determines whether or not to disable infix mode.

`?cmdName t_cmdName`

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

`?cursor x_cursor`

An integer argument that will default to an appropriate cursor. For example, `hiArrowRectangle` for `enterBox` and `hiArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

`l_point_list`

A list of two points.

`nil`

A nil list is returned.

Retrieve data through the callback procedure `t_doneProcName` instead of using the return value for this enterfunction.

### Examples

Displays the prompt Enter the second point in the CIW prompt line.

```
procedure(segDone(w done pts)
 if(done then
 printf("Segment extends from %L to %L.\n"
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

```
 w hiGetCurrentWindow()
 else
 println("Segment entry aborted.")
)
)

procedure(segInit()
 w = hiGetCurrentWindow()
 firstWindow =
 evalstring(cadr(parseString(sprintf(nil "%L" w) ":")))
)
procedure(myEnterSegment()
 segInit()
 enterMultiRep(
 ?prompts list("Enter the first point."
 "Enter the second point.")
 ?doneProc "segDone"
 ?noInfix t
)
)
```

Sets the infix mode on and bind the **s** key to the procedure `myEnterSegment`. Type the **s** key in a Schematics window at location *p1*. Because of the `noInfix` flag, the `infixOn` flag is ignored. The prompt "Enter the first point" is displayed in the CIW. Enter point *p1* in window(2). The rubberbanding line appears between *p1* and the cursor location.

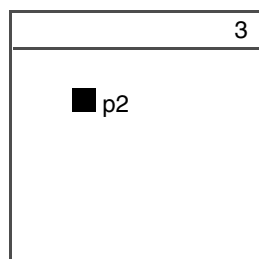
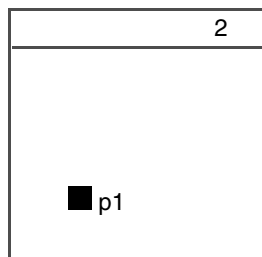
```
window(1)->infixOn = t
hiSetBindKey("Schematics" "<Key>s" "myEnterSegment()")
```

Enter point *p2* in window(3), which views a different cellview. `doneProc` prints out the specified message:

Segment extends from window:2 to window:3.

The enterfunction returns the point list:

```
(p1 p2)
```



## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

## enterString

```
enterString(
 [?prompts l_promptList]
 [?form s_form]
 [?initProc t_initProcName]
 [?doneProc t_doneProcName]
 [?formProc t_formProcName]
 [?alwaysMap g_alwaysMap]
 [?cmdName t_cmdName]
 [?cursor x_cursor]
)
=> t_string
```

## Description

Prompts you to enter a string.

The function terminates when you press a carriage return in the CIW.

## Arguments

*?prompts l\_promptList*

List of prompt strings to be displayed in the CIW. Only the first string is used.

*?form s\_form*

Options form created with `hiCreateOptionsForm`. This form is displayed when you call `hiToggleEnterForm`. It contains options relevant to the enterfunction and its caller.

*?initProc t\_initProcName*

Name of the SKILL function to be called immediately after the enterfunction is started and before any data entry.

*?doneProc t\_doneProcName*

Name of the SKILL function to be called when the enterfunction is terminated.

*?formProc t\_formProcName*

Name of the SKILL function to be called when the options form is displayed.

*?alwaysMap g\_alwaysMap*



## Cadence User Interface SKILL Reference

### User Entry Functions

---

Determines whether or not to immediately display the options form.

?cmdName *t\_cmdName*

Name you associate with the enterfunction. If the command name is set, it can be accessed with `hiGetCurrentCmd`.

?cursor *x\_cursor*

An integer argument that will default to an appropriate cursor. For example, `hicArrowRectangle` for `enterBox` and `hicArrowArc` for `enterArc`. For more information on all available cursors see [hiSetCursor](#).

### Value Returned

*t\_string*

The string you typed.

Retrieve data through the callback procedure *t\_doneProcName* instead of using the return value for this enterfunction.

### Examples

```
enterString(?prompts list("Enter layer name."))
```

The prompt “Enter layer name” is displayed on the prompt line. When you type a string followed by a carriage return, the enterfunction returns that string.

### ***Related Topics***

[User Entry Functions](#)

[hiCreateOptionsForm](#)

[undrawEnterFun](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### finishEnterFun

```
finishEnterFun(
 [g_form]
)
=> t / nil
```

#### Description

Completes an enterfunction if possible.

Some commands cannot be finished until you enter all the required points. When `finishEnterFun` is called in such a situation, the enterfunction is canceled even though `finishEnterFun` returns `t`.

Do not call `finishEnterFun` from within an enterfunction callback procedure.

#### Argument

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>g_form</i> | Form handle used to locate the enterfunction to finish. Normally the enterfunction finished is the one active in the current window. If you want to finish a specific enterfunction, whether or not it is the current window, there is no way to tell which window is the correct one. So you pass the same form that was originally supplied to the enterfunction to <code>finishEnterFun</code> , and it finishes that specific enterfunction regardless of whether it is in the current window or not. |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Values Returned

|                  |                                                         |
|------------------|---------------------------------------------------------|
| <code>t</code>   | The enterfunction in the current window was terminated. |
| <code>nil</code> | Could not terminate the enterfunction.                  |

#### Examples

Displays the prompt `First` is displayed in the CIW.

```
enterPoints(?prompts list("First" "Next"))
```

Enter points *p1*, *p2*, and *p3*. Now type

```
finishEnterFun()
```

## Cadence User Interface SKILL Reference

### User Entry Functions

---

The enterfunction is terminated, the prompt goes away, and ( *p1 p2 p3* ) is returned.

#### ***Related Topics***

User Entry Functions

Enterfunction Flags

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### hiGetCurrentCmd

```
hiGetCurrentCmd(
 w_window
)
=> t_commandName / nil
```

#### Description

Returns the command name associated with the enterfunction currently running in a window, or `nil` if no enterfunction is running.

#### Argument

|                       |            |
|-----------------------|------------|
| <code>w_window</code> | Window ID. |
|-----------------------|------------|

#### Values Returned

|                            |                                                                       |
|----------------------------|-----------------------------------------------------------------------|
| <code>t_commandName</code> | Command name associated with the enterfunction running in the window. |
| <code>nil</code>           | An enterfunction is not running in the window.                        |

#### ***Related Topics***

[User Entry Functions](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### hiInEnterFun

```
hiInEnterFun(
 [w_wtype]
)
=> s_enterFunction / nil
```

#### Description

Returns the symbol of the enterfunction pending in the window.

#### Argument

|                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| <i>w_wtype</i> | Window ID. If you do not specify this argument, the current window is used. |
|----------------|-----------------------------------------------------------------------------|

#### Values Returned

|                        |                                                                               |
|------------------------|-------------------------------------------------------------------------------|
| <i>s_enterFunction</i> | Symbol of the enterfunction pending in the window. For example, 'enterPoints. |
| <i>nil</i>             | No enterfunction is pending in the window.                                    |

#### ***Related Topics***

[User Entry Functions](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### hiMarkNestable

```
hiMarkNestable(
 g_symbolOrList
)
=> t / nil
```

#### Description

Makes enterfunctions nestable. When an enterfunction is running and a new one is called, the new enterfunction replaces the old enterfunction in the stack. When the new enterfunction is finished or canceled, the old enterfunction is resumed. This function reverses the effect of `hiMarkNonNestable`.

#### Argument

|                       |                                                                    |
|-----------------------|--------------------------------------------------------------------|
| <i>g_symbolOrList</i> | Symbol or list of symbols representing one or more enterfunctions. |
|-----------------------|--------------------------------------------------------------------|

#### Values Returned

|     |                                   |
|-----|-----------------------------------|
| t   | The functions were made nestable. |
| nil | There was an error.               |

#### ***Related Topics***

[User Entry Functions](#)

[hiMarkNonNestable](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### hiMarkNonNestable

```
hiMarkNonNestable(
 g_symbolOrList
)
=> t / nil
```

#### Description

Makes enterfunctions non-nestable. When an enterfunction is running and a new one is called, the new enterfunction replaces it in the stack. When the new enterfunction is finished or canceled, the enterfunction it replaced will not be returned to, but will have been canceled by the new enterfunction.

#### Argument

|                       |                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------|
| <i>g_symbolOrList</i> | Symbol or list of symbols representing one or more functions, presumably enterfunctions. |
|-----------------------|------------------------------------------------------------------------------------------|

#### Values Returned

|     |                          |
|-----|--------------------------|
| t   | The operation succeeded. |
| nil | The operation failed.    |

#### ***Related Topics***

[User Entry Functions](#)

[hiGetCurrentCmd](#)

[hiMarkNestable](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

## hiToggleEnterForm

```
hiToggleEnterForm(
 [g_form]
)
=> t / nil
```

### Description

Displays or removes the options form of the enterfunction in the current window.

If the options form was not displayed, it is now displayed; if the options form was displayed, it is removed. If the enterfunction does not have an options form, or if no enterfunction is active, a warning dialog box is displayed.

### Argument

|               |                                                                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>g_form</i> | Options form to toggle on or off. This allows you to toggle an enterfunction form regardless of whether it is associated with the current window. |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|

### Values Returned

|     |                                                 |
|-----|-------------------------------------------------|
| t   | The display status of the form was changed.     |
| nil | The display status of the form was not changed. |

### Examples

Creates `circleForm` that contains options for your circle.

```
enterCircle(?prompts list("Center" "Radius") ?form circleForm)
```

Now type

```
hiToggleEnterForm(circleForm)
```

The form pops up, and you can set options. Type it again and the form disappears again.

### ***Related Topics***

[User Entry Functions](#)



## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### hiUpdate

```
hiUpdate()
=> t
```

#### Description

Updates window prompts and application-specific state variables. Especially important if an enterfunction has been canceled in a window that is not the current window.

#### Argument

None

#### Values Returned

|   |                         |
|---|-------------------------|
| t | The update is finished. |
|---|-------------------------|

#### ***Related Topics***

[User Entry Functions](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### preXY

```
preXY(
 l_point
)
=> t
```

#### Description

Causes an `addPoint( l_point )` command to be added to the command queue in order to be logged and executed at the next toplevel to simulate the user selecting a point.

#### Argument

|                |                    |
|----------------|--------------------|
| <i>l_point</i> | Point to be added. |
|----------------|--------------------|

#### Values Returned

|   |                                      |
|---|--------------------------------------|
| t | Adds the point command to the queue. |
|---|--------------------------------------|

#### Examples

This example enters a polygon of five points where the first three points are already calculated but the `addPointProc` procedure is called for all the points, even the first three.

```
MyPointList = { preXY(0.0:0.0) preXY(5.0:0.0) preXY(5.0:5.0)
 enterPolygon(?prompts list("enter point")
 ?wantPoints 5
 ?addPointProc "MyAddPointProc"
)
}
```

#### ***Related Topics***

[User Entry Functions](#)

[hiGetCurrentCmd](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### setEFunFlightLineTarget

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

#### Description

Sets a target point for the `enterLine` function so that a flight line is drawn from the cursor to this point.

#### Argument

|                |                                                 |
|----------------|-------------------------------------------------|
| <i>l_point</i> | Point in user units to set as the target point. |
|----------------|-------------------------------------------------|

#### Values Returned

|            |                          |
|------------|--------------------------|
| <i>t</i>   | Target point is set.     |
| <i>nil</i> | Target point is not set. |

#### Examples

Sets the target point at 1:0.

```
enterLine()
setEFunFlightLineTarget(1:0)
```

#### ***Related Topics***

[User Entry Functions](#)

[enterLine](#)

## Cadence User Interface SKILL Reference

### User Entry Functions

---

#### undrawEnterFun

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

#### Description

Undraws any rubberbanding shapes. Called when an enterfunction is active. If you enter data in the window, rubberbanding resumes.

#### Argument

None

#### Values Returned

|     |                                                           |
|-----|-----------------------------------------------------------|
| t   | The undraw is successful.                                 |
| nil | There was no enterfunction pending in the current window. |

#### Examples

The prompt "Next" appears in the CIW, and line segments connecting *p1* and *p2*, and *p2* and *p3* appear on the screen.

```
enterPolygon(?prompts list("First" "Next")
 ?pointlist list(p1 p2 p3))
```

Erases the segments, but the prompt remains to signify the enterfunction is still active.

```
undrawEnterFun()
```

#### ***Related Topics***

User Entry Functions

hilnEnterFun

---

## Encapsulation Window

---

This topic provides a list of Cadence® SKILL functions associated with Encapsulation window. An Encapsulation window is similar to the CIW except that it does not have a mouse binding display.

The optional window manager border contains the title Encapsulation. You can replace this name using the SKILL functions `hiSetWindowName` or `hiEncap`. The banner contains pulldown menus providing the default system menu that contains the SKILL `hiRegCloseProc` function.

Each command line is copied into the output area from the input line as it is accepted for execution. The input command line is then placed with the user-specifiable SKILL command wrapper through the `hiSetEncapSkillCmd` command. The SKILL command and any error messages or output produced by that command are displayed in the CIW output area. Only the input history of the encapsulation session is copied into the Encapsulation output window.

The input window behaves like the input window of the CIW.

You control the size and location of the Encapsulation window. The display manager defaults to a certain minimum window size.

The following properties can be set on an encapsulation window:

- `sendAllInput`
- `noAutoNewline`
- `obeySpecialChars`

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- `hiAppendInputCmd`
- `hiEncap`
- `hiFocusToEncap`

## Cadence User Interface SKILL Reference

### Encapsulation Window

---

- hiGetEncapSkillCmd
- hiSetEncapHistory
- hiSetEncapPrompt
- hiSetEncapSkillCmd

## hiAppendInputCmd

```
hiAppendInputCmd(
 w_window
 t_cmd
)
=> t / nil
```

### Description

Appends a command to the input area of an encapsulation window.

If the encapsulation window has the property `obeySpecialChars` set to `t`, this function treats the character `'\n'` in `t_cmd` as a newline (equivalent to pressing the carriage return key in the encapsulation window). If this property is not set, the `'\n'` and any characters following this in `t_cmd` are ignored.

### Arguments

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <code>w_window</code> | Encapsulation window ID returned from <code>hiEncap</code> . |
| <code>t_cmd</code>    | SKILL function string to append to the input area.           |

### Values Returned

|                  |                                                                                                                                           |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | The string is appended to the input area.                                                                                                 |
| <code>nil</code> | The window specified by <code>w_window</code> is invalid or if the function <code>t_cmd</code> is null. A warning message is also issued. |

### Examples

```
w = hiEncap(list(100:100 500:500) "" "encap")
w->obeySpecialChars = t
hiAppendInputCmd(w "ab") ; input has 'ab'
hiAppendInputCmd(w "cd") ; input has 'abcd'
hiAppendInputCmd(w "\n") ; sends 'abcd' to output
hiAppendInputCmd(w "ab\ncd\nef")
 ; sends 'ab' to output
 ; sends 'cd' to output
 ; input has 'ef'
hiAppendInputCmd(w "gh\n") ; sends 'efgh' to output
```

## Cadence User Interface SKILL Reference

### Encapsulation Window

---

#### ***Related Topics***

[Encapsulation Window](#)

[hiEncap](#)



## Cadence User Interface SKILL Reference

### Encapsulation Window

---

## hiEncap

```
hiEncap(
 [g_boxSpec]
 [t_cmd]
 [t_title]
)
=> w_window / nil
```

## Description

Invokes an encapsulation session by creating and displaying an encapsulation window.

The window created has a widget type of `encap` and an application type of `encap`.

If you get the warning message related to allocation of colormap entry, you can ignore it. This message is displayed if you have set `CDS_USE_PRIVATE_CMAP` to `true` or have used a private colormap. It does not affect the encapsulation window.

## Arguments

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>g_boxSpec</i> | Screen coordinates specifying the lower left and upper right corners of the window, assuming 0:0 to be the lower left corner of the screen. If <i>boxSpec</i> is <code>nil</code> , either you are prompted with a rubberbanding box to enter a window size or a default size is used depending on the value of the global variable <code>hivWinStyle</code> . The <code>'default</code> and <code>'interactive</code> symbols are also acceptable values for the <i>boxSpec</i> argument. |
| <i>t_cmd</i>     | SKILL function to which all input sent to the Encapsulation window is redirected. This function acts as a wrapper function to the input command line entered during the current encapsulation session.                                                                                                                                                                                                                                                                                     |
| <i>t_title</i>   | String that appears in the window manager banner of the encapsulation window. If not specified, the string <code>Encapsulation</code> appears.                                                                                                                                                                                                                                                                                                                                             |

## Cadence User Interface SKILL Reference

### Encapsulation Window

---

#### Values Returned

|                       |                                                                 |
|-----------------------|-----------------------------------------------------------------|
| <code>w_window</code> | Opens an encapsulation window and returns the ID of the window. |
| <code>nil</code>      | Cannot open a window.                                           |

#### Examples

Opens an encapsulation window and returns its `windowId`.

```
hiEncap()
=> window:2
```

#### ***Related Topics***

[Encapsulation Window](#)

[hiSetEncapSkillCmd](#)

## Cadence User Interface SKILL Reference

### Encapsulation Window

---

## hiFocusToEncap

```
hiFocusToEncap(
 w_encap
 [w_window]
)
=> t / nil
```

### Description

Temporarily focuses the keyboard input to an encapsulation window.

The keyboard focus to the encapsulation window is maintained until a carriage return is entered or until the pointer focus is moved from one window into another.

### Arguments

|                 |                                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------|
| <i>w_encap</i>  | Encapsulation window ID returned from <code>hiEncap</code> .                                             |
| <i>w_window</i> | Window from which the keyboard focus is being transferred. If not specified, the current window is used. |

### Values Returned

|            |                                                                |
|------------|----------------------------------------------------------------|
| <i>t</i>   | The focus is transferred to the encapsulation window.          |
| <i>nil</i> | The focus is not transferred. An error message is also issued. |

### ***Related Topics***

[Encapsulation Window](#)

[hiEncap](#)

## Cadence User Interface SKILL Reference

### Encapsulation Window

---

#### hiGetEncapSkillCmd

```
hiGetEncapSkillCmd(
 w_window
)
=> t_skillCmd
```

#### Description

Returns the SKILL function associated with a window.

#### Argument

|                 |                                                              |
|-----------------|--------------------------------------------------------------|
| <i>w_window</i> | Encapsulation window ID returned from <code>hiEncap</code> . |
|-----------------|--------------------------------------------------------------|

#### Value Returned

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <i>t_skillCmd</i> | The SKILL wrapper function string associated with the encapsulation window. |
|-------------------|-----------------------------------------------------------------------------|

#### Examples

Returns the SKILL function `processString` for `w`.

```
hiGetEncapSkillCmd(w)
=> "processString"
```

#### ***Related Topics***

[Encapsulation Window](#)

[hiEncap](#)

## hiSetEncapHistory

```
hiSetEncapHistory(
 w_window
 t_msg
 [g_redraw]
)
=> t / nil
```

### Description

Copies a message into the next available insertion position in an encapsulation output window.

### Arguments

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_window</i> | Encapsulation window ID returned from <code>hiEncap</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <i>t_msg</i>    | String that you want appended (inserted) to the last line of the output portion of the encapsulation window.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <i>g_redraw</i> | <p>Determines whether a message sent to the encapsulation output window is displayed or saved. If set to <code>nil</code>, messages sent to the encapsulation window are buffered (saved) and are not displayed until a newline character is explicitly specified in <i>t_msg</i>. For performance reasons, this flag should be set when sending long strings to the window. If set to <code>t</code> (the default), all messages sent to the encapsulation window are immediately displayed with a newline automatically appended.</p> <p>If the property <code>noAutoNewline</code> on the encapsulation window is set to <code>t</code>, the newline is not automatically appended to the <i>t_msg</i> string, and a <code>'\n'</code> character must be explicitly specified in the string before the string is sent and displayed to the window output.</p> |

### Values Returned

|                  |                                                                                        |
|------------------|----------------------------------------------------------------------------------------|
| <code>t</code>   | The string is copied to the encapsulation output window.                               |
| <code>nil</code> | The string is not copied to the encapsulation output window. A warning is also issued. |

## Cadence User Interface SKILL Reference

### Encapsulation Window

---

#### ***Related Topics***

[Encapsulation Window](#)

[hiEncap](#)

## hiSetEncapPrompt

```
hiSetEncapPrompt (
 w_window
 t_prompt
)
=> t / nil
```

### Description

Sets the prompt for an encapsulation window.

### Arguments

|                 |                                                                                        |
|-----------------|----------------------------------------------------------------------------------------|
| <i>w_window</i> | Encapsulation window ID returned by <code>hiEncap</code> .                             |
| <i>t_prompt</i> | String to be used as the prompt. The default prompt is the greater than character (>). |

### Value Returned

|            |                                                         |
|------------|---------------------------------------------------------|
| <i>t</i>   | The prompt is set.                                      |
| <i>nil</i> | The prompt is not set. An error message is also issued. |

### Examples

Returns *t* and displays the prompt "What\_now?" directly above the input line in the encapsulation window.

```
hiSetEncapPrompt (w "What_now?")
```

### ***Related Topics***

[Encapsulation Window](#)

[hiEncap](#)

## hiSetEncapSkillCmd

```
hiSetEncapSkillCmd(
 w_windowId
 t_skillCmd
)
=> t / nil
```

### Description

Registers the SKILL function that wraps or embraces the input command line entered during the current encapsulation session.

Any input entered in the input window of the encapsulation window is redirected to the SKILL function *t\_skillCmd*. This function should accept only one argument—the string that is the line of input to the encapsulation window.

If the property *sendAllInput* is set to *t* on the encapsulation window, an empty string is passed to the *t\_skillCmd* function. If the property is not set, the default action is that single carriage returns are ignored.

### Arguments

|                   |                                                      |
|-------------------|------------------------------------------------------|
| <i>w_windowId</i> | Encapsulation window ID returned by <i>hiEncap</i> . |
| <i>t_skillCmd</i> | SKILL function to use as the wrapper function.       |

### Values Returned

|            |                                                           |
|------------|-----------------------------------------------------------|
| <i>t</i>   | The function is registered.                               |
| <i>nil</i> | The function is not registered. A warning is also issued. |

### Examples

Create the procedure to be used as the wrapper:

```
procedure(processString(input "t")
;process the input string typed to the encapsulation
;window
...
)
```

Create an encapsulation window:



## Cadence User Interface SKILL Reference

### Encapsulation Window

---

```
w = hiEncap(list(100:100 500:500))
hiSetEncapSkillCmd(w "processString")
```

When input is typed in the encapsulation window:

```
countAll
```

the wrapper function is set up, using the encapsulation window input as its argument:

```
processString("countAll")
```

The `processString` function then processes the input string and either redirects it to another process or calls the appropriate function.

### ***Related Topics***

[Encapsulation Window](#)

[hiEncap](#)

## **Cadence User Interface SKILL Reference**

### Encapsulation Window

---

---

## Functions of Viewfile Window

---

This topic provides a list of Cadence® SKILL functions associated with viewfile window functions that are used to manipulate text, hypertext, and HTML view file windows.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiClearClipboard](#)
- [hiClearCurrentTextSelection](#)
- [hiDisableTailViewfile](#)
- [hiEnableTailViewfile](#)
- [hiGenTextIndex](#)
- [hiViewfile](#)
- [hiViewTextFile](#)
- [hiSetViewfile](#)
- [hiSaveViewfile](#)
- [hiSaveAsViewfile](#)
- [hiStartGenTextIndex](#)
- [hiGetTextSelection](#)
- [hiGetTextSelByLoc](#)
- [hiSetTextSelection](#)
- [hiSelectTextByLoc](#)
- [hiGetCurrentTextSelection](#)
- [hiGetCurrentTextSelectionLoc](#)

## **Cadence User Interface SKILL Reference**

### Functions of Viewfile Window

---

- hiGetCurrentTextSelectionLoc
- hiSetCurrentTextSelectionLoc
- view

## Viewfile Management

A text viewfile window also known as a *showFile* window is a window that lets you view the text output. Viewfiles allow you to choose a file to work with, save a file under a different name, find a particular text string, and maneuver within the viewfile window using various criteria. You can also work with a selected class and enable or disable updating from a changing file.

The windows must have a widget type of `text`, `hypertext`, or `html` and should have been created either from a call to `view` or by a call to `hiCreateWindow` with the widget type set to `text`, `hypertext`, or `html`.

### HTML Viewfile Window

Use HTML viewfile windows instead of hypertext viewfile windows.

An HTML viewfile window displays HTML files. It has a widget type of `html`. Links to Web pages will be opened in the browser specified in the *Web Browser* field in the User Preferences form in CIW or the corresponding `ui webBrowser` environment variable in your `.cdsenv` file. The default browser is FireFox.

The HTML window links cannot be used for external links ("`http`" or "`https`") or email ("`mailto`"). They can only be used to view internal files.

Support for external links can be added using a hypertext field that can be added using the `hiCreateHypertextField` SKILL function in an HI form or form window.

You can use the following Opus resources to control the hyperlink colors in the HTML viewfile window:

---

|                                          |                                                    |
|------------------------------------------|----------------------------------------------------|
| <code>Opus.hypertextColor</code>         | Text color of unvisited hyperlink                  |
| <code>Opus.hypertextSelectedColor</code> | Text color of visited hyperlink                    |
| <code>Opus.hypertextFocusedColor</code>  | Text color of hyperlink when the cursor is over it |

---

### Text Device Mode

As with most text editors, when a text file is opened in the viewfile window, the entire file is loaded from disk and retained in memory. This is referred to as `LoadAll` mode. In this mode, the viewfile window continues to display the file content even if the file is deleted from the disk. However, as the file size increases, the initial read-in time and memory requirements also increase, leading to reduced performance. In such cases, you can use `Sparse` mode, as follows:

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

```
(hiSetViewfile win "./largeFile.txt" 'Sparse)
```

When a file is opened in *Sparse* mode, only minimal data is read from disk initially. More data is paged from disk and the old data is discarded from memory as required. For example, during a search operation or when scrolling. As a result, the viewfile window opens instantly, and exhibits better performance and reduced memory usage. However, unlike *LoadAll* mode, the file must remain on disk when using *Sparse* mode. If the file is removed, a warning message is displayed and the viewfile window is closed when attempting to page more data.

Because *Sparse* mode retains minimal data in memory, only a limited amount of data can be copied to the clipboard. Without this limit, the *Select All—Copy* operation would page the entire file contents in order to populate the copy buffer, resulting in an unexpected delay for large files. Therefore, the *Copy* operation is automatically disabled as soon as the data selection exceeds the default size limit of 100000 characters. However, you can modify the limit by using the maximumCopySize environment variable in the `.cdsenv` file.

## Hypertext Viewfile Window

Hypertext Viewfile windows are no longer supported. Instead, you should use HTML Viewfile windows.

A hypertext viewfile window is similar to a text viewfile window except that it can display text with hyperlinks in it. It has a widget type of *hypertext*.

In the text file to be displayed in a hypertext window, specify hyperlinks in the same format as an HTML anchor tag. You can specify three attributes in the anchor tag:

|        |                                                                                                                                                                                                              |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HREF   | Specifies the URL                                                                                                                                                                                            |
| NAME   | Specifies a name that identifies the link; you can jump to this from another link in the document or with the <code>hiScrollWindowToName()</code> function                                                   |
| TARGET | Specifies the window in which to load the page; for example,<br>TARGET= " <u>_blank</u> " will load the page in a new browser window while<br>TARGET= " <u>_self</u> " will load the page in the same window |

For example:

```
" Cadence Design Systems"
" Cadence Design Systems"
" Cadence Design Systems TARGET= \"_blank\" "
" Jump to the Cadence link "
```

All the standard formats for the `HREF` attribute are supported. For example:

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

```
HREF=http://www.myWebsite.com
HREF=mailto://xyz@mycompany.com
HREF="#linkName"
HREF=file://localfile
```

where *localFile* is the name of the local file to load. If *localFile* is a plain text file, it is loaded into the hypertext field when the link is clicked. If *localFile* is in PDF, GIF, or JPEG format, it is sent to the Web browser which either displays the file or prompts the user to save it, based on the browser settings. If *localFile* is in any other format, the browser asks the user whether to open or save the file. In general, however, the hypertext window is intended for plain text, it is not designed to handle all kinds of file formats.)

Web pages will be displayed in the browser specified in the *Web Browser* field in the User Preferences form or the corresponding `ui webBrowser` environment variable in your `.cdsenv` file. The default browser is Netscape.

You can use the following Opus resources to control the hyperlink colors in a hypertext viewfile window:

|                                          |                                                    |
|------------------------------------------|----------------------------------------------------|
| <code>Opus.hypertextColor</code>         | Text color of unvisited hyperlink                  |
| <code>Opus.hypertextSelectedColor</code> | Text color of visited hyperlink                    |
| <code>Opus.hypertextFocusedColor</code>  | Text color of hyperlink when the cursor is over it |

You can use `hiScrollWindowToName` to scroll a hypertext window to a link.

All the functions in this section also apply to hypertext viewfile windows, except for the following selection functions:

```
hiGetTextSelection, hiGetTextSelByLoc, hiSetTextSelection,
hiSelectTextByLoc, hiSetTextSelectAll, hiUnselectText,
hiUnselectTextByLoc, hiUnselectTextClass, hiUnselectTextAll,
hiGetTextClass, hiSetTextClass.
```

### ***Related Topics***

[hiCreateHypertextField](#)

[hiScrollWindowToName](#)

## hiClearClipboard

```
hiClearClipboard(
 [s_mode]
)
=> t / nil
```

### Description

Clears the text stored in the system clipboard.

### Argument

*s\_mode*

The clipboard buffer to inspect. Possible values are:

- 'selection: the UNIX middle-mouse-paste selection.
- 'clipboard: copy buffer, typically used when performing the Copy operation in a GUI.

Default value is 'selection.

### Values Returned

t

The clipboard is cleared.

nil

An invalid argument is specified.

### Examples

```
hiGetClipboard()
=> ""

hiSetClipboard("text to copy")
=> t

hiGetClipboard()
=> "text to copy"

hiGetClipboard('clipboard)
=> "" ; defaults to 'selection, so 'clipboard is not set

hiSetClipboard("some stuff for the clipboard" 'clipboard)
=> t

hiGetClipboard()
=> "text to copy"

hiGetClipboard('clipboard)
=> "some stuff for the clipboard"
```



## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

```
hiGetClipboard('selection)
=> "text to copy"
hiClearClipboard()
=> t
hiGetClipboard()
=> ""
```

#### ***Related Topics***

[Functions of Viewfile Window](#)

## hiClearCurrentTextSelection

```
hiClearCurrentTextSelection(
 w_windowId
)
=> t / nil
```

### Description

Clears current selection, if any, in the specified text window. This is equivalent to clicking anywhere in the text window when text is selected.

### Argument

|                   |                        |
|-------------------|------------------------|
| <i>w_windowId</i> | ID of the text window. |
|-------------------|------------------------|

### Values Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The existing selection was cleared.                   |
| <i>nil</i> | No text was selected, therefore, nothing was cleared. |

### Examples

The `hiSetCurrentTextSelectionLoc` function selects text with indexes 10 : 20, which is equivalent to selecting the text in that range using the mouse. The `hiClearCurrentTextSelection` function clears the selection.

```
win=(view "memoirs.txt")
(hiSetCurrentTextSelectionLoc win 10 20)
selection=(hiGetClipboard)
> "confidentia"
(hiClearCurrentTextSelection win)
```

### ***Related Topics***

[Functions of Viewfile Window](#)

[hiSetCurrentTextSelectionLoc](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiDisableTailViewfile

```
hiDisableTailViewfile(
 w_windowId
)
=> t / nil
```

#### Description

Turns off the ability to repeatedly read from the end of a file.

#### Argument

|                   |                                         |
|-------------------|-----------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on. |
|-------------------|-----------------------------------------|

#### Values Returned

|     |                                                       |
|-----|-------------------------------------------------------|
| t   | The capability is turned off.                         |
| nil | <i>w_windowId</i> is invalid. A warning is displayed. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiEnableTailViewfile

```
hiEnableTailViewfile(
 w_windowId
)
=> t / nil
```

#### Description

Turns on the ability to repeatedly read from the end of a file and update the viewfile window. This continually updates the viewfile window with text that is appended to the end of the file being viewed. This only works when appending to a file. If the file is modified in another way, such as edited using an editor, the results are unpredictable. This function is equivalent to the UNIX command `tail_f` and works the same way.

#### Argument

|                         |                                         |
|-------------------------|-----------------------------------------|
| <code>w_windowId</code> | Window you want the function to act on. |
|-------------------------|-----------------------------------------|

#### Value Returned

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>t</code>   | The capability is turned on.                                |
| <code>nil</code> | <code>w_windowId</code> is invalid. A warning is displayed. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiGenTextIndex

```
hiGenTextIndex(
 w_windowId
 t_text
 x_fromIndex
)
=> x_index / nil
```

#### Description

Returns the character position that is the next occurrence of the text string relative to a specified position in the file.

#### Arguments

|                    |                                              |
|--------------------|----------------------------------------------|
| <i>w_windowId</i>  | Window you want the function to act on.      |
| <i>t_text</i>      | Text string to search for.                   |
| <i>x_fromIndex</i> | Character position to start the search from. |

#### Value Returned

|                |                                                                                                                      |
|----------------|----------------------------------------------------------------------------------------------------------------------|
| <i>x_index</i> | The character position of the matching text.                                                                         |
| <i>nil</i>     | <i>w_windowId</i> is invalid, <i>t_text</i> is not found or null, or no more occurrences of <i>t_text</i> are found. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiViewfile

```
hiViewfile(
)
 => l_filePath / nil
```

#### Description

Brings up the Choose a File to View form in which you can specify a file to view and then calls the `view` function on the file.

#### Arguments

None

#### Value Returned

*l\_filePath*

Returns *l\_filePath* after the form is closed if you clicked *Open* on the form and successfully opened a file.

*l\_filePath* is a list of two strings:

```
list("directoryPath" "fileName")
```

The first element of the list is the path to the file that has been opened and the second element is the name of the file.

*nil*

The form is closed after clicking *Cancel* on the form.

#### Related Topics

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiViewTextFile

```
hiViewTextFile(
 t_file
 ?winSpec g_winspec
 ?title g_title
 ?appName g_appName
 ?help g_help
 ?iconPosition l_iconPosition
 ?deviceMode g_deviceMode
 ?pluginFactory g_pluginFactory
 ?fixedFont g_fixedFont
)
=> w_windowId / nil
```

#### Description

Creates a viewfile window, inserts the menu bar, sets the window and icon names, specifies how to load the file in the window, and displays the file in the window.

This function is similar to the `view` function. However, it uses keywords other than mandatory options to specify the arguments.

#### Arguments

|                           |                                                                                                                                                                                                                                                                |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_file</i>             | File name to view.                                                                                                                                                                                                                                             |
| ?winSpec <i>g_winspec</i> | Screen coordinates of the lower-left and upper-right corners of the window, assuming 0:0 to be the lower-left corner of the window.                                                                                                                            |
| ?title <i>g_title</i>     | Title for the window. This title appears in the window manager banner of the viewfile window. If not specified, the file name is used.                                                                                                                         |
| ?appName <i>g_appName</i> | String representing the window application type or name.<br><br>The default is "Show File".                                                                                                                                                                    |
| ?help <i>g_help</i>       | Help string for the window used to reference a .tgf help file. If you do not specify this argument or specify an empty string, "", the help argument is set to <i>appName</i> , which is the value of the <i>g_appName</i> argument.<br><br>The default is "". |

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

`?iconPosition l_iconPosition`

Icon location (in X Window System screen coordinates) for the window. This causes the window to be iconified initially. If `l_iconPosition` is `nil` or not specified, the window is not iconified initially.

This argument is passed to `hiCreateWindow`.

`?deviceMode g_deviceMode`

Method to be used to load the text file.

Valid values: `LoadAll`, `Sparse`.

`?pluginFactory g_pluginFactory`

This argument is passed to `hiSetViewfile`.

`?fixedFont g_fixedFont`

Allows only fixed-width fonts for display. If the current application uses or changes to a non-fixed width font for text display, it either uses the current fixed-width font or "Noto Mono" as the default font.

The default is `nil`.

### Value Returned

`w_windowId`

ID of the viewfile window.

`nil`

The viewfile window could not be created.

### Examples

Creates a viewfile window to open the text file `log.txt` with fixed width fonts.

```
hiViewTextFile("log.txt" ?fixedFont t)
```

### Related Topics

[Functions of Viewfile Window](#)

[hiViewfile](#)

[hiSetViewfile](#)



## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiSetViewfile

```
hiSetViewfile(
 w_windowId
 t_fileName
 [g_deviceMode]
)
=> t / nil
```

#### Description

Reads a file and displays its contents in a viewfile window that has already been created.

#### Arguments

|                     |                                                                                                                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>   | Window in which you want to display the file                                                                                                                                                   |
| <i>t_fileName</i>   | Name of the file to be read and displayed                                                                                                                                                      |
| <i>g_deviceMode</i> | Specifies the method that is to be used to load the text file. You can choose from the following methods: LoadAll or Sparse. For more information, refer to <a href="#">Text Device Mode</a> . |

#### Value Returned

|            |                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The file is read and displayed.<br><br><i>hiSetViewFile</i> returns <i>t</i> if <i>t_fileName</i> is null/not found. |
| <i>nil</i> | <i>w_windowId</i> is invalid.                                                                                        |

#### Related Topics

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

## hiSaveViewfile

```
hiSaveViewfile(
 [w_windowId]
 [t_fileName]
 [g_dontOverwrite]
)
=> t / nil
```

### Description

Saves the contents of the viewfile window to a file on disk. If you do not specify any arguments, a Save As form appears that lets you select the file.

### Arguments

|                        |                                                                                                                                 |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>      | Window you want the function to act on. The default is the current window.                                                      |
| <i>t_fileName</i>      | Name of the saved file.                                                                                                         |
| <i>g_dontOverwrite</i> | Restricts the file from getting saved if the file name already exists. If set to <code>nil</code> , it will overwrite the file. |

### Value Returned

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <i>t</i>         | The file is saved.                                                                          |
| <code>nil</code> | <i>w_windowId</i> is invalid, <i>t_fileName</i> is not found, or <i>t_fileName</i> is null. |

### Related Topics

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiSaveAsViewfile

```
hiSaveAsViewfile(
 [w_windowId]
 [t_fileName]
)
=> t / nil
```

#### Description

Saves the contents of the current window to the specified name.

This function does not save the file if the file name already exists. You must specify a unique file name before the file is saved.

If you do not specify any arguments, a Save As form appears that lets you select the file.

#### Arguments

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on. The default is the current window. |
| <i>t_fileName</i> | Unique name for the saved file.                                            |

#### Value Returned

|            |                                                                                               |
|------------|-----------------------------------------------------------------------------------------------|
| <i>t</i>   | The file is saved.                                                                            |
| <i>nil</i> | <i>w_windowId</i> is invalid, <i>t_fileName</i> already exists, or <i>t_fileName</i> is null. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiStartGenTextIndex

```
hiStartGenTextIndex(
 w_windowId
 t_text
)
=> x_index / nil
```

#### Description

Searches the specified text string from the beginning of the file and returns the first character position of the matching text.

#### Arguments

|                   |                                         |
|-------------------|-----------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on. |
| <i>t_text</i>     | Text string to search for.              |

#### Value Returned

|                |                                                                                     |
|----------------|-------------------------------------------------------------------------------------|
| <i>x_index</i> | The first character position of the matching text if the text is found.             |
| <i>nil</i>     | <i>w_windowId</i> is invalid, <i>t_text</i> is not found, or <i>t_text</i> is null. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## hiGetTextSelection

```
hiGetTextSelection(
 w_windowId
 [x_class]
)
=> l_text / nil
```

### Description

Retrieves the currently selected text strings from within a specified class.

### Arguments

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_windowId</code> | Text window you want the function to act on.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>x_class</code>    | <p>The group class that the selections should belong to. Possible values range from 0 to <code>MAX_INT</code>. The default is 0.</p> <p>The class 0 corresponds to the <i>Edit – Find</i> command and returns the coordinates of the text found using this command. If you select the <i>Scan the whole file</i> check box in the Search dialog box, it returns a list of multiple coordinates based on the text highlighted using the search operation.</p> |

### Value Returned

|                     |                                                                                              |
|---------------------|----------------------------------------------------------------------------------------------|
| <code>l_text</code> | A list of all the selected text strings of the class specified.                              |
| <code>nil</code>    | <code>w_windowId</code> is invalid or no text is currently selected for the specified class. |

### Related Topics

[Functions of Viewfile Window](#)

[hiGetCurrentTextSelection](#)

[hiGetCurrentTextSelectionLoc](#)

[hiSetCurrentTextSelectionLoc](#)

## hiGetTextSelByLoc

```
hiGetTextSelByLoc (
 w_windowId
 [x_class]
)
=> l_locationPair / nil
```

### Description

Returns a list of starting and ending location pairs of all selected text strings in the specified class.

### Arguments

|                         |                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_windowId</code> | Text window you want the function to act on.                                                                                                                                                                                                                                                                                                                                                   |
| <code>x_class</code>    | Group of selections. Values range from 0 to MAX_INT. The default is 0.<br><br>The class 0 corresponds to the <i>Edit – Find</i> command and returns the coordinates of the text found using this command. If you select the <i>Scan the whole file</i> check box in the Search dialog box, it returns a list of multiple coordinates based on the text highlighted using the search operation. |

### Value Returned

|                             |                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------|
| <code>l_locationPair</code> | A list of the beginning and ending locations of all selected text strings of the class specified. |
| <code>nil</code>            | <code>w_windowId</code> is invalid or no text is currently selected.                              |

### Related Topics

[Functions of Viewfile Window](#)

[hiGetCurrentTextSelection](#)

[hiGetCurrentTextSelectionLoc](#)

[hiSetCurrentTextSelectionLoc](#)

## hiSetTextSelection

```
hiSetTextSelection(
 w_windowId
 t_text
 [x_class]
)
=> t / nil
```

### Description

Highlights the next occurrence of a specified string relative to the current cursor position. If no matching string is found between the current cursor position and the end of the file, it wraps around to the beginning of the file and continues the search.

### Arguments

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on.                                |
| <i>t_text</i>     | Specifies the text string that needs to be searched.                   |
| <i>x_class</i>    | Group of selections. Values range from 0 to MAX_INT. The default is 0. |

### Value Returned

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| <i>t</i>   | A match is found.                                                                   |
| <i>nil</i> | <i>w_windowId</i> is invalid, <i>t_text</i> is not found, or <i>t_text</i> is null. |

### ***Related Topics***

[Functions of Viewfile Window](#)

## hiSelectTextByLoc

```
hiSelectTextByLoc(
 w_windowId
 l_locationPair
 [x_class]
)
=> t / nil
```

### Description

Highlights a list of strings specified as a list of starting and ending location pairs.

### Arguments

|                       |                                                                        |
|-----------------------|------------------------------------------------------------------------|
| <i>w_windowId</i>     | Window you want the function to act on.                                |
| <i>l_locationPair</i> | List of starting and ending location pairs for the selection.          |
| <i>x_class</i>        | Group of selections. Values range from 0 to MAX_INT. The default is 0. |

### Value Returned

|            |                                                                        |
|------------|------------------------------------------------------------------------|
| <i>t</i>   | A match is found.                                                      |
| <i>nil</i> | <i>w_windowId</i> is invalid or <i>l_locationPair</i> is out of range. |

### Examples

Adds the selected text to text class 6, so it will persist when deselected, and will be highlighted in the style assigned to text class 6

```
hiSelectTextByLoc(window(3) list(1597:1602) 6)
```

### Related Topics

[Functions of Viewfile Window](#)



## hiGetCurrentTextSelection

```
hiGetCurrentTextSelection(
 w_windowId
)
=>t_selectedText / nil
```

### Description

Returns the text selected in the specified viewfile window. This is the text which has been selected by using a mouse button action, such as by double-clicking or drag-selecting. If no text appears selected, the function returns `nil`.

### Argument

|                   |                                                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Specifies the ID of the viewfile window for which the current text selection should be returned |
|-------------------|-------------------------------------------------------------------------------------------------|

### Values Returned

|                       |                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------|
| <i>t_selectedText</i> | Returns a string if text is selected in the specified viewfile window.                       |
| <code>nil</code>      | Returns <code>nil</code> if no text appears to be selected in the specified viewfile window. |

### Examples

```
hiGetCurrentTextSelection(window(3))
=> "some text" ;; The string "some text" has been selected in window(3)
```

```
hiGetCurrentTextSelection(window(3))
=> nil ;; No text is selected
```

### ***Related Topics***

#### Functions of Viewfile Window

## hiGetCurrentTextSelectionLoc

```
hiGetCurrentTextSelectionLoc(
 w_windowId
)
=> l_listofCoords / nil
```

### Description

Returns the text coordinates of the text selected in the specified viewfile window. This is the text which has been selected using a mouse action, such as by double-clicking or drag-selecting. If no text is selected in the specified window, the function returns `nil`.

### Argument

|                         |                                                                                                                 |
|-------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>w_windowId</code> | Specifies the ID of the viewfile window for which the location of the current text selection should be returned |
|-------------------------|-----------------------------------------------------------------------------------------------------------------|

### Values Returned

|                             |                                                                                             |
|-----------------------------|---------------------------------------------------------------------------------------------|
| <code>l_listofCoords</code> | Returns a list of two numbers, comprising of the first and last selected character indexes  |
| <code>nil</code>            | Returns <code>nil</code> if no text appears to be selected in the specified viewfile window |

### Examples

```
hiGetCurrentTextSelectionLoc(window(3))
=> (1597 1602) ;; Text between 1597 and 1602 is selected in window(3)
```

```
hiGetCurrentTextSelection(window(3))
=> nil ;; No text is selected
```

### Related Topics

#### Functions of Viewfile Window

## hiSetCurrentTextSelectionLoc

```
hiSetCurrentTextSelectionLoc(
 w_windowId
 x_startIndex
 x_endIndex
)
=> t | nil
```

### Description

Selects the range of text specified by the start and end indexes in the given text window. This is equivalent to selecting the text with the mouse.

### Arguments

|                     |                                                                   |
|---------------------|-------------------------------------------------------------------|
| <i>w_windowId</i>   | ID of the text window.                                            |
| <i>x_startIndex</i> | The index of the first character to be selected, starting from 0. |
| <i>x_endIndex</i>   | The index of the last character to be selected.                   |

### Values Returned

|            |                                       |
|------------|---------------------------------------|
| <i>t</i>   | The requested range was selected.     |
| <i>nil</i> | The requested range was not selected. |

### Examples

The `hiSetCurrentTextSelectionLoc` function selects the first 7 characters. The `hiGetClipboard` function returns the selected text, `Private`, which is stored in the clipboard.

```
win=(view "memoirs.txt")
(hiSetCurrentTextSelectionLoc win 0:6)
(hiGetClipboard)
> "Private"
```

### ***Related Topics***

#### Functions of Viewfile Window

#### hiGetClipboard

## hiSetTextSelectAll

```
hiSetTextSelectAll(
 w_windowId
 t_text
 [x_class]
)
=> t / nil
```

### Description

Highlights all occurrences of a specified string and resets the cursor to the top of the file.

### Arguments

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on.                                      |
| <i>t_text</i>     | Text string to search for.                                                   |
| <i>x_class</i>    | Group of selections. Values range from 0 to MAX_INT. The default value is 0. |

### Value Returned

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| <i>t</i>   | A match is found.                                                                   |
| <i>nil</i> | <i>w_windowId</i> is invalid, <i>t_text</i> is not found, or <i>t_text</i> is null. |

### Related Topics

[Functions of Viewfile Window](#)

## hiUnselectText

```
hiUnselectText(
 w_windowId
 t_text
 [x_class]
)
=> t / nil
```

### Description

Unhighlights the next occurrence of selected text in the viewfile relative to the cursor position.

### Arguments

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on.                                      |
| <i>t_text</i>     | Text string to be unselected.                                                |
| <i>x_class</i>    | Group of selections. Values range from 0 to MAX_INT. The default value is 0. |

### Value Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | A match is found.                                     |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

### ***Related Topics***

[Functions of Viewfile Window](#)

## hiUnselectTextByLoc

```
hiUnselectTextByLoc(
 w_windowId
 l_locationPair
 [x_class]
)
=> t / nil
```

### Description

Unhighlights selected text specified as a list of starting and ending location pairs in the given window.

### Arguments

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <i>w_windowId</i>     | Window you want the function to act on.                                      |
| <i>l_locationPair</i> | List of starting and ending location pairs.                                  |
| <i>x_class</i>        | Group of selections. Values range from 0 to MAX_INT. The default value is 0. |

### Value Returned

|            |                                                         |
|------------|---------------------------------------------------------|
| <i>t</i>   | The text is unselected.                                 |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is also issued. |

### Related Topics

[Functions of Viewfile Window](#)

## hiUnselectTextClass

```
hiUnselectTextClass(
 w_windowId
 x_class
)
=> t / nil
```

### Description

Unhighlights all occurrences of selected text belonging to the specified class.

### Arguments

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on.                                      |
| <i>x_class</i>    | Group of selections. Values range from 0 to MAX_INT. The default value is 0. |

### Value Returned

|            |                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------|
| <i>t</i>   | The text is no longer highlighted.                                                                 |
| <i>nil</i> | <i>w_windowId</i> is invalid or no selected text matches with the class. A warning is also issued. |

### ***Related Topics***

[Functions of Viewfile Window](#)

## hiUnselectTextAll

```
hiUnselectTextAll
 w_windowId
 [t_text]
 [x_class]
)
=> t / nil
```

### Description

Unselects all occurrences of the selected text that match with the specified pattern *t\_text* and class *x\_class* in the viewfile. If the search pattern *t\_text* is not specified, unselects all occurrences of the selected text of all classes. If the class *x\_class* is not specified, unselects all occurrences of the selected text that match with the specified pattern *t\_text*. Afterwards, the cursor position is reset to the top of the file. If an empty string is passed for *t\_text*, and *x\_class* is specified, unselects all strings selected in that class

### Arguments

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <i>w_windowId</i> | Window you want the function to act on.                                      |
| <i>t_text</i>     | Text to be unselected.                                                       |
| <i>x_class</i>    | Group of selections. Values range from 0 to MAX_INT. The default value is 0. |

### Value Returned

|            |                                                                                                                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The text is unselected.                                                                                                                     |
| <i>nil</i> | <i>w_windowId</i> is invalid, no selected text matches with the specified text and class, or nothing is selected. A warning is also issued. |

### Related Topics

[Functions of Viewfile Window](#)



## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

## hiGoToLine

```
hiGoToLine (
 w_windowId
 x_lineNo
 ?highlight g_highlight
 ?align g_align
 ?forceAlign g_forceAlign
)
=> t / nil
```

## Description

Positions the specified line number onscreen for a given text window

## Arguments

|                                 |                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>               | Specifies the ID of the text editor window                                                                                                                                                                                                                                                                                                                                       |
| <i>x_lineNo</i>                 | Specifies the desired line number that you want to highlight.<br>The line numbers start at 1.                                                                                                                                                                                                                                                                                    |
| <i>?highlight g_highlight</i>   | Highlights the line with a green background to help you locate the desired line number.<br><br>Like any selection, the highlight will disappear when the left mouse is clicked in the text window.<br><br>The default value is <i>t</i> .                                                                                                                                        |
| <i>?align g_align</i>           | Indicates where the highlighted line should be positioned with respect to the window of the text editor, that is at the top, in the middle, or at the bottom of the window.<br><br>If the line is already in the visible area of the window, the view-port does not adjusted unless <i>?forceAlign</i> is set to <i>t</i> .<br><br>Valid values are 'top', 'center' or 'bottom'. |
| <i>?forceAlign g_forceAlign</i> |                                                                                                                                                                                                                                                                                                                                                                                  |

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

Adjusts the visible region to respect the value for `?align`, even if the requested line was already visible.

The `?forceAlign` parameter is most likely to be useful while working on code, especially, where you want to alter what is displayed in the viewport. For example, positioning the required line at the top of a verilog block, rather than just ensuring that the line is somewhere on the page.

### Values Returned

|                  |                                            |
|------------------|--------------------------------------------|
| <code>t</code>   | The line number is valid and can be shown. |
| <code>nil</code> | The line number cannot be shown.           |

### Examples

```
win=view("myFile.txt");; Opens a text file
```

```
hiGoToLine(win 100 ?highlight t ?align 'center ?forceAlign t) ;; Highlights line 100 in the center of the window
```

```
hiGoToLine(win 110 ?highlight t ?align 'center ?forceAlign nil);; Highlights line 110, but if it's already visible, don't force it to be centered
```

### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiGoToIndex

```
hiGoToIndex(
 w_windowId
 x_indexNo
 ?highlight g_highlight
 ?align g_align
 ?forceAlign g_forceAlign
)
=> t / nil
```

#### Description

Positions the specified character index within the viewport for a given text window

#### Arguments

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_windowId</code>               | Specifies the ID of the text window                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>x_indexNo</code>                | Specifies the desired index of the character that you want to highlight. The index number starts at 0.                                                                                                                                                                                                                                                                                                                  |
| <code>?highlight g_highlight</code>   | <p>Highlights the index with a green background to help you locate the desired index number.</p> <p>In any selection, the highlight gets disappear when the left mouse is clicked in the text window.</p> <p>The default value is <code>t</code>.</p>                                                                                                                                                                   |
| <code>?align g_align</code>           | <p>Indicates where the highlighted text / character should be positioned with respect to the window of the text editor, that is at the top, in the middle, or at the bottom of the window.</p> <p>If the desired index already in the visible area of the window, the viewport does not get adjusted unless <code>?forceAlign</code> is set to <code>t</code>.</p> <p>Valid values are 'top', 'center' or 'bottom'.</p> |
| <code>?forceAlign g_forceAlign</code> |                                                                                                                                                                                                                                                                                                                                                                                                                         |

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

Adjusts the visible region to respect the value for `?align`, even if the requested index was already visible.

The `?forceAlign` parameter is most likely to be useful while working on code, especially, where you want to alter what is displayed in the viewport. For example, positioning the required line at the top of a verilog block, rather than just ensuring that the line is somewhere on the page.

### Values Returned

|                  |                                              |
|------------------|----------------------------------------------|
| <code>t</code>   | The line number is valid and gets displayed. |
| <code>nil</code> | The line number is invalid.                  |

### Examples

```
win=(view "myFile.txt");; Opens a text window
```

```
hiGoToIndex(win 0 ?highlight t);; Highlights first character
```

```
hiGoToIndex(win 1000 ?highlight t ?align 'center ?forceAlign t);; Highlights
character 1000, and position it centrally in text window
```

### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowLeft

```
hiScrollWindowLeft(
 w_windowId
)
=> t / nil
```

#### Description

Scrolls the window left one screenful of text at a time.

#### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

#### Value Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                               |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowRight

```
hiScrollWindowRight(
 w_windowId
)
=> t / nil
```

#### Description

Scrolls the window right one screenful of text at a time.

#### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

#### Values Returned

|            |                                                         |
|------------|---------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                                 |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is also issued. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowUp

```
hiScrollWindowUp(
 w_windowId
)
=> t / nil
```

#### Description

Scrolls the window up one screenful of text at a time.

#### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

#### Values Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                               |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowDown

```
hiScrollWindowDown(
 w_windowId
)
=> t / nil
```

#### Description

Scrolls the window down one screenful of text at a time.

#### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

#### Values Returned

|            |                                                         |
|------------|---------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                                 |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is also issued. |

#### ***Related Topics***

[Functions of Viewfile Window](#)



## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowTop

```
hiScrollWindowTop(
 w_windowId
)
=> t / nil
```

#### Description

Scrolls the window to the top position of the viewable window.

#### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

#### Value Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                               |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowBottom

```
hiScrollWindowBottom(
 w_windowId
)
=> t / nil
```

#### Description

Scrolls the window to the bottom position of the viewable window.

#### Argument

|                   |                        |
|-------------------|------------------------|
| <i>w_windowId</i> | Specify the window ID. |
|-------------------|------------------------|

#### Values Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                               |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

## hiScrollWindowToCurrentIndex

```
hiScrollWindowToCurrentIndex(
 w_windowId
)
=> t / nil
```

### Description

Scrolls the window to the current cursor index position in the viewable window.

### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

### Values Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The window is scrolled.                               |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

### ***Related Topics***

[Functions of Viewfile Window](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowToIndex

```
hiScrollWindowToIndex(
 w_windowId
 x_index
)
=> t / nil
```

#### Description

Scrolls the window to the index position in the specified window and forces the text at the given position to be displayed.

#### Arguments

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>w_windowId</i> | Specifies the window ID.                          |
| <i>x_index</i>    | Zero-relative character position to be displayed. |

#### Value Returned

|            |                                                                         |
|------------|-------------------------------------------------------------------------|
| <i>t</i>   | The text is displayed.                                                  |
| <i>nil</i> | <i>w_windowId</i> or <i>x_index</i> is invalid. A warning is displayed. |

#### ***Related Topics***

Functions of Viewfile Window

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiScrollWindowToName

```
hiScrollWindowToName(
 w_hypertextWindow
 t_linkName
)
=> t / nil
```

#### Description

Scrolls a hypertext window to a link and displays the text associated with the link. A hypertext window is a text window that can display text with hyperlinks in it.

#### Arguments

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_hypertextWindow</i> | The hypertext window that you want to scroll.<br><br>You can create a hypertext window with <code>hiCreateWindow</code> or <code>hiOpenWindow</code> by specifying "hypertext" for the <i>t_widgetType</i> argument.                                                                                                                                                                                                                                                      |
| <i>t_linkName</i>        | The link to which you want to scroll, specified as a string. For example: "myLink".<br><br>The link name is defined in the text file that you view in the hypertext window. It is defined with the <code>NAME</code> attribute of the HTML anchor tag. For example:<br><br>"<A name=myLink href=http://www.cadence.com><br>Cadence Design Systems</A>"<br><br>For more information on how to specify hyperlinks in a text file, see <a href="#">Viewfile Management</a> . |

#### Values Returned

|            |                                                           |
|------------|-----------------------------------------------------------|
| <i>t</i>   | The window is scrolled to the link.                       |
| <i>nil</i> | <i>w_hypertextWindow</i> or <i>t_linkName</i> is invalid. |

#### Related Topics

[Functions of Viewfile Window](#)

## **Cadence User Interface SKILL Reference**

### Functions of Viewfile Window

---

hiOpenWindow

hiSetViewfile

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiGetCurrentIndex

```
hiGetCurrentIndex(
 w_windowId
)
=> x_index / nil
```

#### Description

Returns the current cursor index position relative to all the text in the window.

The related `hiSetCurrentIndex` SKILL command will respect document sizes, and also output a warning if the current index is set beyond the document limits.

#### Argument

|                         |                          |
|-------------------------|--------------------------|
| <code>w_windowId</code> | Specifies the window ID. |
|-------------------------|--------------------------|

#### Values Returned

|                      |                                                               |
|----------------------|---------------------------------------------------------------|
| <code>x_index</code> | The current cursor position.                                  |
| <code>nil</code>     | <code>w_windowId</code> is invalid. A warning is also issued. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

[hiSetCurrentIndex](#)

## hiSetCurrentIndex

```
hiSetCurrentIndex(
 w_windowId
 x_index
)
=> t / nil
```

### Description

Sets the current cursor index position relative to all the text in a window and scrolls the window, if needed.

The current index position is modified by `hiTextDisplayString`.

The `hiSetCurrentIndex` SKILL command will respect document sizes, and displays a warning if the current index is set beyond the document limits.

### Arguments

|                         |                                                               |
|-------------------------|---------------------------------------------------------------|
| <code>w_windowId</code> | Specifies the window ID.                                      |
| <code>x_index</code>    | Specifies the zero-relative character position of the cursor. |

### Values Returned

|                  |                                                                                     |
|------------------|-------------------------------------------------------------------------------------|
| <code>t</code>   | The cursor position is set.                                                         |
| <code>nil</code> | <code>w_windowId</code> or <code>x_index</code> is invalid. A warning is displayed. |

### Related Topics

[Functions of Viewfile Window](#)

[hiSetCurrentIndex](#)

[hiTextDisplayString](#)



## hiGetClipboard

```
hiGetClipboard(
 s_mode
)
=> t_value / nil
```

### Description

Returns the text stored in the system clipboard.

### Argument

*s\_mode*

The clipboard buffer to inspect. Possible values are:

- 'selection: The UNIX middle-mouse-paste selection.
- 'clipboard: Copies buffer and used during the copy operation in a GUI.

Default value is 'selection.

### Value Returned

*t\_value*

The text is stored in the system clipboard.

nil

An invalid argument is specified.

### Examples

```
hiGetClipboard()
=> ""

hiSetClipboard("text to copy")
=> t

hiGetClipboard()
=> "text to copy"

hiGetClipboard('clipboard)
=> "" ; defaults to 'selection, so 'clipboard is not set

hiSetClipboard("some stuff for the clipboard" 'clipboard)
=> t

hiGetClipboard()
=> "text to copy"

hiGetClipboard('clipboard)
=> "some stuff for the clipboard"
```

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

```
hiGetClipboard('selection)
=> "text to copy"
hiClearClipboard()
=> t
hiGetClipboard()
=> ""
```

### Additional Information

To configure the clipboard/selection behavior:

1. Click *Klipper – Clipboard tool*  icon from the right side of the bottom panel.
2. Choose *Configure Klipper* from the context menu.

In the General tab, you can choose one of the following options:

| Option                                                         | Description                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Synchronize contents of the clipboard and the selection</i> | <p>Synchronizes the content of clipboard and selection buffers.</p> <p>If this option is selected, the return value of <code>hiGetClipboard( 'clipboard )</code> and <code>hiGetClipboard( 'selection )</code> remains the same.</p>                                                                                                          |
| <i>Separate clipboard and selection</i>                        | <p>Keeps the clipboard and selection buffers separate.</p> <p>If this option is selected, the <code>hiGetClipboard( 'clipboard )</code> returns the text saved using the Copy (Ctrl+v) operation. However, the <code>hiGetClipboard( 'selection )</code> returns the selected text.</p> <p>This option is selected by default in KDE 3.x.</p> |

### Related Topics

[Functions of Viewfile Window](#)

## hiSetClipboard

```
hiSetClipboard(
 t_text
 [s_mode]
)
=> t / nil
```

### Description

Sets the text to be pushed to the system clipboard.

### Arguments

|               |                                                                                                                                                                                                                                                                          |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_text</i> | The text that needs to be pushed to the clipboard.                                                                                                                                                                                                                       |
| <i>s_mode</i> | The clipboard buffer to inspect. Possible values are: <ul style="list-style-type: none"><li>■ 'selection: The UNIX middle-mouse-paste selection.</li><li>■ 'clipboard: Copies buffer and used during the copy operation in a GUI.</li></ul> Default value is 'selection. |

### Values Returned

|            |                                             |
|------------|---------------------------------------------|
| <i>t</i>   | The text is pushed to the system clipboard. |
| <i>nil</i> | No text is specified as an argument.        |

### Examples

```
hiGetClipboard()
=> ""

hiSetClipboard("text to copy")
=> t

hiGetClipboard()
=> "text to copy"

hiGetClipboard('clipboard)
=> "" ; defaults to 'selection, so 'clipboard is not set

hiSetClipboard("some stuff for the clipboard" 'clipboard)
=> t

hiGetClipboard()
=> "text to copy"
```

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

```
hiGetClipboard('clipboard)
=> "some stuff for the clipboard"
hiGetClipboard('selection)
=> "text to copy"
hiClearClipboard()
=> t
hiGetClipboard()
=> ""
```

### ***Related Topics***

[Functions of Viewfile Window](#)

## hiGetTextClass

```
hiGetTextClass(
 w_windowId
)
=> x_class / nil
```

### Description

Returns the value of the current active text class. If `hiSetTextClass()` has not been set, the current active text class will be 0. Also, a few other functions may reset the active text class back to 0 or possibly another value. Otherwise the currently active text class will be the one set by the last call to `hiSetTextClass()`.

This SKILL API is included for compatibility reasons and may be removed in a future release.

### Argument

|                         |                          |
|-------------------------|--------------------------|
| <code>w_windowId</code> | Specifies the window ID. |
|-------------------------|--------------------------|

### Values Returned

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>x_class</code> | The current active text class.                              |
| <code>nil</code>     | <code>w_windowId</code> is invalid. A warning is displayed. |

### ***Related Topics***

[Functions of Viewfile Window](#)

[hiSetTextClass](#)

## hiSetTextClass

```
hiSetTextClass(
 w_windowId
 x_class
)
=> t / nil
```

### Description

Sets the active selection (highlight) text class in a window.

This SKILL API is included for compatibility reasons and may be removed in a future release.

### Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Specifies the window ID.                                                                                                                                                                                                                                                                                                                                             |
| <i>x_class</i>    | Group of selections. Values range from 0 to MAX_INT.<br><br>Zero is the default value and uses the window inverse color for highlighting selected text. The highlight selection color for class values of 1 through 25 can be set using the function <code>hiSetTextHighlightColor()</code> . Class values greater than 25 use the same highlight colors as class 0. |

### Values Returned

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>t</i>   | The class is set.                                     |
| <i>nil</i> | <i>w_windowId</i> is invalid. A warning is displayed. |

### ***Related Topics***

Functions of Viewfile Window

hiSetTextHighlightColor

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

## view

```
view(
 t_file
 [g_boxSpec]
 [g_title]
 [g_autoUpdate]
 [l_iconPosition]
 [g_deviceMode]
)
=> w_windowId / nil
```

## Description

Creates a viewfile window, inserts the menu bar, sets the window and icon names, specifies how to load the file in the window, and displays the file in the window. `view` is a wrapper function.

## Arguments

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_file</i>       | File name to view. This is the only required argument for this function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <i>g_boxSpec</i>    | Screen coordinates of the lower left and upper right corners of the window (assuming 0:0 to be the lower left corner of the screen). If <i>g_boxSpec</i> is <code>nil</code> , either you are prompted with a rubberbanding box to enter a window size, or a default size is used, depending on the value of the global variable <code>hivWinStyle</code> . The <code>'default</code> and <code>'interactive</code> symbols are also acceptable values for the <i>g_boxSpec</i> argument. If <i>g_boxSpec</i> is a window ID, the widget type of the window is changed to <code>text</code> , if it isn't already. |
| <i>g_title</i>      | Title for the window. This title appears in the window manager banner of the viewfile window. If not specified, the file name is used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>g_autoUpdate</i> | Determines whether continuous updating of the display from the file is desired. If not specified or <code>nil</code> , the display does not reflect changes made by appending to the end of the file. If <code>t</code> , the display is updated when text is appended to the end of the file. Changes to the file other than appending to the end are not displayed. Setting <i>g_autoUpdate</i> to <code>t</code> is equivalent to calling <code>hiEnableTailViewfile</code> .                                                                                                                                   |

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

|                       |                                                                                                                                                         |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_iconPosition</i> | This argument is passed to <code>hiCreateWindow</code> .                                                                                                |
| <i>g_deviceMode</i>   | Specifies the method that is to be used to load the text file. You can choose from the following methods: <code>LoadAll</code> or <code>Sparse</code> . |

#### Value Returned

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <i>w_windowId</i> | The window ID of the created viewfile window.                              |
| <i>nil</i>        | The viewfile window could not be created. An error message is also issued. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

[hiEnableTailViewfile](#)

[Text Device Mode](#)

[Viewfile Management](#)



## Text Customization Functions

This topic provides a list of Cadence® SKILL functions associated with the highlight color settings. These functions let you set the highlight color, update the text selection colors, refresh the viewfile window, and display text in a particular way.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- hiRefreshTextWindow
- hiSetTextHighlightColor
- hiTextDisplayString
- hiUpdateTextSelectionColorsWord Delimiters

## hiSetTextHighlightColor

```
hiSetTextHighlightColor(
 x_class
 l_foreground
 l_background
)
=> t / nil
```

### Description

Sets the foreground and background highlight colors of a class to the specified RGB value.

The range of the class is from 1 through 25. Only classes 1 through 10 are initialized at startup. Whenever a class number is used from 11 through 25 that has not had its color initialized with this function, the highlight color will be the same as for class 0. Since 25 is the highest class number that can have a unique color combination assigned, any class above 25 will always use the same highlight color as for class 0.

The colors can be specified either as RGB values with ranges from 0 to 1000 or as the return value of *nameToColor*, which translates the predefined color (as contained in the server's rgb.txt file) into its corresponding RGB values. For example, the following command highlights class 5 with white as the foreground color and navy as the background color.

```
hiSetTextHighlightColor(5 nameToColor("white") nameToColor("navy"))
```

Following are the default settings for classes 1 through 10. Class 0 uses the background color as the foreground (text) color and the text (foreground) color as the background color for selections:

| Class  | Foreground | Background |
|--------|------------|------------|
| Number | Color      | Color      |
| 1      | white      | green      |
| 2      | white      | red        |
| 3      | yellow     | navy       |
| 4      | red        | yellow     |
| 5      | white      | blue       |
| 6      | white      | violet     |
| 7      | black      | turquoise  |
| 8      | yellow     | orange     |

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

| Class | Foreground | Background |
|-------|------------|------------|
| 9     | yellow     | magenta    |
| 10    | white      | black      |

---

### Arguments

|                     |                                                                                                                       |
|---------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>x_class</i>      | Class number for selection. Valid range is 1 through 25. Zero is the default value and uses the window inverse color. |
| <i>l_foreground</i> | Color for foreground.                                                                                                 |
| <i>l_background</i> | Color for background.                                                                                                 |

### Values Returned

|     |                                                                                                                                                                                |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| t   | The classes have been set.                                                                                                                                                     |
| nil | <i>x_class</i> is smaller than one or greater than 25, or the RGB value of the foreground and background color is not within the range of 0 to 1000. A warning is also issued. |

### Related Topics

[Functions of Viewfile Window](#)

[Text Customization Functions](#)

## hiTextDisplayString

```
hiTextDisplayString(
 w_windowId
 t_text
 g_erase
 [g_dontScrollTop]
)
=> t / nil
```

### Description

Displays a text string in a window.

The current cursor index and the corresponding cursor position are modified by this function.

### Arguments

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_windowId</i>      | Specifies the window ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>t_text</i>          | Text string to display.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>g_erase</i>         | Determines how the string should be displayed. If <i>g_erase</i> is <i>t</i> , the content of the window is erased and the text string is displayed beginning at the first character position in the window. If <i>g_erase</i> is <i>nil</i> , the text string is displayed following the last character position of the current text.                                                                                                                                   |
| <i>g_dontScrollTop</i> | Determines whether the window should be scrolled. If <i>g_dontScrollTop</i> is not specified or is <i>nil</i> , the window is scrolled back to display the first character position of the window, and the current cursor index position is set to zero. If <i>g_dontScrollTop</i> is <i>t</i> , the window is scrolled to display the last character position of the window, and the current cursor index position is set to the last character position of the window. |

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### Values Returned

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>t</code>   | The string is displayed.                                           |
| <code>nil</code> | <code>w_windowId</code> is invalid or <code>t_text</code> is null. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

[Text Customization Functions](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiRefreshTextWindow

```
hiRefreshTextWindow(
 w_window
)
=> t / nil
```

#### Description

Redraws all the selections and refreshes the viewfile window.

#### Argument

|                 |                          |
|-----------------|--------------------------|
| <i>w_window</i> | Specifies the window ID. |
|-----------------|--------------------------|

#### Values Returned

|     |                             |
|-----|-----------------------------|
| t   | The window is refreshed.    |
| nil | <i>w_window</i> is invalid. |

#### ***Related Topics***

[Functions of Viewfile Window](#)

[Text Customization Functions](#)

## hiUpdateTextSelectionColors

```
hiUpdateTextSelectionColors(
 w_windowId
)
=> t / nil
```

### Description

Redraws all selections visible in the window and updates any colors previously reset with `hiSetTextHighlightColor`.

After resetting the highlight color using `hiSetTextHighlightColor`, this function visually updates the colors of the selections.

### Argument

|                         |                          |
|-------------------------|--------------------------|
| <code>w_windowId</code> | Specifies the window ID. |
|-------------------------|--------------------------|

### Value Returned

|                  |                                     |
|------------------|-------------------------------------|
| <code>t</code>   | The selections are redrawn.         |
| <code>nil</code> | <code>w_windowId</code> is invalid. |

### ***Related Topics***

[Functions of Viewfile Window](#)

[hiSetTextHighlightColor](#)

[Text Customization Functions](#)

## Word Delimiters

This topic provides a list of Cadence® SKILL functions associated with word delimiters customization.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiAddTextWordDelimiter](#)
- [hiGetTextWordDelimiter](#)
- [hiRemoveTextWordDelimiter](#)
- [hiReplaceTextWordDelimiter](#)



## hiGetTextWordDelimiter

```
hiGetTextWordDelimiter(
 w_windowId
)
=> l_delimiters / nil
```

### Description

Returns the current settings of the word delimiter list for the window.

The default word delimiters are defined as any character not alphanumeric.

### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

### Value Returned

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| <i>l_delimiters</i> | A list of characters currently set as word delimiters. |
| <i>nil</i>          | <i>w_windowId</i> is invalid.                          |

### ***Related Topics***

[Word Delimiters](#)

## hiAddTextWordDelimiter

```
hiAddTextWordDelimiter(
 w_windowId
 t_delimiters
)
=> t / nil
```

### Description

Adds characters to the word delimiter list for the window.

### Arguments

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| <i>w_windowId</i>   | Specifies the window ID.                            |
| <i>t_delimiters</i> | Characters to be added to the word delimiter list . |

### Value Returned

|            |                                                                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The characters are added.                                                                                                                                                |
| <i>nil</i> | One of the characters is already defined as a word delimiter, <i>t_delimiters</i> is NULL or an empty string, or <i>w_windowId</i> is invalid. A warning is also issued. |

### ***Related Topics***

[Word Delimiters](#)

## hiRemoveTextWordDelimiter

```
hiRemoveTextWordDelimiter(
 w_windowId
 t_delimiters
)
=> t / nil
```

### Description

Removes characters from the word delimiter list for the window.

### Arguments

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| <i>w_windowId</i>   | Specifies the window ID.                               |
| <i>t_delimiters</i> | Characters to be removed from the word delimiter list. |

### Value Returned

|            |                                                                                                                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The characters are removed.                                                                                                                                                                           |
| <i>nil</i> | The character in <i>t_delimiters</i> is not found in the word delimiter list, <i>t_delimiters</i> is <code>NULL</code> or an empty string, or <i>w_windowId</i> is invalid. A warning is also issued. |

### ***Related Topics***

[Word Delimiters](#)

## hiReplaceTextWordDelimiter

```
hiReplaceTextWordDelimiter(
 w_windowId
 t_delimiters
)
=> t / nil
```

### Description

Replaces the original word delimiters for the window with a new set of characters.

### Arguments

|                     |                                                      |
|---------------------|------------------------------------------------------|
| <i>w_windowId</i>   | Specifies the window ID.                             |
| <i>t_delimiters</i> | New set of characters to be used as word delimiters. |

### Values Returned

|            |                                                                                                            |
|------------|------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The set is replaced.                                                                                       |
| <i>nil</i> | <i>t_delimiters</i> is NULL or an empty string, or <i>w_windowId</i> is invalid. A warning is also issued. |

### ***Related Topics***

[Word Delimiters](#)

## Miscellaneous Viewfile Functions

The functions described in this section let you retrieve text at a specific location, get text source length, retrieve a line and column number, a text index, or a display location, and disable the text selection default.

This topic provides a list of Cadence® SKILL functions associated with miscellaneous actions.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiGetTextCharAtLoc](#)
- [hiGetTextSourceLength](#)
- [hiGetTextLineColumn](#)
- [hiGetTextIndexLoc](#)
- [hiGetTextDispLoc](#)

## hiGetTextCharAtLoc

```
hiGetTextCharAtLoc (
 w_windowId
 x_location
)
=> c_text / nil
```

### Description

Returns the character of the specified character index location for a window.

### Arguments

|                   |                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------|
| <i>w_windowId</i> | Specifies the window ID.                                                                      |
| <i>x_location</i> | Character index location. If not specified, the current cursor index location is the default. |

### Value Returned

|               |                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------|
| <i>c_text</i> | The character at the specified location.                                                             |
| <i>nil</i>    | <i>w_windowId</i> is invalid or the specified location is out of range.<br>A warning is also issued. |

### ***Related Topics***

[Miscellaneous Viewfile Functions](#)

## Cadence User Interface SKILL Reference

### Functions of Viewfile Window

---

#### hiGetTextSourceLength

```
hiGetTextSourceLength(
 w_windowId
)
=> x_length / nil
```

#### Description

Returns the length of the source in a window.

#### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

#### Values Returned

|                 |                                                         |
|-----------------|---------------------------------------------------------|
| <i>x_length</i> | The length of the text.                                 |
| <i>nil</i>      | <i>w_windowId</i> is invalid. A warning is also issued. |

#### ***Related Topics***

[Miscellaneous Viewfile Functions](#)

## hiGetTextLineColumn

```
hiGetTextLineColumn(
 w_windowId
 x_location
)
=> l_lineColumn / nil
```

### Description

Returns the line and column number for an index location in a window.

### Arguments

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <i>w_windowId</i> | Specifies the window ID.                               |
| <i>x_location</i> | Character index location in which you are interested . |

### Values Returned

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| <i>l_lineColumn</i> | A list of the line and column numbers.                                                            |
| <i>nil</i>          | <i>w_windowId</i> is invalid or the character location is out of range. A warning is also issued. |

### Related Topics

[Miscellaneous Viewfile Functions](#)



## hiGetTextIndexLoc

```
hiGetTextIndexLoc (
 w_windowId
 x_line
 x_column
)
=> x_location / nil
```

### Description

Returns the character index location of a line and column number in a window.

### Arguments

|                   |                              |
|-------------------|------------------------------|
| <i>w_windowId</i> | Specifies the window ID.     |
| <i>x_line</i>     | Specifies the line number.   |
| <i>x_column</i>   | Specifies the column number. |

### Values Returned

|                   |                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------|
| <i>x_location</i> | The character location.                                                                                   |
| <i>nil</i>        | <i>w_windowId</i> is invalid or the line number or column number is out of range. A warning is displayed. |

### ***Related Topics***

Miscellaneous Viewfile Functions

## hiGetTextDispLoc

```
hiGetTextDispLoc(
 w_windowId
)
=> l_topBottom / nil
```

### Description

Returns the top and bottom character locations of the current visible region of a window.

### Argument

|                   |                          |
|-------------------|--------------------------|
| <i>w_windowId</i> | Specifies the window ID. |
|-------------------|--------------------------|

### Values Returned

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <i>l_topBottom</i> | A list that contains the top and bottom character locations. |
| <i>nil</i>         | <i>w_windowId</i> is invalid. A warning is also issued.      |

### ***Related Topics***

[Miscellaneous Viewfile Functions](#)

---

## Online Help Mechanism

---

This chapter provides information about the online help mechanism and describes the following topics:

The online help mechanism described in this chapter handles help requests from applications integrated into the Virtuoso environment.

You can request help by

- Clicking a *Help* button on a form, dialog box, or window.
- Selecting a form field or menu item and pressing the `Help` key on your keyboard (or the `F1` key, if your keyboard does not have a `Help` key).

Virtuoso responds to your help request by sending contextual information to the help service function, `hiHelp()`. `hiHelp()` uses this information to do one of the following:

- Display online help information in Doc Assistant using tag files (`.tgf` files) and the Doc Assistant Application Programming Interface (API).

Doc Assistant is the Cadence online documentation system, which displays HTML pages. Task Assistant is a quick-help interface that is used to display task- or feature-specific help. Task Assistant is integrated with Doc Assistant.

- Execute the callback function associated with the help request, if any.

As a user interface developer, you can control the contextual information that is sent to `hiHelp()` in the following ways:

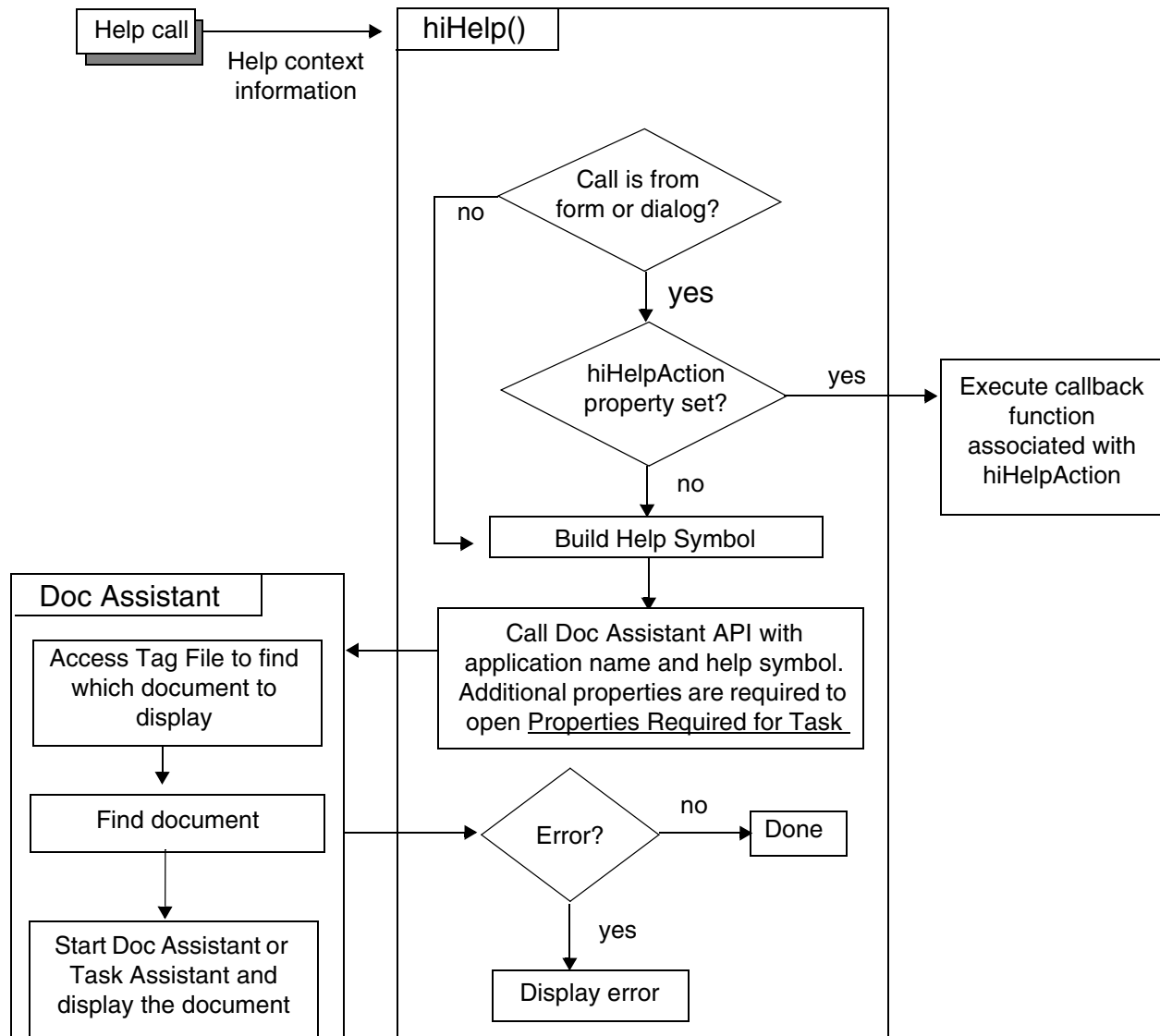
- By specifying the `help` argument when you create a form with `hiCreateAppForm()`, a dialog box with `hiDisplayAppDBox()`, or a window with `hiCreateWindow()` or `hiDisplayWindow()`.
- By setting the `hiHelpAppName` and `hiHelpSymName` properties.
- By setting the Properties Required for Task Assistant.

## Cadence User Interface SKILL Reference

### Online Help Mechanism

- By setting the `hiHelpAction` property on a form or dialog box, or specifying a custom help menu item for a window, and providing your own callback function (if you do not want to use `.tgf` files and Doc Assistant).

The following diagram illustrates how the help system works.



**Note:** Cadence no longer supports help response files (`.HRF` files) and recommends that you convert your old `.HRF` files to `.tgf` files.

## **Cadence User Interface SKILL Reference**

### Online Help Mechanism

---

#### ***Related Topics***

[Specify Your Own Callback for Help](#)

[Online Help Functions](#)

[Debugging Help](#)

## Form and Dialog Box Help

You can specify the `applID` and help symbol for forms and dialog boxes in two ways:

- By specifying the `g_help` argument of `hiCreateAppForm()` or `hiDisplayAppDBox()` when you create the form or dialog box

If you want to set the `applID` only, specify a single string or symbol as the value of `g_help`. This sets the `hiHelpAppName` property.

If you want to set both the `applID` and the help symbol, specify a list of two strings or symbols as the value of `g_help`. The first element in the list sets the `hiHelpAppName` property and the second element sets the `hiHelpSymName` property.

If you want to specify the help symbol only, specify a list of two strings or symbols as the value of `g_help`, with `nil` as the first element of the list and the help symbol as the second element. This sets the `hiHelpSymName` property without setting the `hiHelpAppName` property.

- By attaching the `hiHelpAppName` and `hiHelpSymName` properties to the symbol that represents the form name or dialog box name, after the form or dialog box is created

For example:

```
putprop('formSymbol' propValue 'hiHelpAppName)
putprop('formSymbol' propValue 'hiHelpSymName)
```

You can also specify `nil` as the property value to clear any previously-set properties.

When a user clicks *Help*, `hiHelp()` is called. `hiHelp()` first checks for the `hiHelpSymName` and `hiHelpAppName` and properties. If `hiHelpSymName` is set, its value is used as the help symbol, otherwise `hiHelp()` uses the form name or dialog box name as the help symbol.

If `hiHelpAppName` is set, its value is used as the `applID`. If `hiHelpAppName` is not set or its value is `nil`, `hiHelp()` calls `hiGetAppType()` to get the application name of the current window, which it uses as the `applID`. If the application name on the current window is `nil`, `hiHelp()` uses `dfii` as the `applID`.

When multiple applications share forms and dialog boxes or when applications are encapsulated, you cannot rely on `hiGetAppType()` to return the correct application name. Therefore, you should set `hiHelpAppName` for all forms and dialog boxes. If one application sets this property, all other applications that share the form or dialog box should also set this property or assign it to `nil` before displaying the form or dialog box.

You need to set a specific set of properties to open a document in the Task Assistant interface instead of Doc Assistant.

## Window Help

You can specify the help symbol for windows in the following ways:

- By specifying the `t_help` argument of `hiCreateWindow()` or `hiOpenWindow()` when you create the window. This value becomes the return value of `hiGetHelp()`, which is passed to `hiHelp()`.
- By placing the `hiHelpSymName` property on a symbol of the same name as the help string you specified as the `t_help` argument, after the window is created.

For example:

```
hiOpenWindow(?appType "MyApp" ?help "MyAppHelp")
putprop('MyAppHelp "MyHelpSymbol" 'hiHelpSymName)
```

You can specify the `appID` by putting the `hiHelpAppName` property on a symbol of the same name as the help string you specified as the `t_help` argument to `hiCreateWindow()` or `hiOpenWindow()`. For example:

```
hiOpenWindow(?appType "MyApp" ?help "MyAppHelp")
putprop('MyAppHelp "MyApp" 'hiHelpAppName)
```

When a user clicks *Help* in a window, the `hiGetHelp()` function is called and its return value is passed to `hiHelp()`. If you specified the `t_help` argument of `hiCreateWindow()` or `hiOpenWindow()`, its value is used as the return value of `hiGetHelp()`. If you did not specify this argument, the return value of `hiGetHelp()` is the value of the `t_appType` argument that you set when you created the window.

For windows help to work correctly, either the `t_help` or `appType` argument or both must be set when you create the window.

`hiHelp()` first calls `hiGetCommandName()` to see if a command is running; if it is, the return value of `hiGetCommandName()` is used as the help symbol. If a command is not running, `hiHelp()` checks the help string it received from `hiGetHelp()` for the `hiHelpSymName` and `hiHelpAppName` properties. If the help string has the `hiHelpSymName` property set, its value is used as the help symbol, otherwise `hiHelp()` uses the help string as the help symbol.

If the help string has the `hiHelpAppName` property set, its value is used as the `appID`. If the property is not set or its value is `nil`, `hiHelp()` calls `hiGetAppType()` to get the application name of the current window, which it uses as the `appID`. If the application name on the current window is `nil`, `hiHelp()` uses `dfii` as the `appID`.

## Menu Help

You can specify the help symbol and applID for a menu by setting the `hiHelpSymName` and `hiHelpAppName` properties on the menu symbol. You can use the `putprop()` function to set the properties. For example:

```
putprop('myMenuSymbol "MyHelpSymbol" 'hiHelpSymName)
putprop('myMenuSymbol "MyApp" 'hiHelpAppName)
```

When a user selects a menu and presses the *Help* key (or the *F1* key if the keyboard does not have a *Help* key), `hiHelp()` is called. The help that is invoked applies to the entire menu, not just to a menu item.

`hiHelp()` checks the menu symbol for the `hiHelpSymName` and `hiHelpAppName` properties. If the `hiHelpSymName` property is set, its value is used as the help symbol. Otherwise the menu symbol is used as the help symbol.

If the `hiHelpAppName` property is set, its value is used as the applID. If the property is not set or its value is `nil`, `hiHelp()` calls `hiGetAppType()` to get the application name of the current window, which it uses as the applID. If the application name on the current window is `nil`, `hiHelp()` uses `dfii` as the applID.

### ***Related Topics***

[Properties Required for Task Assistant](#)



## Properties Required for Task Assistant

By default, `hiHelp()` opens the Doc Assistant interface. If the help content for a form, dialog, window, or menu is required to be displayed in the Task Assistant interface, you can set some additional properties to open Task Assistant.

The following table describes the properties required for Task Assistant:

| Property                        | Purpose                                                                                                                                                                                                                                                                                                                           |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>hiHelpAsstTool</code>     | A string value specifying the name of the <code>.tgf</code> file that contains a mapping between help symbols and help documents for the Task Assistant                                                                                                                                                                           |
| <code>hiHelpAsstTag</code>      | A string value specifying the help symbol to be searched in the <code>.tgf</code> file                                                                                                                                                                                                                                            |
| <code>hiHelpAsstContext</code>  | <p>A string value specifying the current context of the tool. This helps in establishing context-sensitivity while calling help.</p> <p>This is an optional property. Use it to provide a unique help symbol for the currently active tab on a form or docked window, or the active section or element in the user interface.</p> |
| <code>hiHelpAsstTitle</code>    | <p>A string value specifying a title for Task Assistant</p> <p>This is an optional property. If not specified, <code>hiHelp()</code> takes the title from the form, dialog box, or docked window, and adds the <code>Help Assistant</code> suffix to it.</p>                                                                      |
| <code>hiHelpAsstPosition</code> | <p>A list with X and Y coordinates of the top left corner of Task Assistant</p> <p>This is an optional property. By default, Task Assistant is placed at X=100 and Y=300.</p>                                                                                                                                                     |

If both `hiHelpAsstTool` and `hiHelpAsstTag` are specified, `hiHelp()` looks for the help symbol in the `.tgf` file specified by `hiHelpAsstTool`. If these properties are not set, their values are incorrect, or no `.tgf` file is found with the name mentioned by `hiHelpAsstTool`, `hiHelp()` switches to the default help search mechanism by using the `hiHelpSymName` and `hiHelpAppName` properties.

The following example shows how to specify the properties for a form:

```
putprop('formSymbol "myTAHelp" 'hiHelpAsstTool)
putprop('formSymbol "createPorts" 'hiHelpAsstTag)
putprop('formSymbol "Fault Analysis Task Assistant" 'hiHelpAsstTitle)
```

## Cadence User Interface SKILL Reference

### Online Help Mechanism

---

The following example shows how to specify a help symbol for a window and set the properties required to open Task Assistant:

```
hiOpenWindow(?appType "MyApp" ?help "MyAppHelp")
putprop('MyAppHelp "myTAHelp" 'hiHelpAsstTool)
putprop('MyAppHelp "MyHelpSymbol" 'hiHelpAsstTag)
```

### ***Related Topics***

[Form and Dialog Box Help](#)

## Specify Your Own Callback for Help

If you do not want to use the default `.tgf` help file mechanism, you can specify your own action for the *Help* button on forms and dialog boxes by providing a callback function and by setting the `hiHelpAction` property. When a user clicks *Help*, the `hiHelp()` function is called. `hiHelp()` then executes the callback function associated with the `hiHelpAction` property, instead of referencing a `.tgf` file.

You cannot set the `hiHelpAction` property on windows.

### Setting the `hiHelpAction` Property on a Form or Dialog Box

- ➔ After creating a form or dialog box, add the `hiHelpAction` property to the symbol that represents the form or dialog box name. Use the following syntax:

```
putprop(formOrDBoxSymbol propertyValue 'hiHelpAction)
```

where *formOrDBoxSymbol* is the symbol that represents the form or dialog box name and *propertyValue* is the name (symbol or string) of the callback function that you provide.

For example, if your form symbol is `'myForm` and your callback function is `myHelpCB()`, you would do the following:

```
putprop('myForm 'myHelpCB 'hiHelpAction)
```

If you do not want your callback function to be passed the default arguments, you must specify the value of `hiHelpAction` as a string that contains parentheses.

### Writing a Callback Function for a Form or Dialog Box

If you want your callback function to be passed the default arguments,

- ➔ Create a callback function that takes four arguments, each of which is a symbol.

The callback function will be passed the following arguments:

|                        |                                          |
|------------------------|------------------------------------------|
| <code>objType</code>   | <code>'form</code> or <code>'dbox</code> |
| <code>objName</code>   | Symbol name of the form or dialog box    |
| <code>objDetail</code> | Always <code>nil</code>                  |

## Cadence User Interface SKILL Reference

### Online Help Mechanism

---

|                         |                                                                                                                                                                                                                                                                                             |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>moreDetail</code> | <code>nil</code> if the <i>Help</i> button is clicked<br><br>Symbol name of the field if the <i>Help</i> or <i>F1</i> key is pressed when the cursor is over a field<br><br>' <code>hiFormNoField</code> if the <i>Help</i> or <i>F1</i> key is pressed when the cursor is not over a field |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

These are the same arguments that are passed to `hiHelp()`.

If you do not want your callback function to be passed the default arguments,

1. Create the callback function to your own specifications.
2. Set the value of the `hiHelpAction` property to a string that contains parentheses. See Example 2 below for details.

Your callback function does not have to return any specific values—`hiHelp()` ignores the return values from your callback.

```
myForm = hiCreateAppForm(?name 'myForm ...)
putprop('myForm 'myHelpCB 'hiHelpAction)
procedure(myHelpCB(objType objName objDetail moreDetail "ssgg")
 hiDisplayAppDBox(?name 'myHelpDBox ?dboxBanner "My Help"
 ?dboxText strcat("My Help Text for "
 get_pname(objType) ""
 get_pname(objName)))

myForm2 = hiCreateAppForm(?name 'myForm2 ...)
putprop('myForm2 "myHelpFunc(\"no help for myForm2\")" 'hiHelpAction)
procedure(myHelpFunc(helpText "t")
 hiDisplayAppDBox(?name 'myHelpDBox2 ?dboxBanner "My Other Help"
 ?dboxText helpText)
)
```

## Specifying Your Own Callback for Windows

You cannot set the `hiHelpAction` property on windows. If you want to specify your own help callback for windows, you can override the default *Help* button with a pulldown menu and add your own callback as a menu item.

To specify your own callback for windows,

1. Write your callback function.
2. Create a menu item with `hiCreateMenuItem()` and specify your callback function as the `?callback` argument.
3. Create a menu with `hiCreateMenu()` and specify "Help" as the menu title.

## Cadence User Interface SKILL Reference

### Online Help Mechanism

---

4. Insert the menu in the window banner with `hiInsertBannerMenu()`.

Specify 99 as the value of the *x\_menuPosition* argument to ensure that the *Help* menu is displayed at the right end of the banner.

#### ***Related Topics***

Properties Required for Task Assistant

hiHelp

hiCreateMenu

hiCreateMenuItem

## Online Help Functions

This section describes the SKILL functions that support the help mechanism. While these functions are available for your use, you typically do not need to use them directly. They are described here because they are used by the default help system.

The following SKILL functions are described in this section:

- [hiHelp](#)
- [hiLookupHRFtag](#)
- [hiModalDBoxHelp](#)
- [hiModelessDBoxHelp](#)
- [hiObShowTag](#)

# Cadence User Interface SKILL Reference

## Online Help Mechanism

---

### hiHelp

```
hiHelp(
 s_objType
 S_objName
 [g_objDetail]
 [g_moreDetail]
)
=> t / nil
```

### Description

Constructs a help symbol from its arguments and invokes the Doc Assistant API to display the help document associated with the help symbol. If the `hiHelpAction` property is set, `hiHelp` executes the callback associated with the property, instead of calling the Doc Assistant API. `hiHelp` is the main service routine for the online help system.

Clicking on a help button in a Virtuoso application invokes a call to `hiHelp()`. You can also execute `hiHelp()` from the CIW, as a callback from a form selection menu, or as a call from any SKILL procedure.

### Arguments

|                                                                   |                                                                                                                                                                                                                                                  |
|-------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>s_objType</i>                                                  | Symbol that identifies the category of the help request.<br><i>s_objType</i> is one of the following: <code>'form</code> , <code>'menu</code> ,<br><code>'command</code> , <code>'window</code> , <code>'dbox</code> , <code>'helpError</code> . |
| <i>S_objName</i>  <br><i>g_objDetail</i>  <br><i>g_moreDetail</i> | The meanings of these arguments depend upon the value of<br><i>objType</i> as defined below:                                                                                                                                                     |

| objType | objName   | objDetail | moreDetail                                                                                                                                                                                                                                                                                                     |
|---------|-----------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 'form   | form name | nil       | Returns <code>nil</code> if the <i>Help</i> button is clicked.<br><br>Displays symbol name of the field if the <i>Help</i> or <i>F1</i> key is pressed when the cursor is over a field.<br><br><code>'hiFormNoField</code> if the <i>Help</i> or <i>F1</i> key is pressed when the cursor is not over a field. |
| 'menu   | menu name | nil       | NA                                                                                                                                                                                                                                                                                                             |

## Cadence User Interface SKILL Reference

### Online Help Mechanism

| objType    | objName                                                                 | objDetail        | moreDetail                                                                                                                                                                                                                                                                                                                |
|------------|-------------------------------------------------------------------------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 'command   | command name                                                            | command argument | NA                                                                                                                                                                                                                                                                                                                        |
| 'window    | help string set on the window or the application name set on the window | window ID        | Displays the current command name.                                                                                                                                                                                                                                                                                        |
| 'dbox      | dialog box name                                                         | NA               | Returns <code>nil</code> if the <i>Help</i> button is clicked.<br><br>Displays the symbol name of the field if the <i>Help</i> or <i>F1</i> key is pressed when the cursor is over a field.<br><br>Returns <code>'hiFormNoField</code> if the <i>Help</i> or <i>F1</i> key is pressed when the cursor is not over a field |
| 'helpError | error name                                                              | NA               | NA                                                                                                                                                                                                                                                                                                                        |

### Values Returned

`t` The function is successful.

`nil` There was an error.

### Examples

In the following menu, `myMenu`:

Selection 1  
Selection 2  
Selection 3  
HELP

the *HELP* menu item is programmed with the following callback:

```
hiHelp('menu "myMenu")
```

When a user selects *HELP*, the request generates the help symbol `myMenu`. If the application ID is `exampleApp`, then the Doc Assistant API looks for the help symbol `myMenu` in `exampleApp.tgf` and displays the document associated with it.



## Cadence User Interface SKILL Reference

### Online Help Mechanism

---

#### Additional Information

At first, `hiHelp()` checks the `hiHelpAsstTool` and `hiHelpAsstTag` properties of `objName`. properties. If valid values are found for these properties, `hiHelp()` uses these values to reference a `.tgf` file and displays the help document in Task Assistant.

If these properties are not set or are set to invalid values, `hiHelp()` proceeds as explained below:

- Uses the `objName` and, in some cases, the `objDetail` to identify the help symbol and `appID` that are used to reference a `.tgf` file.
- Checks the `objName` for the `hiHelpSymName` property. If the property is set, `hiHelp()` uses its value as the help symbol. Applications can set this property on the symbol that is passed as the `objName`. If it is not set, `hiHelp()` constructs the help symbol. This feature is useful for those forms and windows that are defined dynamically and where the name of the form or window is not known until run time.
- Checks the `objName` for the `hiHelpAppName` property. Applications can control the name of their own application-specific `.tgf` file by setting this property on the object that is passed as the `objName`. If this property is set and its value is not `nil`, `hiHelp` uses its value as the `appID`.

If `hiHelpAppName` is not set, `hiHelp()` gets the `appID` by calling `hiGetAppType()` on the current window, which returns the application name of the current window.

When multiple applications share forms and dialog boxes or when applications are encapsulated, you cannot rely on `hiGetAppType()` to return the correct application name. Therefore, you should set `hiHelpAppName` for all forms and dialog boxes. If one application sets this property, all the other applications that share the form or dialog box should also set this property or assign it to `nil` before displaying the form or dialog box.

- Converts the `appID` to lowercase, removes leading and trailing spaces, and changes any spaces within it to underscores.
- Determines the name of the `.tgf` file by appending the `.tgf` suffix to the `appID`.

#### Related Topics

#### Online Help Functions

## hiLookupHRFtag

```
hiLookupHRFtag(
 S_tagName
 S_fileName
)
=> l_command / nil
```

### Description

Returns the command that has been defined as the help response for a help tag in a .HRF file. You can get more information about the usage of this function from the *your\_install\_dir/tools/dfII/samples/local/hrfHelp.il* file.

### Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>S_tagName</i>  | The help tag for which you want to find the help response.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>S_fileName</i> | The .HRF file in which you want to look for the help response. The contents of this file must be sorted in C sort order and must be in the following format:<br><br><i>tagName command</i><br><br>where <i>tagName</i> is the name of the help tag and <i>command</i> is the SKILL function that defines the response to the help request. <i>tagName</i> cannot contain spaces. Anything following <i>tagName</i> (including spaces) is considered a SKILL command. Each <i>tagName</i> entry can span only one line. |

### Value Returned

|                  |                                                                                                                                                                                              |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_command</i> | The command that has been defined as the help response for the help tag.                                                                                                                     |
| <i>nil</i>       | The help tag does not exist in the file specified; the file does not exist; or the file is not sorted in C sort order, in which case the tag may not be found even if it exists in the file. |

### Additional Information

Cadence no longer supports help response files (.HRF files) and recommends that you convert your old .HRF files to .tgf files.

## Cadence User Interface SKILL Reference

### Online Help Mechanism

---

If you want to use your old .HRF files:

1. Read and follow the instructions in the following file:

`your_install_dir/tools/dfII/samples/local/hrfHelp.il`

2. Load the `hrfHelp.il` file.

There is also a new function, `hiLookupHRFtag`, which you can use to look up help tags in .HRF files. The function is described below; for an example of its usage, see the `hrfHelp.il` file.

### ***Related Topics***

[Online Help Functions](#)

## hiModalDBoxHelp

```
hiModalDBoxHelp(
 t_helpString
 [t_dboxName]
)
=> t / nil
```

### Description

Displays a message in a modal dialog box. A modal dialog box is a blocking dialog box—it waits for an acknowledgement before allowing the application to continue.

hiModalDBoxHelp is a service routine for the online help system.

### Arguments

|                     |                                                                                                                                                                                                                                                                                                                         |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_helpString</i> | The string to be formatted into the modal dialog box.                                                                                                                                                                                                                                                                   |
| <i>t_dboxName</i>   | The name of the modal dialog box. If you do not provide this argument, the dialog box is assigned the name <code>modalDBoxHelp</code> . Clicking the <i>Help</i> button on this dialog box generates a call to <code>hiHelp</code> with <code>dbox</code> as <i>objType</i> and the dialog box name as <i>objName</i> . |

### Values Returned

|            |                              |
|------------|------------------------------|
| <i>t</i>   | The dialog box is displayed. |
| <i>nil</i> | There was an error.          |

### Examples

Displays the message "This is a help message." in a modal dialog box.

```
hiModalDBoxHelp("This is a help message.")
```

### Related Topics

#### Online Help Functions

## hiModelessDBoxHelp

```
hiModelessDBoxHelp(
 t_helpString
 [t_dboxName]
)
=> t / nil
```

### Description

Displays a message in a modeless dialog box. A modeless dialog box is a non-blocking dialog box—it does not wait for an acknowledgement before allowing the application to continue.

hiModelessDBoxHelp is a service routine for the online help system.

### Arguments

|                     |                                                                                                                                                                                                                                                                                                                                       |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_helpString</i> | The string to be formatted into the modeless dialog box.                                                                                                                                                                                                                                                                              |
| <i>t_dboxName</i>   | The name of the modeless dialog box. If you do not provide this argument, the dialog box is assigned the name <code>modelessDBoxHelp</code> . Clicking the <i>Help</i> button on this dialog box generates a call to <code>hiHelp</code> with <code>dbox</code> as the <i>objType</i> and the dialog box name as the <i>objName</i> . |

### Values Returned

|            |                             |
|------------|-----------------------------|
| <i>t</i>   | The function is successful. |
| <i>nil</i> | There was an error.         |

### Examples

Displays the message "This is a help message." in a modeless dialog box.

```
hiModelessDBoxHelp("This is a help message.")
```

### Related Topics

#### Online Help Functions

## hiObShowTag

```
hiObShowTag(
 t_tool
 t_tag
 t_version
)
=> t / nil
```

### Description

Opens Doc Assistant to display the help document related to the specified tool and tag.

### Arguments

|                  |                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>t_tool</i>    | Name of the tool from which the help is being called. This maps to the name of a tag file in the documentation hierarchy. |
| <i>t_tag</i>     | Name of the tag used to reference the documentation page to open.                                                         |
| <i>t_version</i> | Version of the help document. Use "" to open the document in the default, usually the latest, version.                    |

### Values Returned

|            |                                    |
|------------|------------------------------------|
| <i>t</i>   | Help document opened successfully. |
| <i>nil</i> | Failed to open the help document.  |

### Examples

Opens the latest version of the documentation page referenced by the `schObjPropForm` tag entry in the `Schematics.tgf` file.

```
hiObShowTag("Schematics" "schObjPropForm" "")
```

### ***Related Topics***

#### Online Help Functions

#### hiHelp

## Debugging Help

To display information about any help call in the Command Interpreter Window (CIW), do the following:

1. In the CIW, select *Options – Log Filter*.
2. In the Set Log File Display Filter form, select the *Show Output: user* option.

The online help system prints various messages in the CIW, including the help symbol and application ID, for each help request.

For more detailed debugging information:

- ➔ In the CIW, set the `helpDebug` variable by typing:

```
helpDebug = t
```

Whenever `helpDebug` is set to anything except `nil`, the help mechanism displays additional debugging information in the CIW.

### ***Related Topics***

[Online Help Functions](#)

## **Cadence User Interface SKILL Reference**

### Online Help Mechanism

---



---

## QtReplay Functions

---

This topic provides a list of Cadence® SKILL functions associated with QtReplay, that log and replay graphical user interface (GUI) actions. QtReplay functions have the prefix `nr`. These functions are used for GUI elements that are created directly with Qt instead of user interface SKILL (HI). These GUI could be a part of either a mixed HI/Qt application or a non-HI based Qt application.

Like other replay functions, you do not call these functions directly. They are logged in the log file when the user performs an action, such as clicking *OK* in a dialog box. When the file is replayed, the functions are called.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [nrDialogAccept](#)
- [nrGetIndexesFromSelection](#)
- [nrGetSelectedIndexes](#)
- [nrListColumnMove](#)
- [nrListColumnSort](#)
- [nrListItemAction](#)
- [nrListItemExpand](#)
- [nrListItemRename](#)
- [nrResizeWindow](#)
- [nrSetCurrentWindow](#)
- [nrSetFileFilter](#)
- [nrSetWindowVisibility](#)
- [nrUpdateMenu](#)

## nrDialogAccept

```
nrDialogAccept(
 o_dialogBox
)
=> t / nil
```

### Description

Function logged when a dialog box is accepted, when the *Return* key is pressed or buttons such as *OK*, *Yes*, or *Done* in the dialog box are clicked. This function is logged only for native Qt dialog boxes. This function is intended only for replay use and should not be called explicitly.

### Argument

|                    |                      |
|--------------------|----------------------|
| <i>o_dialogBox</i> | The dialog box name. |
|--------------------|----------------------|

### Values Returned

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The dialog was accepted or this function was called from SKILL to replay the action of accepting the dialog. |
| <i>nil</i> | There was an error.                                                                                          |

### Related Topics

[QtReplay Functions](#)

## nrGetIndexesFromSelection

```
nrGetIndexesFromSelection(
 t_widget
 l_selection
)
=> l_indices / nil
```

### Description

Expands the specified compact list representing a selection of items from other lists or tree widgets into the corresponding expanded list representation detailing each selected item.

The compact list representation is created when you select a range of items in a list or a tree widget, or is returned when you query the `value` property of a widget in the following way:

`form->tree->value`



This function applies only to Cadence widgets created in Qt with the native replay layer. Using this function with other widget types, such as widgets created using HI SKILL functions, will result in an error.

### Arguments

|                          |                                                                       |
|--------------------------|-----------------------------------------------------------------------|
| <code>o_widget</code>    | Name of the widget for which the expanded selection is required.      |
| <code>l_selection</code> | Compact list representing a selection of other lists or tree widgets. |

### Values Returned

|                        |                                                                           |
|------------------------|---------------------------------------------------------------------------|
| <code>l_indices</code> | List describing each selected index in the specified list or tree widget. |
| <code>nil</code>       | The operation failed.                                                     |

### Examples

Selects an adjacent range of items (`Item_1` to `Item_5`) in a tree widget:

```
range = form->treeWidget->value
```

## Cadence User Interface SKILL Reference

### QtReplay Functions

---

```
> (("Item_1" "Item_5"))
```

The variable `range` stores only the selected items from the extended range, instead of returning one item for each selected index.

If the range is stored in a variable, the indexes in that range can be queried in the following way:

```
nrGetIndexesFromSelection(form->treeWidget range)
```

The names of both the widget and the range are required. The widget is required to expand the compact list to a full list of indexes. The current selection in the widget does not impact the current selection or the return value from this function.

Returns the following list of expanded indexes:

```
(("Item_1" 0)
 ("Item_2" 0)
 ("Item_3" 0)
 ("Item_4" 0)
 ("Item_5" 0)
)
```

Each element in the list is itself a list representing a single selected index in a list or a tree widget, comprising an identifier for the row (usually the text in the first column), and the column number.

Most item views that select the full row (all columns) use 0 as the column number

Item views that feature more granular selection (allowing individual row/column combinations) use a non-zero column number.

### ***Related Topics***

[QtReplay Functions](#)

## nrGetSelectedIndexes

```
nrGetSelectedIndexes(
 t_widget
)
=> l_indices / nil
```

### Description

Returns a list comprising the selected indexes of lists or tree widgets.

The compact list representation is created when you select a range of items in a list or a tree widget, or is returned when you query the `value` property of a widget in the following way:

```
form->tree->value
```

This function can be used to obtain the expanded list.



This function applies only to Cadence widgets created in Qt with the native replay layer. Using this function with other widget types, such as widgets created using HI SKILL functions will, result in an error.

### Arguments

|                       |                                                                  |
|-----------------------|------------------------------------------------------------------|
| <code>o_widget</code> | Name of the widget for which the expanded selection is required. |
|-----------------------|------------------------------------------------------------------|

### Values Returned

|                        |                                                                           |
|------------------------|---------------------------------------------------------------------------|
| <code>l_indices</code> | List describing each selected index in the specified list or tree widget. |
| <code>nil</code>       | The operation failed.                                                     |

### Examples

Selects an adjacent range of items (Item\_1 to Item\_5) in a tree widget:

```
range = form->treeWidget->value
> ("Item_1" "Item_5")
```

## Cadence User Interface SKILL Reference

### QtReplay Functions

---

The variable `range` stores only the selected items from the extended range, instead of returning one item for each selected index.

Retrieve the extended list of selected indexes directly from the widget.

```
nrGetSelectedIndexes(form->treeWidget)
```

Returns the following list of expanded indexes:

```
(("Item_1" 0)
 ("Item_2" 0)
 ("Item_3" 0)
 ("Item_4" 0)
 ("Item_5" 0)
)
```

Each element in the list is itself a list representing a single selected index in a list or a tree widget, comprising an identifier for the row (usually the text in the first column), and the column number.

Most item views that select the full row (all columns) use 0 as the column number

Item views that feature more granular selection, allowing individual row/column combinations, use a non-zero column number.

### ***Related Topics***

[QtReplay Functions](#)

## nrListColumnMove

```
nrListColumnMove(
 o_listView
 x_position
 x_newPosition
)
=> t / nil
```

### Description

Function logged when a column in a list/tree view (created with the Qt `QListView` class) is moved. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                            |                                     |
|----------------------------|-------------------------------------|
| <code>o_listView</code>    | The list view.                      |
| <code>x_position</code>    | The current position of the column. |
| <code>x_newPosition</code> | The new position of the column.     |

### Values Returned

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | A column in a list-tree view was moved or this function was called from SKILL to replay the moving of the column. |
| <code>nil</code> | There was an error.                                                                                               |

### ***Related Topics***

[QtReplay Functions](#)

[nrListColumnSort](#)

[nrListItemAction](#)

[nrListItemRename](#)

[nrListItemExpand](#)

## nrListColumnSort

```
nrListColumnSort(
 o_listView
 x_position
)
=> t / nil
```

### Description

Function logged when a column in a list/tree view, created with the Qt `QListView` class, is sorted. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>o_listView</code> | The list view.                             |
| <code>x_position</code> | The position of the column that is sorted. |

### Value Returned

|                  |                                                                                                                            |
|------------------|----------------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | A column in a list-tree view was sorted or this function was called from SKILL to replay the action of sorting the column. |
| <code>nil</code> | There was an error.                                                                                                        |

### ***Related Topics***

[QtReplay Functions](#)

[nrListColumnMove](#)

[nrListItemAction](#)

[nrListItemRename](#)

[nrListItemExpand](#)



## nrListItemAction

```
nrListItemAction(
 o_listView
 t_position
)
=> t / nil
```

### Description

Function logged when the action associated with an item in a list/tree view created with the Qt `QListView` class is invoked by double-clicking on the item. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                   |                                                                     |
|-------------------|---------------------------------------------------------------------|
| <i>o_listView</i> | The list view.                                                      |
| <i>t_position</i> | The index of the item in the list for which the action was invoked. |

### Values Returned

|            |                                                                                                                                        |
|------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | The action associated with a list view item was invoked or this function was called from SKILL to replay the invocation of the action. |
| <i>nil</i> | There was an error.                                                                                                                    |

### ***Related Topics***

[QtReplay Functions](#)

[nrListItemRename](#)

[nrListColumnMove](#)

[nrListColumnSort](#)

[nrListItemExpand](#)

## nrListItemExpand

```
nrListItemExpand(
 o_listView
 t_listItem
 g_expandedOrCollapsed
)
=> t / nil
```

### Description

Function logged when an item in a list/tree view created with the Qt `QListView` class is expanded or collapsed. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                              |                                                                                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <i>o_listView</i>            | The list view.                                                                                                                            |
| <i>t_listItem</i>            | The list item that is expanded or collapsed. The item name is relative to the root node of the tree, for example: "NewLib/buffer/symbol". |
| <i>g_expandedOrCollapsed</i> | <i>t</i> or <i>nil</i> . <i>t</i> indicates the list item was expanded, <i>nil</i> indicates the list item was collapsed.                 |

### Values Returned

|            |                                                                                          |
|------------|------------------------------------------------------------------------------------------|
| <i>t</i>   | A list-tree view item was expanded or collapsed, or this function was called from SKILL. |
| <i>nil</i> | There was an error.                                                                      |

### Examples

```
nrListItemExpand(cdsLibManager->listTree "NewLib/buffer" t)
```

### Related Topics

[QtReplay Functions](#)

[nrListColumnMove](#)

## Cadence User Interface SKILL Reference

### QtReplay Functions

---

nrListColumnSort

nrListItemAction

nrListItemRename

## nrListItemRename

```
nrListItemRename(
 o_listView
 t_itemName
 x_columnIndex
 t_newItemName
)
=> t / nil
```

### Description

Function logged when an item in a list/tree view (created with the Qt `QListView` class) is renamed. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                            |                            |
|----------------------------|----------------------------|
| <code>o_listView</code>    | The list view.             |
| <code>t_itemName</code>    | The name of the item.      |
| <code>x_columnIndex</code> | The column index.          |
| <code>t_newItemName</code> | The new name for the item. |

### Values Returned

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | A list view item was renamed or this function was called from SKILL to replay the action of renaming of the item. |
| <code>nil</code> | There was an error.                                                                                               |

### ***Related Topics***

[QtReplay Functions](#)

[nrListItemAction](#)

[nrListColumnMove](#)

[nrListColumnSort](#)

[nrListItemExpand](#)

## nrResizeWindow

```
nrResizeWindow(
 o_window
 l_coordinates
)
=> t / nil
```

### Description

Function logged when a window is resized or moved. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                      |                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------|
| <i>o_window</i>      | The window.                                                                                                  |
| <i>l_coordinates</i> | The window's new screen coordinates, specified as a list of the following format:<br><br>'(x y width height) |

### Values Returned

|            |                                                                       |
|------------|-----------------------------------------------------------------------|
| <i>t</i>   | A window was resized or moved, or this function was called from SKILL |
| <i>nil</i> | There was an error.                                                   |

### Examples

```
nrResizeWindow(cdsLibManager list(107 35 551 483))
```

### Related Topics

[QtReplay Functions](#)

[nrSetCurrentWindow](#)

[nrSetWindowVisibility](#)

## **nrSetCurrentWindow**

```
nrSetCurrentWindow(
 o_window
 [t_subWindowName]
)
=> t / nil
```

### **Description**

Function logged when the current window is set. This function is intended only for replay use and should not be called explicitly.

### **Arguments**

|                        |                                                                                                         |
|------------------------|---------------------------------------------------------------------------------------------------------|
| <i>o_window</i>        | The window that has become the current window.                                                          |
| <i>t_subWindowName</i> | Optional argument that specifies the subwindow name in the context of a QWorkspace multi-window widget. |

### **Values Returned**

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| <i>t</i>   | The current window is set or this function was called from SKILL. |
| <i>nil</i> | There was an error.                                               |

### **Examples**

```
nrSetCurrentWindow(cdsLibManager)
```

### ***Related Topics***

[QtReplay Functions](#)

[nrResizeWindow](#)

[nrSetWindowVisibility](#)

## nrSetFileFilter

```
nrSetFileFilter(
 o_dialog
 t_filter
)
=> t / nil
```

### Description

Function logged when a file filter is selected in a file dialog box. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| <i>o_dialog</i> | The dialog box that contains the file filter list.          |
| <i>t_filter</i> | A string specifying the file filter item that was selected. |

### Values Returned

|            |                                                                                                                        |
|------------|------------------------------------------------------------------------------------------------------------------------|
| <i>t</i>   | A file filter is set for the dialog or this function was called from SKILL to replay the action of setting the filter. |
| <i>nil</i> | There was an error.                                                                                                    |

### Examples

```
nrSetFileFilter(cdsLibManager->LoadDefaultEnvironment "All Files (*)")
```

### ***Related Topics***

[QtReplay Functions](#)

## nrSetWindowVisibility

```
nrSetWindowVisibility(
 o_window
 t_showOrHide
)
=> t / nil
```

### Description

Function logged when a window is displayed, removed from the screen, or iconified. When a window is displayed, the second argument of this function is the string `show`, when a window is removed, the second argument is `hide`, and when a window is iconified, the second argument is `min`. This function is intended only for replay use and should not be called explicitly.

### Arguments

|                           |                                                                                                                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>o_window</code>     | The window.                                                                                                                                                                                |
| <code>t_showOrHide</code> | When a window is displayed, this argument is the string "show"; when a window is removed from the screen, this argument is "hide"; and when a window is minimized, this argument is "min". |

### Values Returned

|                  |                                                                                       |
|------------------|---------------------------------------------------------------------------------------|
| <code>t</code>   | A window was displayed, closed, or iconified, or this function was called from SKILL. |
| <code>nil</code> | There was an error.                                                                   |

### Examples

```
nrSetWindowVisibility(basicRenameForm "show")
```

### ***Related Topics***

[QtReplay Functions](#)

[nrResizeWindow](#)

[nrSetCurrentWindow](#)



## nrUpdateMenu

```
nrUpdateMenu(
 o_popupMenu
)
=> t / nil
```

### Description

Function logged when the contents of a dynamic popup menu are modified. This function is logged for menus derived from the `nrReplayPopupMenu` class, which is a part of the native Qt replay library. The function is logged whenever the `contentsChanged` signal is emitted from the menu. This function is intended only for replay use and should not be called explicitly.

### Argument

|                          |                 |
|--------------------------|-----------------|
| <code>o_popupMenu</code> | The popup menu. |
|--------------------------|-----------------|

### Values Returned

|                  |                                                                                                                                     |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <code>t</code>   | A popup menu's contents were modified, or this function was called from SKILL to replay the action of the menu's contents changing. |
| <code>nil</code> | There was an error.                                                                                                                 |

### Examples

```
nrUpdateMenu(cdsLibManager->popupMenu)
```

### ***Related Topics***

[QtReplay Functions](#)

## **Cadence User Interface SKILL Reference**

### QtReplay Functions

---

---

## Performance Diagnostic Functions

---

This topic provides a list of Cadence® SKILL functions associated with the Health Monitor tool.

Many factors can affect the performance of tools or applications running in the Virtuoso custom IC design platform. Identifying the root cause is a collaborative, iterative, and lengthy process to narrow down the root cause of the issue. This process can be iterative and lengthy. The diagnostic function lets you run various tests on the system to identify issues more effectively.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [perfDiagInstall](#)
- [perfRefreshControl](#)
- [perfStartCallStackProc](#)
- [perfStopCallStackProc](#)

## Cadence User Interface SKILL Reference

### Performance Diagnostic Functions

---

#### perfDiagInstall

```
perfDiagInstall
 g_RdOnly
 [?openCallstack g_openCallstack | ?loadOnly g_loadOnly]
=> t / nil
```

#### Description

Installs the Health Monitor tool if it is not installed by default during the Virtuoso Studio launch. You can use this tool to diagnose various performance issues in the Virtuoso Studio design platform, such as recording callstacks and terminating an unresponsive Virtuoso Studio session.

To open the Health Monitor tool by default, add the following statement to the `.cdsinit` file:

```
perfDiagInstall(?openCallstack t)
```

#### Arguments

*g\_RdOnly*

A boolean that installs the Health Monitor tool intended for the Cadence support team for on-site debugging.

The default is `nil`.

*?openCallstack g\_openCallstack*

A boolean that displays the Health Monitor tool to perform tasks, such as recording callstacks, terminating frozen sessions, and so on.

When set to `t`, the Health Monitor tool can be pinned to the top of the screen to save screen space.

When set to `nil` and if the tool freezes, you can open the Health Monitor tool from the command-line using the `cdsPerfDiag` command, or by clicking the *Find* button on the control form of another Virtuoso Studio session.

The default is `nil`.

## Cadence User Interface SKILL Reference

### Performance Diagnostic Functions

---

`?loadOnly g_loadOnly`

A boolean that loads the SKILL menu only.

**Note:** Cadence recommends using this option only when your machine is slow for a typical installation.

If you specify `t`, you must manually open the Health Monitor tool for the first time, before the tool freezes.

The default is `nil`.

### Values Returned

`t` The installation of the Health Monitor tool is successful.

`nil` The installation of the Health Monitor tool is not successful.

### Examples

Installs the Health Monitor tool and opens it before the tool freezes:

```
perfDiagInstall(?openCallstack t)
=> t
```

### ***Related Topics***

[Diagnostic Center Overview](#)

## **perfRefreshControl**

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

### **Description**

Refreshes the Health Monitor form to apply the changes made using the Performance Diagnostic environment variables.

### **Argument**

None

### **Values Returned**

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | The operation fails.         |

### **Examples**

While changing the `perf` environment variables, it is mandatory to call the `perfRefreshControl` function to refresh the control UI in order to take effect for the modified environment variables.

```
envSetVal("perf" "enableExpertMode" `boolean t)
=> perfRefreshControl()
```

### ***Related Topics***

[Collecting Data Using Health Monitor Tool](#)

## **perfStartCallStackProc**

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

### **Description**

Starts collecting the callstacks using the Health Monitor tool.

### **Arguments**

None

### **Values Returned**

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | The operation fails.         |

### **Examples**

You can specify a bindkey to start collecting the callstacks.

```
hiSetBindKey("Layout" "3" "perfStartCallStackProc()")
=> t
```

### ***Related Topics***

[Collecting Data Using Health Monitor Tool](#)

## Cadence User Interface SKILL Reference

### Performance Diagnostic Functions

---

#### **perfStopCallStackProc**

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

#### **Description**

Stops the Health Monitor tool from collecting the callstacks.

#### **Arguments**

None

#### **Values Returned**

|     |                              |
|-----|------------------------------|
| t   | The operation is successful. |
| nil | The operation fails.         |

#### **Examples**

You can specify a bindkey as shown below to stop collecting the callstacks.

```
hiSetBindKey("Layout" "4" "perfStopCallStackProc()")
=> t
```

#### ***Related Topics***

[Collecting Data Using Performance Diagnostic Tool](#)



---

## Display Lists

---

Display lists are used to construct and display pictures within the Virtuoso user interface. display lists are collections of shapes displayed within a form, menu, button, or window. Additionally, display lists are used by the DAG Browser to display pictures.

Display lists use pens to hold the graphical properties a shape is drawn with. A pen stores color, line style, and fill style. Pens are referred to by number. When you add a shape to a display list, you specify which pen number it should use.

Display lists are built in two ways:

- By calling the SKILL functions to add objects to the display list.
- By drawing the picture in the icon editor. You can save the picture in a file and load it whenever you need the display list again.

The display list is composed of the following objects:

- Points
- Segments
- Boxes
- Arcs
- Circles
- Donuts
- Polygons
- Paths
- RasterText
- StrokeText
- SkillObjects

## Cadence User Interface SKILL Reference

### Display Lists

---

#### ■ EventObjects

Display lists are not hierarchical in that a display list cannot include another display list.

Coordinates are Cartesian and specified using integer numbers. The range of the integers can be anything you wish. The display list is scaled to fit into the space provided.

## Add Objects to the Display List

This topic provides a list of Cadence® SKILL functions associated with adding objects to the display list.

Objects are added to the display list in the following manner:

```
dlAddObject(w_dlist x_penNumber..... [s_tagSymbol])
=> t | nil
```

The first two arguments are always the same. *w\_dlist* is the display list the object should be added to, and *x\_penNumber* is the number of the pen the object will use when it is drawn.

Pens are collected into a group called a pen table. If you use the default pen table, then there are 64 pens available to you. The pen numbers start at 0; therefore, if you use the default pen table, you can use numbers ranging from 0 to 63. The *s\_tagSymbol* is an optional symbol given so you can refer to this shape again. It is always the final argument to a *dlAdd* function. If you define your own pen table, then you can use fewer or more pens; the more pens you create, the more memory is required.

The coordinates are specified using integer numbers and can be any range you choose. All coordinates specified assume 0:0 to be the lower-left corner of the display list object. The display list is automatically scaled to fit into a given area, much like a *geFull* function does with a database object. You should, however, try to choose your coordinate space so it isn't too small. This can avoid integer round-off when converting from display list coordinates to screen coordinates.

All *dlAddObject* functions return *t* on success, and *nil* on failure. Conditions for failure are either an invalid *dlist* argument, or a pen number outside a valid range.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [dlAddArc](#)
- [dlAddBox](#)
- [dlAddCircle](#)
- [dlAddDonut](#)
- [dlAddEventObject](#)
- [dlAddPath](#)
- [dlAddPoint](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

- dlAddPolygon
- dlAddRasterText
- dlAddSegment
- dlAddSkillObject
- dlAddStrokeText
- dlMakeDisplayList
- dlSetClearOnDraw

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddArc

```
dlAddArc (
 w_dlist
 x_penNumber
 l_point1
 l_point2
 n_startAngle
 n_sweepAngle
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds an arc to the display list.

#### Arguments

|                                        |                                                                                                                                                                                                               |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dlist</i>                         | Display list to add to.                                                                                                                                                                                       |
| <i>x_penNumber</i>                     | Pen number to use.                                                                                                                                                                                            |
| <i>l_point1</i> and<br><i>l_point2</i> | The lower left and upper right coordinates of the bounding box enclosing the arc and the ellipse containing the arc.                                                                                          |
| <i>n_startAngle</i>                    | The starting angle, specified in degrees, measured such that 0 degrees is the 3 o'clock position.                                                                                                             |
| <i>n_sweepAngle</i>                    | Measured in degrees, from the starting angle that the arc will span. A positive degree sweeps the arc in an counterclockwise direction. A negative <i>sweepAngle</i> sweeps the arc in a clockwise direction. |
| <i>s_tagSymbol</i>                     | Symbol to tag this object with.                                                                                                                                                                               |

#### Value Returned

|            |                       |
|------------|-----------------------|
| <i>t</i>   | Operation successful. |
| <i>nil</i> | Operation failed.     |

#### Related Topics

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddBox

```
dlAddBox (
 w_dlist
 x_penNumber
 l_point1
 l_point2
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds a box to the display list.

#### Arguments

|                                        |                                      |
|----------------------------------------|--------------------------------------|
| <i>w_dlist</i>                         | Display list to add to.              |
| <i>x_penNumber</i>                     | Pen number to use.                   |
| <i>l_point1</i> and<br><i>l_point2</i> | Any two opposite corners of the box. |
| <i>s_tagSymbol</i>                     | Symbol to tag this object with.      |

#### Value Returned

|            |                       |
|------------|-----------------------|
| <i>t</i>   | Operation successful. |
| <i>nil</i> | Operation failed.     |

#### ***Related Topics***

[Add Objects to the Display List](#)

## **dlAddCircle**

```
dlAddCircle(
 w_dlist
 x_penNumber
 l_point
 x_radius
 [s_tagSymbol]
)
=> t / nil
```

### **Description**

Adds a circle to the display list.

### **Arguments**

|                    |                                 |
|--------------------|---------------------------------|
| <i>w_dlist</i>     | Display list to add to.         |
| <i>x_penNumber</i> | Pen number to use.              |
| <i>l_point</i>     | Center point of the circle.     |
| <i>x_radius</i>    | Radius of the circle.           |
| <i>s_tagSymbol</i> | Symbol to tag this object with. |

### **Value Returned**

|            |                       |
|------------|-----------------------|
| <i>t</i>   | Operation successful. |
| <i>nil</i> | Operation failed.     |

### ***Related Topics***

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddDonut

```
dlAddDonut (
 w_dlist
 x_penNumber
 l_point
 x_innerRadius
 x_outerRadius
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds a donut to the display list.

#### Arguments

|                      |                                          |
|----------------------|------------------------------------------|
| <i>w_dlist</i>       | Display list to add to.                  |
| <i>x_penNumber</i>   | Pen number to use.                       |
| <i>l_point</i>       | Center of the donut.                     |
| <i>x_innerRadius</i> | Radius of the inner circle of the donut. |
| <i>x_outerRadius</i> | Radius of the outer circle of the donut. |
| <i>s_tagSymbol</i>   | Symbol to tag this object with.          |

#### Value Returned

|            |                       |
|------------|-----------------------|
| <i>t</i>   | Operation successful. |
| <i>nil</i> | Operation failed.     |

#### ***Related Topics***

Add Objects to the Display List



## dlAddEventObject

```
dlAddEventObject (
 w_dlist
 l_point1
 l_point2
 t_procName
 t_procArgs
 [t_highlightMode]
 [g_doesGraphics]
 [s_tagSymbol]
)
=> t / nil
```

### Description

Adds an *EventObject* to the display list. *EventObjects* are SKILL procedures called whenever the user presses a mouse button or a keyboard key. If the *s\_doesGraphics* argument is *t*, the *EventObject* is called whenever the display list is drawn.

### Arguments

|                                     |                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dlist</i>                      | Display list to add to.                                                                                                                                                                                                                                                                                                                                                                    |
| <i>l_point1</i> and <i>l_point2</i> | The lower-left and upper-right coordinates of a box, within which the <i>SkillObject</i> will draw. The display list processing functions use this box to determine the visibility of the <i>SkillObject</i> . The procedure you write to define the <i>SkillObject</i> must be written to draw and then return as quickly as possible. If not, the performance of the system will suffer. |

## Cadence User Interface SKILL Reference

### Display Lists

---

*t\_procName*

SKILL function to call.

The *t\_procName* procedure is called in the following manner:

```
t_procName(l_point1 l_point2 s_eventType
t_eventDetail t_procArgs)
```

*s\_eventType* is one of the following symbols:

*draw* (the display list is drawing)

*button1* (the left mouse button was pressed)

*button2* (the middle mouse button was pressed)

*button3* (the right mouse button was pressed)

*key* (a key was pressed)

In the case of the *key* *eventType*, the *t\_eventDetail* tells you which key was pressed. Otherwise, the *eventDetail* argument is *nil*.

*t\_procArgs*

Arguments passed to the function.

*t\_highlightMode*

When the user enters into an event object, you can specify the system to draw a box around the sensitive area, or invert the sensitive area, or to do nothing. These are specified by *box* (or *b*), *invert* (or *i*), or *none*. The default value is *none*.

*g\_doesGraphics*

Tells the display list processing functions whether you use *dlDraw* functions inside your SKILL procedure or not. *dlDraw* functions are discussed later on. If you don't call *dlDraw* functions, the rendering of your display list will be faster. This argument should be *t* or *nil*.

*s\_tagSymbol*

Symbol to tag this object with.

### Value Returned

*t*

Operation successful.

*nil*

Operation failed.

### Related Topics

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddPath

```
dlAddPath(
 w_dlist
 x_penNumber
 x_pathWidth
 t_pathEndType
 l_pointsList
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds a path to the display list.

#### Arguments

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dlist</i>       | Display list to add to.                                                                                                                                                                                                                                                                                                                                                                                            |
| <i>x_penNumber</i>   | Pen number to use.                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>x_pathWidth</i>   | Width of the path.                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>t_pathEndType</i> | End type can have one of three values: <code>Truncate</code> , <code>Extend</code> , or <code>Round</code> . <code>Truncate</code> means the path will be square-ended and stop at the two endpoints. <code>Extend</code> means the path will be square-ended, but it will extend 1/2 the width from each end point. <code>Round</code> means the path will have a rounded endpoint and stop at the two endpoints. |
| <i>l_pointsList</i>  | Coordinates of the center line of the path.                                                                                                                                                                                                                                                                                                                                                                        |
| <i>s_tagSymbol</i>   | Symbol to tag this object with.                                                                                                                                                                                                                                                                                                                                                                                    |

#### Value Returned

|            |                       |
|------------|-----------------------|
| <i>t</i>   | Operation successful. |
| <i>nil</i> | Operation failed.     |

#### Related Topics

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddPoint

```
dlAddPoint(
 w_dlist
 x_penNumber
 l_point
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds a point to the display list.

#### Arguments

|                    |                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------|
| <i>w_dlist</i>     | Display list to add to.                                                                       |
| <i>x_penNumber</i> | Pen number to use.                                                                            |
| <i>l_point</i>     | List containing the point. For example,<br><code>list(200 200)</code> or <code>200:200</code> |
| <i>s_tagSymbol</i> | Symbol to tag this object with.                                                               |

#### Value Returned

|            |                       |
|------------|-----------------------|
| <i>t</i>   | Operation successful. |
| <i>nil</i> | Operation failed.     |

#### ***Related Topics***

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddPolygon

```
dlAddPolygon(
 w_dlist
 x_penNumber
 l_pointsList
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds a polygon to the display list.

#### Arguments

|                     |                                                                                                                                                                                                                                                                 |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dlist</i>      | Display list to add to.                                                                                                                                                                                                                                         |
| <i>x_penNumber</i>  | Pen number to use.                                                                                                                                                                                                                                              |
| <i>l_pointsList</i> | <p>Polygon which is a list of points comprising its vertices. For example, the list of points can be specified in either of these ways:</p> <pre>'((0 0) (100 180) (200 200))<br/>list( 0:0 100:180 200:200)</pre> <p>You do not need to close the polygon.</p> |
| <i>s_tagSymbol</i>  | Symbol to tag this object with.                                                                                                                                                                                                                                 |

#### Related Topics

[Add Objects to the Display List](#)

## **dlAddRasterText**

```
dlAddRasterText (
 w_dlist
 x_penNumber
 l_point
 t_text
 t_fontName
 [t_verticalJustify]
 [t_horizontalJustify]
 [s_tagSymbol]
)
=> t / nil
```

### **Description**

Adds a text string to the display list.

### **Arguments**

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dlist</i>             | Display list to add to.                                                                                                                                                                                                                                                                                                                                                                                                         |
| <i>x_penNumber</i>         | Pen number to use.                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>l_point</i>             | Origin point of the text.                                                                                                                                                                                                                                                                                                                                                                                                       |
| <i>t_text</i>              | Text string to be added to the display list                                                                                                                                                                                                                                                                                                                                                                                     |
| <i>t_fontName</i>          | X font used to draw text. The font name can vary from system to system. Check your <code>/usr/lib/X11/fonts</code> directory see what fonts are available, or ask your system administrator. Usually fonts <code>fixed</code> and “9x15” are available. If the font name you specify is not found, then <code>fixed</code> is used. Raster text does not scale, but has a fixed width and height.                               |
| <i>t_verticalJustify</i>   | <p>Presentation of the text, and one of the following strings:</p> <ul style="list-style-type: none"><li>■ “dlcTop” (the top of the text will be at the y of the point)</li><li>■ “dlcMiddle” (the center of the text will be at the y of the point)</li><li>■ “dlcBottom” (the bottom of the text will be at the y of the point)</li></ul> <p>If not specified, <i>t_verticalJustify</i> is set to <code>dlcBottom</code>.</p> |
| <i>t_horizontalJustify</i> |                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Cadence User Interface SKILL Reference

### Display Lists

---

Presentation of the text, and one of the following strings:

- "dlcLeft" (the string will start at the x of the point)
- "dlcCenter" (the middle of the string will be at the x of the point)
- "dlcRight" (the end of the string will be at the x of the point)

If not specified, *t\_horizontalJustify* is set to dlcLeft.

*s\_tagSymbol*

Symbol to tag this object with.

### ***Related Topics***

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dlAddSegment**

```
dlAddSegment (
 w_dlist
 x_penNumber
 l_point1
 l_point2
 [s_tagSymbol]
)
=> t / nil
```

#### **Description**

Adds a line segment to the display list.

#### **Arguments**

|                    |                                 |
|--------------------|---------------------------------|
| <i>w_dlist</i>     | Display list to add to.         |
| <i>x_penNumber</i> | Pen number to use.              |
| <i>l_point1</i>    | First point of the segment.     |
| <i>l_point2</i>    | Second point of the segment.    |
| <i>s_tagSymbol</i> | Symbol to tag this object with. |

#### ***Related Topics***

[Add Objects to the Display List](#)



## dlAddSkillObject

```
dlAddSkillObject (
 w_dlist
 x_penNumber
 l_point1
 l_point2
 t_procName
 t_procArgs
 [s_tagSymbol]
)
=> t / nil
```

### Description

Adds a *SkillObject* to the display list. *SkillObjects* are SKILL procedures called whenever the object should be drawn.

### Arguments

|                                                    |                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>w_dlist</code>                               | Display list to add to.                                                                                                                                                                                                                                                                                                                                                                   |
| <code>x_penNumber</code>                           | Pen number to use.                                                                                                                                                                                                                                                                                                                                                                        |
| <code>l_point1</code> and<br><code>l_point2</code> | The lower-left and upper-right coordinates of a box within which the <i>SkillObject</i> will draw. The display list processing functions use this box to determine the visibility of the <i>SkillObject</i> . The procedure you write to define the <i>SkillObject</i> must be written to draw and then return as quickly as possible. If not, the performance of the system will suffer. |
| <code>t_procName</code>                            | SKILL function to call.                                                                                                                                                                                                                                                                                                                                                                   |
| <code>t_procArgs</code>                            | Arguments passed to the function. The display list processing routines call your function in the following way:<br><br><code>t_procName( l_point1 l_point2 t_procArgs)</code>                                                                                                                                                                                                             |
| <code>s_tagSymbol</code>                           | Symbol to tag this object with.                                                                                                                                                                                                                                                                                                                                                           |

### Related Topics

[Add Objects to the Display List](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlAddStrokeText

```
dlAddStrokeText (
 w_dlist
 x_penNumber
 l_point
 t_text
 t_justification
 t_fontName
 x_height
 [t_orientation]
 [g_drafting]
 [s_tagSymbol]
)
=> t / nil
```

#### Description

Adds a stroke text string to the display list.

#### Arguments

|                        |                                                                                                                                                             |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dlist</i>         | Display list to add to.                                                                                                                                     |
| <i>x_penNumber</i>     | Pen number to use.                                                                                                                                          |
| <i>l_point</i>         | Origin point of the text                                                                                                                                    |
| <i>t_text</i>          | Text string to be added to the display list                                                                                                                 |
| <i>t_justification</i> | One of the following strings: upperLeft, centerLeft, left, lowerLeft, upperCenter, center, lowerCenter, upperRight, centerRight, right, lowerRight          |
| <i>t_fontName</i>      | One of the following font names: euro, gothic, math, roman, script, stick, or swedish                                                                       |
| <i>x_height</i>        | Height of text.                                                                                                                                             |
| <i>t_orientation</i>   | One of the following values: 0, 90, 180, 270, mx, sideways, my, upsideDown, mr90, sideways, mr270, sideways&270. Default value is "0".                      |
| <i>g_drafting</i>      | If this is set to t, the label will always be readable from left to right, or top to bottom. It will never appear mirrored or backward. The default is nil. |

## Cadence User Interface SKILL Reference

### Display Lists

---

*s\_tagSymbol*                      Symbol to tag this object with.

#### ***Related Topics***

Add Objects to the Display List

## **dlMakeDisplayList**

```
dlMakeDisplayList(
)
 => w_displayList / nil
```

### **Description**

Creates an empty display list you can add objects to.

### **Value Returned**

|                      |                   |
|----------------------|-------------------|
| <i>w_displayList</i> | The display list. |
| nil                  | Operation failed. |

### **Examples**

Creates a display list to draw inside a window and to use as its icon.

```
/* First, make the DisplayList. */
dl = dlMakeDisplayList()
/* Now make a pen table for it. */
penTable = dlMakePenTable(5)
/* Assign the penTable to the display list. */
dlSetPenTable(dl penTable)
/* Define a couple of colors. */
colorIndex = hiStringToIcon(nameToColor("blue"))
dlSetPenColor(1 colorIndex penTable)
/* Set this pen "filled." */
dlSetPenFillStyle(1 "SolidFill" penTable)
/* Put the objects in. */
dlAddBox(dl 1 0:0 100:100)
dlAddBox(dl 2 10:10 90:90)
dlAddCircle(dl 3 50:50 20)
dlAddStrokeText(dl 4 50:50 "Window" "0" "gothic" 25 "center")
/* Make a window. Put the DisplayList into the window */
/* and set the window's icon. */
w = hiOpenWindow()
dlAttachDlistToWindow(dl w)
/* Convert the dlist to an icon. */
icon = dlDlistToIcon(dl 50 50)
hiSetWindowIcon(w icon)
/* Save it to a file. */
dlSaveDlist(dl "save.dlist" "newDl")
/* Try resizing the window. The dlist will fit inside it. */
/* Iconify the window. */
```

## Cadence User Interface SKILL Reference

### Display Lists

---

#### ***Related Topics***

[Add Objects to the Display List](#)

## **dlSetClearOnDraw**

```
dlSetClearOnDraw(
 w_dlist
 x_widgetId
 g_draw
)
=> t / nil
```

### **Description**

Specifies whether the area the display list is drawn in is cleared before the display list is drawn.

### **Arguments**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>w_dlist</i>    | The display list.                                 |
| <i>x_widgetId</i> | The widget the display list is attached to.       |
| <i>g_draw</i>     | Clears the area before the display list is drawn. |

### ***Related Topics***

[Add Objects to the Display List](#)

## Display List Draw Functions

This topic provides a list of Cadence® SKILL functions associated with Display list Draw. These `dlDraw` functions are only used within a *SkillObject* procedure. They render shapes to the screen. The display list processing functions transform the coordinates of your *SkillObject* to the box you specified when you added it to the display list. Therefore, the coordinates you specify when calling a `dlDraw` function are specified such that 0:0 is coincident with the lower-left corner of the SKILL object.

For example,

```
dlAddSkillObject(dlist 4 100:100 200:200
 "mySkillObject" "10")
dlAddSkillObject(dlist 4 300:300 400:400
 "mySkillObject" "20")
```

The display list functions call your procedure:

```
mySkillObject(100:100 200:200 10)
mySkillObject(300:300 400:400 20)
```

If you want to draw a circle and have the radius passed to you, define your *SkillObject* as:

```
procedure(mySkillObject(ll ur radius)
 prog(()
 dlDrawCircle(50:50 radius)
 return(t)
) ; end prog
) ; end procedure
```

There is a `dlDraw` function for all shapes the `dlAddObject` functions provide, except for *dlAddSkillObject*. The arguments to the *dlDraw* functions are the same as the `dlAddObject` functions, except for the *w\_dlist* and *x\_penNumber* arguments (which are not necessary, since the values from the SKILL object are used).

The `dlDraw` functions returns `t` when the object is drawn, and `nil` if the `dlDraw` function is called outside a *SkillObject* procedure.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [dlDrawArc](#)
- [dlDrawBox](#)
- [dlDrawCircle](#)
- [dlDrawDonut](#)
- [dlDrawPath](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

- dlDrawPoint
- dlDrawPolygon
- dlDrawRasterText
- dlDrawSegment
- dlDrawStrokeText



## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlDrawArc

```
dlDrawArc(
 l_point1
 l_point2
 f_startAngle
 f_sweepAngle
)
=> t / nil
```

#### Description

Draws an arc to the screen from within a *SkillObject* procedure.

#### Arguments

|                                        |                                                                                                                                                                                                          |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_point1</i> and<br><i>l_point2</i> | The lower-left and upper-right corners of a box enclosing the arc.                                                                                                                                       |
| <i>f_startAngle</i>                    | Measured such that 0 degrees is the 3 o'clock position.                                                                                                                                                  |
| <i>f_sweepAngle</i>                    | Measured in degrees, from the starting angle the arc will span. A positive degree sweeps the arc in an counterclockwise direction. A negative <i>sweepAngle</i> sweeps the arc in a clockwise direction. |

#### Related Topics

[Display List Draw Functions](#)

## **dlDrawBox**

```
dlDrawBox(
 l_point1
 l_point2
)
=> t / nil
```

### **Description**

Draws a box to the screen from within a *SkillObject* procedure.

### **Arguments**

|                                        |                                    |
|----------------------------------------|------------------------------------|
| <i>l_point1</i> and<br><i>l_point2</i> | Any two opposite corners of a box. |
|----------------------------------------|------------------------------------|

### **Examples**

```
dlDrawBox(' (0 0) ' (1 1))
```

### ***Related Topics***

[Display List Draw Functions](#)

## **dlDrawCircle**

```
dlDrawCircle(
 l_point
 x_radius
)
=> t / nil
```

### **Description**

Draws a circle to the screen from within a *SkillObject* procedure.

### **Arguments**

|                 |                         |
|-----------------|-------------------------|
| <i>l_point</i>  | Center point of circle. |
| <i>x_radius</i> | Radius of the circle.   |

### ***Related Topics***

[Display List Draw Functions](#)

## **dlDrawDonut**

```
dlDrawDonut (
 l_point
 x_innerRadius
 x_outerRadius
)
=> t / nil
```

### **Description**

Draws a donut to the screen from within a *SkillObject* procedure.

### **Arguments**

|                      |                          |
|----------------------|--------------------------|
| <i>l_point</i>       | Center point of donut.   |
| <i>x_innerRadius</i> | Inside radius of donut.  |
| <i>x_outerRadius</i> | Outside radius of donut. |

### ***Related Topics***

[Display List Draw Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlDrawPath

```
dlDrawPath(
 x_pathWidth
 t_pathEndType
 l_pointsList
)
=> t / nil
```

#### Description

Draws a path to the screen from within a *SkillObject* procedure. A path is specified by a center line and a width.

#### Arguments

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_pathWidth</i>   | Specifies the path width.                                                                                                                                                                                                                                                                                                                                                                                        |
| <i>t_pathEndType</i> | End type can have one of three values: <code>Truncate</code> , <code>Extend</code> or <code>Round</code> . <code>Truncate</code> means the path will be square-ended and stop at the two endpoints. <code>Extend</code> means the path will be square-ended, but it will extend 1/2 the width from each end point. <code>Round</code> means the path will have a rounded endpoint and stop at the two endpoints. |
| <i>l_pointsList</i>  | Coordinates of the center line.                                                                                                                                                                                                                                                                                                                                                                                  |

#### Related Topics

[Display List Draw Functions](#)

## **dlDrawPoint**

```
dlDrawPoint(
 l_point
)
=> t / nil
```

### **Description**

Draws a point to the screen from within a *SkillObject* procedure.

### **Argument**

|                |                        |
|----------------|------------------------|
| <i>l_point</i> | Position of the point. |
|----------------|------------------------|

### ***Related Topics***

[Display List Draw Functions](#)

## **dlDrawPolygon**

```
dlDrawPolygon(
 l_pointsList
)
=> t / nil
```

### **Description**

Draws a polygon to the screen from within a *SkillObject* procedure.

### **Argument**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <i>l_pointsList</i> | List of points comprising its vertices. You do not need to close the polygon. |
|---------------------|-------------------------------------------------------------------------------|

### ***Related Topics***

[Display List Draw Functions](#)

## **dIDrawRasterText**

```
dIDrawRasterText (
 l_point
 t_text
 t_fontName
 [t_verticalJustify]
 [t_horizontalJustify]
)
=> t / nil
```

### **Description**

Draws a text string to the screen from within a *SkillObject*. The text is drawn using an X Window System font.



## Cadence User Interface SKILL Reference

### Display Lists

---

#### Arguments

|                            |                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_point</i>             | Origin point of the text.                                                                                                                                                                                                                                                                                                                                                                                |
| <i>t_text</i>              | Text string to be added to the display list.                                                                                                                                                                                                                                                                                                                                                             |
| <i>t_fontName</i>          | X font used to draw text. The font name can vary from system to system. Check your <code>/usr/lib/X11/fonts</code> directory see what fonts are available, or ask your system administrator. Usually fonts "fixed" and "9x15" are available. If the font name you specify is not found, then "fixed" is used. Raster text does not scale, but has a fixed width and height.                              |
| <i>t_verticalJustify</i>   | <p>Presentation of the text, and one of the following strings:</p> <ul style="list-style-type: none"><li>■ "dlcTop" : Top of the text will be at the y of the point.</li><li>■ "dlcCenter": Center of the text will be at the y of the point.</li><li>■ "dlcBottom" : Bottom of the text will be at the y point.</li></ul> <p>If not specified, <i>t_verticalJustify</i> is set to "dlcBottom".</p>      |
| <i>t_horizontalJustify</i> | <p>Presentation of the text, and one of the following strings:</p> <ul style="list-style-type: none"><li>■ "dlcLeft" : String starts at the x of the point.</li><li>■ "dlcCenter" : Middle of the string is located at the x of the point.</li><li>■ "dlcRight" : End of the string is located at the x of the point.</li></ul> <p>If not specified, <i>t_horizontalJustify</i> is set to "dlcLeft".</p> |

#### Related Topics

[Display List Draw Functions](#)

## **dlDrawSegment**

```
dlDrawSegment(
 l_point1
 l_point2
)
=> t / nil
```

### **Description**

Draws a line segment to the screen from within a *SkillObject* procedure.

### **Arguments**

|                 |                                                                          |
|-----------------|--------------------------------------------------------------------------|
| <i>l_point1</i> | First point of the segment, assuming 0 : 0 to be the lower-left corner.  |
| <i>l_point2</i> | Second point of the segment, assuming 0 : 0 to be the lower-left corner. |

### ***Related Topics***

[Display List Draw Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlDrawStrokeText

```
dlDrawStrokeText(
 l_point
 t_text
 t_justification
 t_fontName
 x_height
 t_orientation
)
=> t / nil
```

#### Description

Draws a stroke text string to the screen from within a *SkillObject* procedure. The height and therefore scale can be specified.

#### Arguments

|                        |                                                                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>l_point</i>         | Origin point of the text.                                                                                                                          |
| <i>t_text</i>          | Text string to be added to the display list.                                                                                                       |
| <i>t_justification</i> | One of the following strings: upperLeft, centerLeft, left, lowerLeft, upperCenter, center, lowerCenter, upperRight, centerRight, right, lowerRight |
| <i>t_fontName</i>      | One of the following font names: euro, gothic, math, roman, script, stick, or swedish                                                              |
| <i>x_height</i>        | Height of text.                                                                                                                                    |
| <i>t_orientation</i>   | One of the following values: 0, 90, 180, 270, mx, sideways, my, upsideDown, mr90, sideways, mr270, sideways&270. Default value is "0".             |

#### Related Topics

[Display List Draw Functions](#)

## Display List Pen Functions

This topic provides a list of Cadence® SKILL functions associated with display list pens.

Display list pens store the graphic attributes of an object. These attributes are:

- `Color`
- `FillStyle`
- `FillPattern`
- `LineStyle`
- `LinePattern`

Pens are referred to by number. Pens are also grouped together into pen tables. In the default pen table, there are 64 pens and they are numbered 0 through 63.

When you create a display list, it is initialized to use a default pen table. Changes made to a pen table affect all the display lists that use that pen table, much like changing a technology file changes all the cells that use it. To allow you to have a unique pen table, functions are provided to create new pen tables and to specify that a display list should use a particular pen table.

The color of the pens is set by color index. As such, the display list pens do not allocate any colors, but share the colors already in the color table. Therefore, they must be considered read-only. In order to set a pen's color, you must call `hiStringToIcon` to find the color index that most closely matches the color you desire.

Fill patterns and Line patterns are different; you can create and change these patterns at will.

Pens also specify the order in which objects are rendered. When display lists are drawn, all the objects on pen 0 are drawn, then all the objects on pen 1 are drawn, and so on up to pen 63. As such, objects on pen 4 draw on top of objects on pen 3.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- `dlMakePenTable`
- `dlMakeStipple`
- `dlSetCurrentPen`
- `dlSetPenColor`

## Cadence User Interface SKILL Reference

### Display Lists

---

- dlSetPenFillStyle
- dlSetPenFilled
- dlSetPenStipple
- dlSetPenTable

## **dlMakePenTable**

```
dlMakePenTable(
 [x_maxPens]
)
=> w_penTable / nil
```

### **Description**

Creates a pen table a display list can use.

### **Argument**

|                  |                                                                                                                                                                                                                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_maxPens</i> | Number of pens in the pen table. It is best to create only as many pens as you plan to use; any unused pens waste memory. The pen numbers range from 0 to <i>x_maxPens</i> -1. The function returns the pen table, or <code>nil</code> if an error occurred. If not specified, the default number of pens is 64. |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### ***Related Topics***

[Display List Pen Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

## dlMakeStipple

```
dlMakeStipple(
 t_StippleString
)
=> x_stippleId / nil
```

### Description

Creates a stipple pattern from the supplied string.

### Argument

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_StippleString</i> | String that has a special form. The string is composed of 1s, 0s and newline characters <code>\n</code> and " " (space) characters. The string describes an array of bits that define a stipple pattern. Each newline <code>\n</code> or " " (space) character in the string begins a new row of bits. Each row must have the same number of characters in it, or the functions return <code>nil</code> . This is a sample stipple pattern: |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

```
stipString = "0101\n\
1010\n\
0101\n\
1010"
```

```
stippleId = dlMakeStipple(stipString)
```

In this example, a 4x4 stipple pattern is created and its ID returned. The 1 character specifies the bits turned on, and the 0 character returns the bits turned off. The maximum size of a pattern is 64x64.

### Values Returned

|                  |                                                  |
|------------------|--------------------------------------------------|
| <i>stippleID</i> | Returns <i>stippleID</i> of the stipple pattern. |
| <i>nil</i>       | Operation failed.                                |

### ***Related Topics***

[Display List Pen Functions](#)

## **dlSetCurrentPen**

```
dlSetCurrentPen(
 x_penNumber
)
=> t / nil
```

### **Description**

Sets the specified pen number to the current pen. Changes the pen number objects are drawn with from within a *SkillObject* procedure. This function should only be called from within a *SkillObject* procedure.

### **Argument**

|                    |                    |
|--------------------|--------------------|
| <i>x_penNumber</i> | Pen number to set. |
|--------------------|--------------------|

### ***Related Topics***

[Display List Pen Functions](#)



## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlSetPenColor

```
dlSetPenColor(
 x_penNumber
 x_colorIndex
 [w_penTable]
)
=> t / nil
```

#### Description

Changes the specified pen to the specified color index from the pen table.

#### Arguments

|                     |                                                                                                                                                                                                                                                              |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_penNumber</i>  | Pen number to use.                                                                                                                                                                                                                                           |
| <i>x_colorIndex</i> | Color desired.                                                                                                                                                                                                                                               |
| <i>w_penTable</i>   | Pen table this pen belongs to. If you do not specify the <i>w_penTable</i> , the default pen table is used. In order to obtain a desired color, you may use <code>hiStringToIcon</code> to find the color index that most closely matches the desired color. |

#### Value Returned

|     |                                     |
|-----|-------------------------------------|
| t   | The pen color is changed.           |
| nil | The pen color could not be changed. |

#### ***Related Topics***

[Display List Pen Functions](#)

## **dISetPenFillStyle**

```
dISetPenFillStyle(
 x_penNumber
 t_fillStyle
 [w_penTable]
)
=> t / nil
```

### **Description**

Sets the fill style of a pen.

### **Arguments**

|                    |                                                                                                                                                                                                                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_penNumber</i> | Pen number to be set.                                                                                                                                                                                                                                                                                                                                       |
| <i>t_fillStyle</i> | One of the following strings:<br><br>"OutlineNoFill":<br><br>"SolidFill": All objects are filled<br><br>"XFill": Similar to OutlineNoFill except boxes have an X drawn inside.<br><br>"StippleFill": Objects are filled with the stipple pattern.<br><br>"OutlineStippleFill": objects are drawn stipple filled with a solid line drawn around the outside. |
| <i>w_penTable</i>  | Specifies the pen table. If not specified, the default pen table is used.                                                                                                                                                                                                                                                                                   |

### **Value Returned**

|     |                                  |
|-----|----------------------------------|
| t   | The fill style was set.          |
| nil | The fill style could not be set. |

### ***Related Topics***

[Display List Pen Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlSetPenFilled

```
dlSetPenFilled(
 x_penNumber
 [w_penTable]
)
=> t / nil
```

#### Description

Sets the fill style of a pen to `SolidFill`.

#### Arguments

|                    |                                                                                       |
|--------------------|---------------------------------------------------------------------------------------|
| <i>x_penNumber</i> | Pen number to be set.                                                                 |
| <i>w_penTable</i>  | Pen table to use. If you do not specify this argument, the default pen table is used. |

#### Value Returned

|            |                                  |
|------------|----------------------------------|
| <i>t</i>   | The fill style was set.          |
| <i>nil</i> | The fill style could not be set. |

#### ***Related Topics***

[Display List Pen Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlSetPenStipple

```
dlSetPenStipple(
 x_penNumber
 x_stippleId
 [w_penTable]
)
=> t / nil
```

#### Description

Sets the stipple pattern for a pen.

#### Arguments

|                    |                                                                                                                                                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_penNumber</i> | Pen number to be set.                                                                                                                                                      |
| <i>x_stippleId</i> | Stipple pattern used.                                                                                                                                                      |
| <i>w_penTable</i>  | Pen table to set. If you do not supply <i>w_penTable</i> , then the pen in the default pen table will be set, otherwise the pen in the pen table specified by will be set. |

#### Value Returned

|     |                                   |
|-----|-----------------------------------|
| t   | Stipple pattern is set.           |
| nil | Stipple pattern could not be set. |

#### Related Topics

[Display List Pen Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlSetPenTable

```
dlSetPenTable(
 w_displayList
 w_penTableOrDisplayList
)
=> t / nil
```

#### Description

Sets the pen table a display list will use.

#### Arguments

*w\_displayList*            Display list you wish to set.

*w\_penTableOrDisplayList*

Either a pen table or another display list. If the argument is a display list, the *w\_displayList* is set to use the same pen table as the second *displayList*.

#### Value Returned

*t*                        Stipple pattern is set.

*nil*                     Stipple pattern could not be set.

#### ***Related Topics***

[Display List Pen Functions](#)

## View Display List Functions

This topic provides a list of Cadence® SKILL functions associated with different ways of viewing the display list. A display list can be viewed in empty windows. A single display list can be attached to as many windows as you wish. They are scaled to fit within a window.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [dlAttachDlistToWidget](#)
- [dlAttachDlistToWindow](#)
- [dlClearDisplayList](#)
- [dlCloseWidget](#)
- [dlConfigureButton](#)
- [dlDetachDlistFromWidget](#)
- [dlDetachDlistFromWindow](#)
- [dlDisplay](#)
- [dlDlistToIcon](#)
- [dlEnableItem](#)
- [dlGetRasterTextWidth](#)
- [dlFitDlistOnDraw](#)
- [dlMakeDlistButton](#)
- [dlMakeWidget](#)
- [dlMapWidget](#)
- [dlSetWidgetName](#)
- [dlUnMapWidget](#)

## **dlAttachDlistToWidget**

```
dlAttachDlistToWidget(
 w_displayList
 x_widgetId
)
=> x_widgetId / nil
```

### **Description**

Attaches a display list to a widget.

### **Arguments**

|                      |                                                  |
|----------------------|--------------------------------------------------|
| <i>w_displayList</i> | Display list to attach.                          |
| <i>x_widgetId</i>    | Specifies the widget to attach the display list. |

### **Value Returned**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>x_widgetId</i> | Display list is attached to the specified widget.     |
| <i>nil</i>        | Display list is not attached to the specified widget. |

### ***Related Topics***

[View Display List Functions](#)

## **dlAttachDlistToWindow**

```
dlAttachDlistToWindow(
 w_displayList
 w_windowId
)
=> t / nil
```

### **Description**

Attaches a display list to the window defined by *w\_windowId*. This function can be used to preview a display list.

### **Arguments**

|                      |                                                                               |
|----------------------|-------------------------------------------------------------------------------|
| <i>w_displayList</i> | Display list you wish to attach to the window.                                |
| <i>w_windowId</i>    | ID number of an empty window. It cannot be the window of another application. |

### **Value Returned**

|            |                                             |
|------------|---------------------------------------------|
| <i>t</i>   | Display list is attached to the window.     |
| <i>nil</i> | Display list is not attached to the window. |

### ***Related Topics***

[View Display List Functions](#)



## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dlClearDisplayList**

```
dlClearDisplayList(
 w_displayList
)
=> t / nil
```

#### **Description**

Removes all the objects from a display list.

#### **Argument**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <i>w_displayList</i> | Display list you are clearing of all objects. |
|----------------------|-----------------------------------------------|

#### **Value Returned**

|            |                                                |
|------------|------------------------------------------------|
| <i>t</i>   | Removes all the objects from the display list. |
| <i>nil</i> | Operation failed.                              |

#### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dlCloseWidget**

```
dlCloseWidget(
 x_widgetId
)
=> t / nil
```

#### **Description**

Closes or destroys a widget.

#### **Argument**

|                   |                                     |
|-------------------|-------------------------------------|
| <i>x_widgetId</i> | Specifies the widget ID to destroy. |
|-------------------|-------------------------------------|

#### **Value Returned**

|     |                                |
|-----|--------------------------------|
| t   | Destroys the specified widget. |
| nil | Operation failed.              |

#### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlConfigureButton

```
dlConfigureButton(
 x_button
 x_xPosition
 x_yPosition
 x_width
 x_height
)
=> t / nil
```

#### Description

Moves a button to another location. Both the x and y position are specified in X Window System coordinates, where 0:0 is the upper left corner of the screen.

#### Arguments

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <i>x_button</i>    | Button being moved.                                    |
| <i>x_xPosition</i> | New x position of the upper-left corner of the button. |
| <i>x_yPosition</i> | New y position of the upper-left corner of the button. |
| <i>x_width</i>     | Width of new button size.                              |
| <i>x_height</i>    | Height of new button size.                             |

#### ***Related Topics***

[View Display List Functions](#)

## **dlDetachDlistFromWidget**

```
dlDetachDlistFromWidget(
 w_displayList
 x_widgetId
)
=> t / nil
```

### **Description**

Detaches a display list from a widget.

### **Arguments**

|                      |                                         |
|----------------------|-----------------------------------------|
| <i>w_displayList</i> | Display list to detach.                 |
| <i>x_widgetId</i>    | Widget to detach the display list from. |

### ***Related Topics***

[View Display List Functions](#)

## **dlDetachDlistFromWindow**

```
dlDetachDlistFromWindow(
 w_displayList
 w_windowId
)
=> t / nil
```

### **Description**

Detaches a display list (*w\_displayList*) from the window defined by *w\_windowId*.

### **Arguments**

|                      |                                                                               |
|----------------------|-------------------------------------------------------------------------------|
| <i>w_displayList</i> | Display list you wish to detach from the window.                              |
| <i>w_windowId</i>    | ID number of an empty window. It cannot be the window of another application. |

### ***Related Topics***

[View Display List Functions](#)

## **dlDisplay**

```
dlDisplay(
 w_displayList
)
=> t / nil
```

### **Description**

Redraws the display list (*w\_displayList*) in all the windows and buttons it is currently attached to.

### **Argument**

|                      |                         |
|----------------------|-------------------------|
| <i>w_displayList</i> | Display list to redraw. |
|----------------------|-------------------------|

### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dIDlistToIcon**

```
dIDlistToIcon(
 w_displayList
 x_width
 x_height
 [x_backColorIndex]
)
=> l_iconId / nil
```

#### **Description**

Converts a displayList to an icon.

#### **Arguments**

|                         |                                                                                                                                                                                                                                                               |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_displayList</i>    | Display list.                                                                                                                                                                                                                                                 |
| <i>x_width</i>          | Width of icon specified in screen units.                                                                                                                                                                                                                      |
| <i>x_height</i>         | Height of icon specified in screen units.                                                                                                                                                                                                                     |
| <i>x_backColorIndex</i> | Background color of the generated icon. If you do not specify this argument, the icon will have the same background color as Graphic Editor windows. The returned <i>l_iconId</i> can be used in forms or as the window manager icon for a window or windows. |

#### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlEnableItem

```
dlEnableItem(
 w_dlist
 x_penNumber
 s_symbolTag
 g_enableOrDisable
)
=> t / nil
```

#### Description

Allows you to turn on or off shapes within a display list that match the provided symbol. All shapes on pen *x\_penNumber* are searched and checked to see if the *symbolTag* matches.

#### Arguments

|                          |                                                            |
|--------------------------|------------------------------------------------------------|
| <i>w_dlist</i>           | Display list to add.                                       |
| <i>x_penNumber</i>       | Pen number to turn shapes on or off.                       |
| <i>s_symbolTag</i>       | Symbol to match to all shapes on pen <i>x_penNumber</i> .  |
| <i>g_enableOrDisable</i> | <i>t</i> enables the shape; <i>nil</i> disables the shape. |

#### Related Topics

[View Display List Functions](#)



### dlGetRasterTextWidth

```
dlGetRasterTextWidth(
 t_fontName
 t_text
)
=> x_width / -1
```

### Description

Retrieves width of the text in pixels for the specified raster font.

### Arguments

*t\_fontName*

Name of a font, which can be:

- An X logical font descriptor (XLFD) such as  
`"*-courier-medium-r-*-12-*` and  
`"*-helvetica-medium-r-*-14-*`.
- A standard alias name such as `"6x13"` or `"fixed"`.
- A font type such as `"text"`, `"ciw"`, `"label"`, `"error"`, or `"help"`. When a font type is specified, the default font name used in IC6.1.6 or previous releases is used as shown below:

| font type | font name                                |
|-----------|------------------------------------------|
| -----     | -----                                    |
| text      | <code>*-courier-medium-r-*-12-*</code>   |
| ciw       | <code>*-courier-medium-r-*-12-*</code>   |
| label     | <code>*-helvetica-medium-r-*-12-*</code> |
| error     | <code>*-helvetica-medium-r-*-12-*</code> |
| help      | <code>*-helvetica-medium-r-*-14-*</code> |

*t\_text*

Text string for which the width is required.

## Cadence User Interface SKILL Reference

### Display Lists

---

#### Value Returned

|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| <code>x_width</code> | The width of the text in pixels for the specified raster font. |
| <code>-1</code>      | The font could not be found.                                   |

#### Examples

The following examples retrieve the widths of the specified raster texts:

```
dlGetRasterTextWidth("-*-courier-medium-r-*-*-12-*" "hello")
35
dlGetRasterTextWidth("text" "hello")
35
dlGetRasterTextWidth("-*-helvetica-medium-r-*-*-24-*" "hello")
51
```

#### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dIFitDlistOnDraw**

```
dIFitDlistOnDraw(
 w_displayList
 x_widgetId
 g_doFit
)
=> t / nil
```

#### **Description**

Specifies whether a display list should automatically fit within the specified widget.

#### **Arguments**

|                      |                                                    |
|----------------------|----------------------------------------------------|
| <i>w_displayList</i> | Display list.                                      |
| <i>x_widgetId</i>    | ID the display list is attached to.                |
| <i>g_doFit</i>       | The display list automatically fits in the widget. |

#### ***Related Topics***

[View Display List Functions](#)

## **dlMakeDlistButton**

```
dlMakeDlistButton(
 x_parentWidgetId
 x_xPosition
 x_yPosition
 x_width
 x_height
 w_displayList
 t_pressCallback
 [x_backgroundColor]
)
=> t / nil
```

### **Description**

Creates a button to display a display list in. The display list button is a rectangular region with a widget or window in which a display list is drawn and which dispatches a callback whenever the left mouse button is pressed within it.

### **Arguments**

|                          |                                                                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_parentWidgetId</i>  | Parent widget to make the button within.                                                                                                     |
| <i>x_xPosition</i>       | x position of the upper-left corner of the button, in X Window System screen coordinates (where 0:0 is the upper-left corner of the widget). |
| <i>x_yPosition</i>       | y position of the upper-left corner of the button, in X Window System screen coordinates (where 0:0 is the upper-left corner of the widget). |
| <i>x_width</i>           | Width of the button.                                                                                                                         |
| <i>x_height</i>          | Height of the button.                                                                                                                        |
| <i>w_displayList</i>     | Display list to attach to the button.                                                                                                        |
| <i>t_pressCallback</i>   | SKILL expression evaluated whenever the left mouse button is pressed.                                                                        |
| <i>x_backgroundColor</i> | Optional background color of the button.                                                                                                     |

### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dlMakeWidget**

```
dlMakeWidget (
 [x_backgroundColor]
 [l_point1]
 [l_point2]
)
=> x_widgetId / nil
```

#### **Description**

Creates a general purpose window (or widget). Uses of this widget can be for fixed menus or graphic display panels. Both points are specified in X Window System coordinates, where 0:0 is the upper-left corner of the screen.

#### **Arguments**

|                          |                                     |
|--------------------------|-------------------------------------|
| <i>x_backgroundColor</i> | Background color of the widget.     |
| <i>l_point1</i>          | Upper left position of the widget.  |
| <i>l_point2</i>          | Lower right position of the widget. |

#### ***Related Topics***

[View Display List Functions](#)

## **dlMapWidget**

```
dlMapWidget(
 x_widgetId
)
=> t / nil
```

### **Description**

Puts a widget back on the screen after a call to `dlUnMapWidget`. This function also applies to buttons.

### **Argument**

|                   |                |
|-------------------|----------------|
| <i>x_widgetId</i> | Widget to map. |
|-------------------|----------------|

### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dlMoveButton**

```
dlMoveButton(
 x_button
 x_xPosition
 x_yPosition
)
=> t | nil
```

#### **Description**

Moves a button to another location. Both the x and y position are specified in X Window System coordinates, where 0:0 is the upper-left corner of the screen.

#### **Arguments**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <i>x_button</i>    | Button being moved.                                    |
| <i>x_xPosition</i> | New x position of the upper-left corner of the button. |
| <i>x_yPosition</i> | New y position of the upper-left corner of the button. |

#### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

## dlQueryRasterFont

```
dlQueryRasterFont(
 t_fontName
)
=> l_fontProperties / nil
```

### Description

Returns the font property list for the raster font.

### Argument

*t\_fontName*

Name of a font, which can be:

- An X logical font descriptor (XLFD) such as "-\*-courier-medium-r-\*-\*-12-\*" and "-\*-helvetica-medium-r-\*-\*-14-\*".
- A standard alias name such as "6x13" or "fixed".
- A font type such as "text", "ciw", "label", "error", or "help". When a font type is specified, the default font name used in IC6.1.6 or previous releases is used as shown below:

| font type | font name                      |
|-----------|--------------------------------|
| -----     | -----                          |
| text      | -*-courier-medium-r-*-*-12-*   |
| ciw       | -*-courier-medium-r-*-*-12-*   |
| label     | -*-helvetica-medium-r-*-*-12-* |
| error     | -*-helvetica-medium-r-*-*-12-* |
| help      | -*-helvetica-medium-r-*-*-14-* |



## Cadence User Interface SKILL Reference

### Display Lists

---

#### Value Returned

|                         |                                                                                                                            |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>l_fontProperties</i> | A disembodied Property List (DPL) with same properties as that returned by the <a href="#">hiQueryFont</a> SKILL function. |
| <i>nil</i>              | Font cannot be found.                                                                                                      |

#### Examples

Retrieves the font property list of the XLFD "-\*-courier-medium-r-\*-\*-12-\*"

```
dlQueryRasterFont("-*-courier-medium-r-*-*-12-*")
(nil fontName "-*-courier-medium-r-*-*-12-*" fontAscent 10
fontDescent 3 fontHeight 13 fontWidth
7 minFontWidth 7 maxFontWidth 7)
```

Retrieves the font property list of the font type `label`.

```
dlQueryRasterFont("label")
(nil fontName "-*-helvetica-medium-r-*-*-12-*" fontAscent 11
fontDescent 3 fontHeight 14 fontWidth
7 minFontWidth 2 maxFontWidth 12)
```

#### ***Related Topics***

[View Display List Functions](#)

## **dlResizeButton**

```
dlResizeButton(
 x_button
 x_width
 x_height
)
=> t / nil
```

### **Description**

Resizes the specified button.

### **Arguments**

|                 |                            |
|-----------------|----------------------------|
| <i>x_button</i> | Button being moved.        |
| <i>x_width</i>  | Width of new button size.  |
| <i>x_height</i> | Height of new button size. |

### **Value Returned**

|     |                                  |
|-----|----------------------------------|
| t   | The specified button is resized. |
| nil | Operation failed.                |

### ***Related Topics***

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### dlSaveDlist

```
dlSaveDlist(
 w_displayList
 t_fileName
 t_dlistName
)
=> t / nil
```

#### Description

Saves the display list.

#### Arguments

|                      |                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_displayList</i> | Display list.                                                                                                                                                                     |
| <i>t_fileName</i>    | Name of file whose format is the SKILL functions necessary to regenerate the display list. When loaded, all the add functions will be called, and a display list will be created. |
| <i>t_dlistName</i>   | Name of the display list.                                                                                                                                                         |

#### Value Returned

|            |                         |
|------------|-------------------------|
| <i>t</i>   | Saves the display list. |
| <i>nil</i> | Operation failed.       |

#### Related Topics

[View Display List Functions](#)

## Cadence User Interface SKILL Reference

### Display Lists

---

#### **dISetDlistPosition**

```
dISetDlistPosition(
 w_displayList
 x_widgetId
 x_xOffset
 x_yOffset
)
=> t / nil
```

#### **Description**

Specifies the scale factor to map from display list coordinates to screen coordinates. The offsets only have effect if auto-fitting was turned off (via `dfIIFitDlistOnDraw`).

#### **Arguments**

|                      |                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>w_displayList</i> | Display list.                                                                                                     |
| <i>x_widgetId</i>    | ID the display list is attached to.                                                                               |
| <i>x_xOffset</i>     | Offset (in display list coordinates) of the display list in the x axis, assuming 0:0 to be the upper-left corner. |
| <i>x_yOffset</i>     | Offset (in display list coordinates) of the display list in the y axis, assuming 0:0 to be the upper-left corner. |

#### ***Related Topics***

[View Display List Functions](#)

## **dlSetDlistScale**

```
dlSetDlistScale(
 w_displayList
 x_widgetId
 f_scale
)
=> t / nil
```

### **Description**

Specifies the scale factor to map from display list coordinates to screen coordinates. This scale factor only has effect if auto-fitting was turned off (using `dfIIFitDlistOnDraw`).

### **Arguments**

|                      |                                     |
|----------------------|-------------------------------------|
| <i>w_displayList</i> | Display list.                       |
| <i>x_widgetId</i>    | ID the display list is attached to. |
| <i>f_scale</i>       | Scale factor.                       |

### ***Related Topics***

[View Display List Functions](#)

## **dlSetWidgetName**

```
dlSetWidgetName (
 x_widgetId
 t_name
)
=> t / nil
```

### **Description**

Sets the text displayed by the window manager banner.

### **Arguments**

|                   |                          |
|-------------------|--------------------------|
| <i>x_widgetId</i> | Widget to name.          |
| <i>t_name</i>     | Text to name the widget. |

### ***Related Topics***

[View Display List Functions](#)

## **dlUnMapWidget**

```
dlUnMapWidget(
 x_widgetId
)
=> t / nil
```

### **Description**

Removes the widget from the screen, but does not destroy it. This function also applies to buttons.

### **Argument**

|                   |                  |
|-------------------|------------------|
| <i>x_widgetId</i> | Widget to unmap. |
|-------------------|------------------|

### ***Related Topics***

[View Display List Functions](#)

## **Cadence User Interface SKILL Reference**

### Display Lists

---

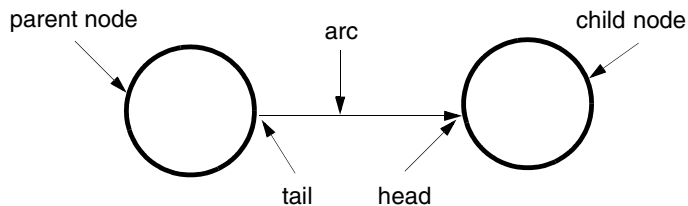


---

## Graph Browser

---

A Directed Acyclic Graph (*dag*) is a generic data structure that is commonly used in computer programs to represent objects and show how those objects relate to each other. The objects are represented in a directed graph by nodes. How the nodes relate to each other is represented by arcs that connect the nodes together. The graph is classified as *directed* because the arc connecting any two nodes together has a direction. The direction of the arc is from the parent node (at the tail of the arc) to the child node (at the head of the arc).



Although the name *dag* implies that the graph cannot have cycles, you can make cycles in the graphs you construct with the dag interface. When the dag functions traverse the graph you make, they take the necessary precautions to avoid infinitely looping because of cycles in the graph. However, the automatic placement or routing of graphs with cycles might not be visually pleasing.

The Virtuoso dag interface provides many sophisticated capabilities for constructing browser tools. These include

- The concept of classes of nodes, with inheritance of information from class to node.
- Context-sensitive pop-up menus and bindkeys, based on the class of the node over which the menu is invoked.
- Many capabilities for controlling the layout and appearance of the graph presentation.
- Automatic placement and routing algorithms for the nodes and arcs in the graph.
- Ability to display a text string and an icon for each node. The icon can be customized using the display list interface. The icon and text strings displayed can be node-specific or class-specific.

## Cadence User Interface SKILL Reference

### Graph Browser

---

- Ability to use any number of browser tools simultaneously.
- Ability to share nodes, arcs, and classes among many tools. The same node can have different appearances in different tools.
- Ability to use many of the normal *hi* interface functions for a window, including pulldown menus for a dag browser window.

### ***Related Topics***

Terms Used in Cadence User Interface

Properties of the dagArc

Properties of the dagClass

Properties of the dagNode

## Terms Used in Cadence User Interface

These key terms are used to define various features and properties of Cadence User Interface:

### ■ Coordinate Systems

There are two coordinate systems used in specifying dag object properties and dag function parameters:

- ❑ *screen coordinates*. These have their origin at the upper left and are specified in terms of pixels.
- ❑ *dag coordinates*. These have their origin at the lower left. You determine the magnitude of these coordinates by the height and width you assign to classes.

### ■ Context-Sensitive Pop-Up Menus and Actions

Defines context-sensitive pop-up menus and accelerators (bindkeys) for nodes or classes. The specifications for these are called action lists.

### ■ Display Lists, Icons, and SKILL Objects

Creates icon for display in the browser using the display list interface. To make those icons reflect information available only at run time, you can use the SKILL object feature of display lists. You can also interactively create display lists without SKILL objects using the Graphics Editor *Save Icon* command.

### ■ Intrinsic Property Inheritance

Browser nodes inherit a subset of the intrinsic properties of their class. There is a distinction between properties that you add to a browser object (user-defined) and those that are there when an object is created (intrinsic). Only intrinsic properties are inherited.

Inheritance means that, if a node does not have the desired property, the dag functions look for a property by the same name in the node's class. If it does not find the named property in either the node or class, it returns `nil`.

### ■ Kinds of Browser (dag) Objects

The object types are: class, node, arc, and tool. Each of these can have user-defined properties in addition to the intrinsic properties listed.

### ■ Window Types and Their Properties

A window type is a basic SKILL data type. Dag arc, class, node, and tool objects all are of this data type. In the type specifications for dag properties and dag function parameters and return values, a `w` prefix indicates that the object must be this data type.

## Cadence User Interface SKILL Reference

### Graph Browser

---

The straight arrow notation (→) is used to access and modify both the intrinsic and user-defined properties of browser objects. For example, to find out the name of a class object, you would enter

```
name = class->name
```

and, to change the name of a class, you would enter

```
class->name = "newName"
```

Unless otherwise specified all attributes of browser objects can be modified as shown above.

### ***Related Topics***

[Properties of the dagArc](#)

[Properties of the dagClass](#)

[Properties of the dagNode](#)

[Properties of the dagTool](#)

## Properties of the dagArc

The table shows the properties of dagArc. The text that follows explains each property in more detail.

| Name     | SKILL Type   | Description                                                                                                                                                                                                                                                            |
|----------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type     | symbol       | Type of the object. The symbol for this object is <i>dagArc</i> . This property is read-only.                                                                                                                                                                          |
| head     | wType (node) | dagNode that this arc is attached to at its head. This property is read-only. It can be <code>nil</code> , meaning there is no child node attached to the arc.                                                                                                         |
| tail     | wType (node) | dagNode that this arc is attached to at its tail. This property is read-only. It can be <code>nil</code> , meaning there is no parent node attached to the arc.                                                                                                        |
| arcPen   | fixNum       | Display list pen number that is used to render this arc when it is drawn within a browser tool.                                                                                                                                                                        |
| arcWidth | fixNum       | Width of the arc when it is drawn in a browser tool. If the <i>arcWidth</i> is zero, a thin line is drawn; if it is greater than zero, the arc is rendered as a path. The <i>arcWidth</i> dimension is unit-less and is scaled proportionally within the browser tool. |

### ***Related Topics***

[Properties of the dagClass](#)

[Properties of the dagNode](#)

[Properties of the dagTool](#)

## Properties of the dagClass

The table shows the properties of the dagClass. The text that follows explains each property in more detail.

A dagClass is a mechanism for grouping properties that are common to multiple nodes. You specify what the class is for a node when you create the node; each node is in only one class. Classes are never directly displayed. A class can only affect what is displayed by the properties inherited by nodes of the class.

| Name        | SKILL Type | Description                                                                                                                                                                                                                                                                 |
|-------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name        | string     | Name of the class.                                                                                                                                                                                                                                                          |
| type        | symbol     | Type of the object. The symbol for this object is dagClass. This property is read-only.<br><br>The action list, display list, width, and height properties described below are inheritable.                                                                                 |
| actionList  | list       | List of actions this class can do. actionLists are described in more detail later in this chapter. This property is read-only. The default value is nil.                                                                                                                    |
| displayList | wType      | Display list nodes of this class used when drawn in a browser tool.                                                                                                                                                                                                         |
| width       | fixNum     | Width of all the nodes of this class when displayed in a browser tool.                                                                                                                                                                                                      |
| height      | fixNum     | Property that (if true) causes the dag functions to not display, reserve space in automatic placement for or route in automatic routing to this node and any of its children. If property is <i>pruneChildren</i> , then this node will display, but its children will not. |

The width and height properties are expressed using cardinal numbers. The dimensions are unit-less and can be any range that you wish. The browser tool scales these dimensions, much like the Graphics Editor does when displaying a database.

### ***Related Topics***

[Properties of the dagArc](#)

[Properties of the dagNode](#)

## **Cadence User Interface SKILL Reference**

### **Graph Browser**

---

#### Properties of the dagTool

## Properties of the dagNode

The table shows the properties of the dagNode. The text that follows explains each property in more detail.

| Name            | SKILL Type | Description                                                                                                                                                                                                                                                                             |
|-----------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name            | string     | Name of the node. This name is automatically displayed to represent the node in the graph when <i>textOnly</i> is <code>t</code> for the tool. This property is read-only.                                                                                                              |
| textColor       | fixNum     | Color that the name of this node is drawn in when displayed in a dagTool. This property only takes effect when <i>textOnly</i> is <code>t</code> for the tool.                                                                                                                          |
| buttonColor     | fixNum     | Background color that this node will have when displayed in a dagTool when <i>textOnly</i> is <code>t</code> for the tool.                                                                                                                                                              |
| label           | string     | The value of this property for the node is shown when <i>showLabels</i> is <code>t</code> for the tool. The label is shown in addition to what would otherwise have been displayed for the node based on the value of <i>textOnly</i> for the tool. The default is the empty string "". |
| type            | symbol     | Type of this object. The symbol for this object is <i>dagNode</i> . This property is read-only.                                                                                                                                                                                         |
| actionList      | list       | List of actions that this node can do. These are only the actions that are specific to this node; see <i>expandedActions</i> below. Action lists are described in more detail in the action list section. This property is read-only. The default is <code>nil</code> .                 |
| expandedActions | list       | List of all the actions that a node can do. This list is the concatenation of the actions specified on the class and those specified on the node. This property is read-only. The default is <code>nil</code> .                                                                         |
| displayList     | wType      | Display list that this node uses when drawn in a browser tool. It must either be defined for the node or inherited from its class. When <i>textOnly</i> is <code>t</code> for the tool, the contents of the display list property will be ignored.                                      |



## Cadence User Interface SKILL Reference

### Graph Browser

| Name                       | SKILL Type                                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>class</code>         | <code>wType</code>                        | Class of this node as returned by <i>dagCreateClass</i> . This property is read-only.<br><br>The <i>dagNode</i> properties <i>parents</i> , <i>parentObjects</i> , <i>children</i> , and <i>childObjects</i> are created whenever a linking function is called to link two nodes together. They are modified whenever an unlinking function is called to break two nodes apart. These properties can be modified only by calling the linking and unlinking functions; they are read-only properties of the node. |
| <code>parentObjects</code> | <code>list</code>                         | List of nodes that this node attaches as parents.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>parents</code>       | <code>list</code>                         | List of <i>dagArcs</i> that this node attaches as parents.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>childObjects</code>  | <code>list</code>                         | List of nodes that this node attaches as children.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>children</code>      | <code>list</code>                         | List of <i>dagArcs</i> that this node attaches as children.                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>x</code>             | <code>fixNum</code>                       | X position in a <i>dagTool</i> at which this node resides. Node positions are specified relative to the lower left corner of the window and of the node bounding box.                                                                                                                                                                                                                                                                                                                                            |
| <code>y</code>             | <code>fixNum</code>                       | Y position in a <i>dagTool</i> at which this node resides. Node positions are specified relative to the lower left corner of the window and of the node bounding box.                                                                                                                                                                                                                                                                                                                                            |
| <code>invisible</code>     | <code>[t or nil]</code>                   | Property that causes the dag functions not to display the node and automatic routing not to route to it. However, automatic placement will reserve room for it in the placed graph.                                                                                                                                                                                                                                                                                                                              |
| <code>pruned</code>        | <code>t, nil, or<br/>pruneChildren</code> | Node and its children should not be processed or drawn by the dag functions.                                                                                                                                                                                                                                                                                                                                                                                                                                     |

### ***Related Topics***

[Properties of the dagArc](#)

[Properties of the dagClass](#)

## **Cadence User Interface SKILL Reference**

### **Graph Browser**

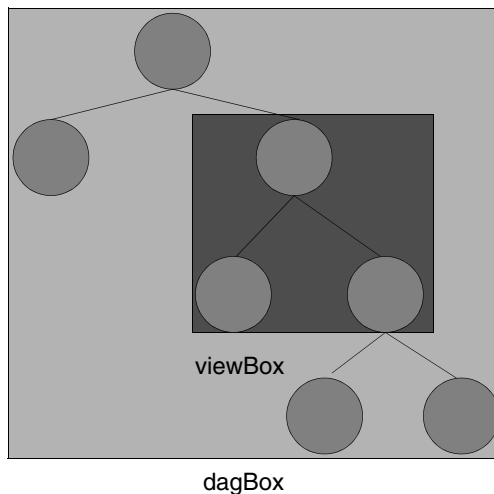
---

#### Properties of the dagTool

## Properties of the dagTool

The table shows the properties of the dagTool. A dagTool associates a window with a graph to display in that window. The dagTool is an object just like nodes and arcs and has properties that you can reference and assign.

The view properties specify the box that the *dagTool* displays.



These properties govern the display of the graph in this window.

| Name       | SKILL Type | Description                                                                                                                                                               |
|------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name       | string     | Name of the tool. This is displayed in the window banner of the tool window.                                                                                              |
| type       | symbol     | Type of the object. The symbol for this object is <i>dagClass</i> . This property is read-only.                                                                           |
| window     | wType      | Window of the dagTool. It's initialized when the <i>dagOpenTool</i> function is called. This property is read-only.                                                       |
| toolNumber | fixNum     | Number of this tool. Each tool is assigned a unique number upon creation. This property is read-only.                                                                     |
| horizontal | t or nil   | Controls the dagTool to draw the nodes from left to right, rather than from top to bottom. The default is <i>nil</i> , meaning draw the graph from the top to the bottom. |

## Cadence User Interface SKILL Reference

### Graph Browser

| Name                         | SKILL Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>scaleToFit</code>      | <code>t or nil</code> | Controls the <i>dagTool</i> to draw the graph so that bounding boxes of all of its nodes fit within the given window. Icons and scalable font text strings are scaled proportionally; raster text strings are not. If <code>nil</code> , the <i>dagTool</i> uses the view box (specified by <i>viewLx</i> , <i>viewLy</i> , <i>viewUx</i> , and <i>viewUy</i> ) and draws only the visible, unpruned nodes that exist within this box. The default is <code>t</code> . Using this property causes the <i>anchorTo</i> and <i>anchorObject</i> properties to be ignored. |
| <code>noPlace</code>         | <code>t or nil</code> | Restricts the <i>dagTool</i> to invoke the placer. If you set this to <code>t</code> , then you are responsible for positioning the nodes yourself. The default is <code>nil</code> .                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>noRoute</code>         | <code>t or nil</code> | Restricts the <i>dagTool</i> to invoke the router. The default is <code>nil</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>preDisplayProc</code>  | <code>string</code>   | <p>The pre and post display procedures can be used to do work just before or just after the graph is displayed.</p> <p>The <i>preDisplayProc</i> is called with the following form:</p> <pre>preDisplayProc( tool dirtyOrNot )</pre> <p>The <i>dirtyOrNot</i> parameter is true if any nodes in the graph have been modified (either by having new children created, or by having a property modified).</p> <p>If you have disabled automatic placement, you can do the placement you want in this function.</p>                                                        |
| <code>postDisplayProc</code> | <code>string</code>   | <p>The <i>postDisplayProc</i> is called with the following form:</p> <pre>postDisplayProc( tool )</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Cadence User Interface SKILL Reference

### Graph Browser

| Name                        | SKILL Type | Description                                                                                                                                                                                                                                                                                                                        |
|-----------------------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>closeProc</code>      | string     | SKILL procedure that is called when the <i>dagTool</i> is closed. The <i>closeProc</i> takes the following arguments.<br><br><code>closeProc( w_dagTool )</code>                                                                                                                                                                   |
| <code>startingObject</code> | wType      | <i>dagNode</i> that the tool starts with. This node is the root of the graph you want displayed in the tool. This property is read-only.                                                                                                                                                                                           |
| <code>anchorObject</code>   | wType      | Node the tool justifies to. The <i>dagTool</i> positions its viewing box about this node.                                                                                                                                                                                                                                          |
| <code>anchorTo</code>       | symbol     | Justification style for positioning the graph around the <i>anchorObject</i> . The graph is justified around the corresponding part of the <i>anchorObject</i> node. It is ignored when the <i>scaleToFit</i> property is <code>t</code> . It can be one of the symbols name on the left in the following table or <i>'none'</i> . |
| <code>x</code>              | fixNum     | X position of the tool (in screen coordinates). This property is read-only.                                                                                                                                                                                                                                                        |
| <code>y</code>              | fixNum     | Y position of the tool (in screen coordinates). This property is read-only.                                                                                                                                                                                                                                                        |
| <code>width</code>          | fixNum     | Width of the tool (in screen dimensions). This property is read-only.                                                                                                                                                                                                                                                              |
| <code>height</code>         | fixNum     | Height of this tool (in screen dimensions). This property is read-only.                                                                                                                                                                                                                                                            |
| <code>dagLx</code>          | fixNum     | Least x coordinate specified in the graph. This property is read-only.                                                                                                                                                                                                                                                             |
| <code>dagLy</code>          | fixNum     | Least y coordinate specified in the graph. This property is read-only.                                                                                                                                                                                                                                                             |
| <code>dagUx</code>          | fixNum     | Greatest x coordinate specified in the graph. This property is read-only.                                                                                                                                                                                                                                                          |
| <code>dagUy</code>          | fixNum     | Greatest y coordinate specified in the graph. This property is read-only.                                                                                                                                                                                                                                                          |
| <code>viewLx</code>         | fixNum     | Lower left x coordinate that is displayed                                                                                                                                                                                                                                                                                          |
| <code>viewLy</code>         | fixNum     | Lower left y coordinate that is displayed.                                                                                                                                                                                                                                                                                         |

## Cadence User Interface SKILL Reference

### Graph Browser

| Name               | SKILL Type | Description                                                                                                                                                                                      |
|--------------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| viewUx             | fixNum     | Upper right x coordinate.                                                                                                                                                                        |
| viewUy             | fixNum     | Upper right y coordinate.                                                                                                                                                                        |
| textOnly           | t or nil   | Tells the <i>dagTool</i> to draw the nodes as text boxes, rather than using displayLists. The default is <i>nil</i> . The text is displayed in a raster font.                                    |
| showLabels         | t or nil   | Turns on label display mode in the tool. The <i>dagTool</i> displays the <i>label</i> property of the <i>dagNode</i> .                                                                           |
| labelJustification | symbol     | Specifies where the label is displayed relative to the <i>dagNode</i> when the <i>showLabels</i> property is <i>t</i> . It follows the same syntax and behavior as the <i>anchorTo</i> property. |
| fontName           | string     | Specifies which raster font is used to display the labels when the <i>showLabels</i> property is <i>t</i> .                                                                                      |

### ***Related Topics***

[Properties of the dagArc](#)

[Properties of the dagClass](#)

[Properties of the dagNode](#)

## Actions and Action Lists

The ability to construct and display a directed graph is sufficient for many applications. In addition, however, the dag interface allows you to put actions on a class or node. Actions are SKILL procedures this node or nodes of this class can invoke using pop-up menus. When a node is displayed in the browser tool and has actions associated with it, you can press the right mouse button over the node and a menu of actions is displayed.

Actions are assigned to a node or class using a list. This list is of the form:

```
(t_actionName t_actionProcedure t_actionArgs [t_actionAccelerator]
 [g_enabled])
```

|                          |                                                                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_actionName</i>      | The name of the action as it should appear in the popup menu.                                                                                                                                                                                        |
| <i>t_actionProcedure</i> | The name of the SKILL procedure called when the named action is selected.                                                                                                                                                                            |
| <i>t_actionArgs</i>      | String specifying all the arguments passed to the actionProcedure or the empty string ("" ) if there are none. This string and the action procedure name are combined with sprintf to construct the expression evaluated when the action is invoked. |

## Cadence User Interface SKILL Reference

### Graph Browser

---

#### *t\_actionAccelerator*

Assigns accelerator string by specifying key or mouse button to trigger the specified action. Valid values are as follows:

- any single alphanumeric (a-z, A-Z, 0-9) character: Opens by using the equivalent key from the keyboard.
- "mouseLeft": Opens by a single click of the left mouse button
- "mouseMiddle": Opens by a single click of the middle mouse button)
- "mouseRight": Opens by a single click of the middle mouse button)
- "mouseLeft2": Opens by a double click of the left mouse button)
- "mouseMiddle2": Opens by a double click of the middle mouse button)
- "mouseRight2": Opens by a double click of the middle mouse button)

If both "mouseMiddle" and "mouseRight" are defined, then the "mouseRight" binding is ignored. Similarly, if both "mouseMiddle2" and "mouseRight2" are defined, then the "mouseRight2" binding is ignored.

#### *g\_enabled*

Specifies whether the action is enabled (*t*) or disabled (*nil*). Disabled actions are displayed in gray in a popup menu and their action accelerators are not enabled.

## Examples

Prints `hello` using the string `"println"`:

```
"Hello" "println" "\"hello\"" " " t)
=> println("hello")
```

Adds the two number 1 3 using the string `"addTwoNumbers"`:

```
("Add" "addTwoNumbers" "1 3" "a" t)
=> addTwoNumbers(1 3)
```



## Cadence User Interface SKILL Reference

### Graph Browser

---

Pressing the `a` key while the cursor is positioned over a `dagNode` with this accelerator also triggers this action.

One cautionary note about the action menus is that the popup menu is created the first time the user presses the right mouse button over the node. No changes to the *actionList* will be reflected after the popup is created.

The context-sensitive bindkeys (action accelerators) supersede the ordinary *hi* interface bindkey functionality.

### ***Related Topics***

[Properties of the `dagArc`](#)

[Properties of the `dagClass`](#)

[Properties of the `dagNode`](#)

[Properties of the `dagTool`](#)

## dag Functions

This topic provides a list of Cadence® SKILL functions associated with Directed Acyclic Graph (*dag*) interface.

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [dagAddActionToObject](#)
- [dagCreateClass](#)
- [dagCreateNode](#)
- [dagDeleteActionFromObject](#)
- [dagDestroyNode](#)
- [dagDisplayTool](#)
- [dagGetCurrentObject](#)
- [dagGetCurrentTool](#)
- [dagLinkParentToChild](#)
- [dagNumToTool](#)
- [dagOpenTool](#)
- [dagPopTool](#)
- [dagPushTool](#)
- [dagRefreshObject](#)
- [dagSetActionStatus](#)
- [dagSetCurrentTool](#)
- [dagSetExpandedActionStatus](#)
- [dagSetFont](#)
- [dagUnlinkParentFromChild](#)

## **dagAddActionToObject**

```
dagAddActionToObject(
 l_action
 w_object
)
=> t_actionName / nil
```

### **Description**

Adds the action to the specified dag node or class. However, if the action is already in the list, it is not replaced. For an existing action, use `dagDeleteActionFromObject` to delete the action and then call `dagAddActionToObject`.

Nodes inherit the actions of their class.

### **Arguments**

|                 |                                         |
|-----------------|-----------------------------------------|
| <i>l_action</i> | List specifying the action.             |
| <i>w_object</i> | Node or class the action is applied to. |

### **Value Returned**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <i>t_actionName</i> | The name of the action (car of the action list). |
| <i>nil</i>          | The function was not successful.                 |

### **Examples**

Adds an action to the myClass *dagClass*:

```
action = ' ("Hello" "println" "\"hello\"")
dagAddActionToObject(action myClass)
```

### ***Related Topics***

[dag Functions](#)

[Actions and Action Lists](#)

## **dagCreateClass**

```
dagCreateClass(
 t_className
)
=> w_class / nil
```

### **Description**

Creates a *dagClass* and returns it. If a class by this name already exists, it is overwritten.

### **Argument**

|                    |                                                                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>t_className</i> | Name of the class. Make the name of each class unique among the names of classes used in any dag browser tool. This is probably best done by adding a unique prefix to the class name. |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### **Value Returned**

|                |                     |
|----------------|---------------------|
| <i>w_class</i> | The new class.      |
| <i>nil</i>     | There was an error. |

### ***Related Topics***

[dag Functions](#)

## **dagCreateNode**

```
dagCreateNode (
 t_nodeName
 w_class
)
=> w_node / nil
```

### **Description**

Creates a dagNode and returns it.

### **Arguments**

|                   |                                                                                |
|-------------------|--------------------------------------------------------------------------------|
| <i>t_nodeName</i> | Name of the node. The name of each node must be unique among all its siblings. |
| <i>w_class</i>    | Class of this node as returned by <a href="#"><u>dagCreateClass</u></a> .      |

### **Value Returned**

|               |                     |
|---------------|---------------------|
| <i>w_node</i> | The new node.       |
| <i>nil</i>    | There was an error. |

### ***Related Topics***

[dag Functions](#)

## **dagDeleteActionFromObject**

```
dagDeleteActionFromObject(
 t_actionName
 w_object
)
=> t / nil
```

### **Description**

Deletes the action named *t\_actionName* from the specified class or node. If a node action was inherited from its class, the action must be deleted from the class, not the node.

### **Arguments**

|                     |                                         |
|---------------------|-----------------------------------------|
| <i>t_actionName</i> | Name of the action.                     |
| <i>w_object</i>     | Class or node the action is applied to. |

### **Value Returned**

|            |                             |
|------------|-----------------------------|
| <i>t</i>   | The action was deleted.     |
| <i>nil</i> | The action was not deleted. |

### ***Related Topics***

[dag Functions](#)

## **dagDestroyNode**

```
dagDestroyNode(
 w_node
)
=> t / nil
```

### **Description**

Destroys a *dagNode*. All descendant nodes and arcs of this node are also destroyed, as well as the arcs linking this node to its parents.

### **Argument**

|               |                       |
|---------------|-----------------------|
| <i>w_node</i> | Node to be destroyed. |
|---------------|-----------------------|

If you do not want the descendants of this node to be destroyed, unlink the node immediate children with [dagUnlinkParentFromChild](#) before calling this function.

### **Values Returned**

|     |                         |
|-----|-------------------------|
| t   | The node was destroyed. |
| nil | There was an error.     |

### ***Related Topics***

[dag Functions](#)

## dagDisplayTool

```
dagDisplayTool(
 [w_dagTool]
 [g_runPlacer]
)
=> t / nil
```

### Description

Redisplays a *dagTool*. You can use this function if you want to re-display all of the objects in the graph rather than calling `dagRefreshObject` for each one.

In general, this routine need not be called. The dag will automatically refresh all tools in which it detects changes. If you wish to force the redisplay at a specific point in a procedure, you can call this routine.

See [dagRefreshObject](#) for the circumstances in which the dag does not automatically redisplay.

### Arguments

|                    |                                                                                                                                                                                  |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>w_dagTool</i>   | <i>dagTool</i> to be displayed. If no tool is specified, the current tool is displayed.                                                                                          |
| <i>g_runPlacer</i> | Informs the <i>dagTool</i> whether to re-run the placement algorithm for the graph or just redisplay the nodes where they are currently located. The default is <code>t</code> . |

### Values Returned

|                  |                                     |
|------------------|-------------------------------------|
| <code>t</code>   | The <i>dagTool</i> was redisplayed. |
| <code>nil</code> | There was an error.                 |

### Related Topics

[dag Functions](#)



## **dagGetCurrentObject**

```
dagGetCurrentObject()
=> w_object / nil
```

### **Description**

Identifies the node for which the action was selected. It takes no arguments and returns the current node, or `nil`.

This function is intended for use in the action functions and in display list SKILL objects. The current object is set by the dag routines under the following conditions:

- When each dagnode is drawn, that node is made current.
- When the user invokes an action routine of a node, that node is made current.
- When the currently active node is deleted, in which case `dagGetCurrentObject` will return `nil`.

`dagGetCurrentObject` returns the object where the pointer is located at the time the function executes. If you start a long procedure and then move the pointer to another object before `dagGetCurrentObject` executes, `dagGetCurrentObject` returns the new object, not the original one.

If you call other procedures from your action callback, you should call `dagGetCurrentObject` in the first procedure only, and then pass a variable to the subprocedures. Do not call `dagGetCurrentObject` multiple times in a callback tree.

### **Arguments**

None

### **Value Returned**

|                       |                           |
|-----------------------|---------------------------|
| <code>w_object</code> | The current node.         |
| <code>nil</code>      | There is no current node. |

### **Examples**

```
procedure(myDagCallback()
 prog((curObject)
 curObject = dagGetCurrentObject()
 subProc(curObject)
```

## Cadence User Interface SKILL Reference

### Graph Browser

---

```
)
)
 procedure(subProc(curObject)
 ; whatever
)
```

### ***Related Topics***

[dag Functions](#)

## **dagGetCurrentTool**

```
dagGetCurrentTool()
=> w_dagTool / nil
```

### **Description**

Returns the current *dagTool*, and `nil` if there is not a current *dagTool*. The current *dagTool* is different from the current window. The current tool might not be the current window.

This function is intended for use in the action functions and in display list SKILL objects. Immediately before any action callback procedure is called, the current tool value is updated.

Also, before a tool is redrawn, the current tool is updated. If the current tool is closed (deleted), `dagGetCurrentTool` will return `nil`.

### **Argument**

None

### **Value Returned**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>w_dagTool</code> | The current <i>dagTool</i> .         |
| <code>nil</code>       | There is no current <i>dagTool</i> . |

### ***Related Topics***

[dag Functions](#)

## **dagLinkParentToChild**

```
dagLinkParentToChild(
 w_parent
 w_child
)
=> w_arc / nil
```

### **Description**

Creates a *dagArc* that links the parent node to the child node. It returns the arc it creates. If an error occurs, it returns `nil`.

The automatic placement algorithm places the children of a parent node in the order in which they were added with this function.

### **Arguments**

|                 |                                                           |
|-----------------|-----------------------------------------------------------|
| <i>w_parent</i> | Specifies the parent node, head of the arc to be created. |
| <i>w_child</i>  | Specifies the child node, tail of the arc to be created.  |

### **Value Returned**

|                  |                     |
|------------------|---------------------|
| <i>w_arc</i>     | The new arc.        |
| <code>nil</code> | There was an error. |

### ***Related Topics***

[dag Functions](#)

## **dagNumToTool**

```
dagNumToTool(
 x_toolNumber
)
=> w_dagTool / nil
```

### **Description**

Returns the *dagtool* that matches the window number specified by *x\_toolNumber*.

### **Argument**

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <i>x_toolNumber</i> | The <i>dagTool</i> number for which you are trying to get the associated <i>dagtool</i> . |
|---------------------|-------------------------------------------------------------------------------------------|

### **Value Returned**

|                  |                            |
|------------------|----------------------------|
| <i>w_dagTool</i> | The dagTool.               |
| nil              | The dagTool was not found. |

### ***Related Topics***

[dag Functions](#)

## dagOpenTool

```
dagOpenTool(
 x_backGroundColor
 l_bBox
 w_startingObject
 t_name
 t_help
)
=> w_dagTool / nil
```

### Description

Creates a *dagTool* object and opens a window in which to display the graph starting from the specified starting node.

### Arguments

|                          |                                                                                                                                                                                                                     |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x_backGroundColor</i> | The background color of the tool. Specifying <code>-1</code> for this argument is the same as passing <code>hivEditorBackgroundColor</code> , which means the window is allowed to use the editor background color. |
| <i>l_bBox</i>            | List specifying the lower-left and upper-right coordinates of the tool (in screen coordinates). If this list is <code>nil</code> , you are prompted to position the window on the screen.                           |
| <i>w_startingObject</i>  | The node that the tool uses to start processing from. This property can be changed after the tool is opened.                                                                                                        |
| <i>t_name</i>            | Name displayed in the dag browser tool window banner.                                                                                                                                                               |
| <i>t_help</i>            | A string giving the name of a symbol for the help message for this tool. It determines what is shown when you press the Help button in the dag browser tool's banner line.                                          |

## Cadence User Interface SKILL Reference

### Graph Browser

---

#### Value Returned

|                        |                                  |
|------------------------|----------------------------------|
| <code>w_dagTool</code> | The dagTool that was opened.     |
| <code>nil</code>       | The dagTool could not be opened. |

#### Examples

```
tool = dagOpenTool(hiMatchColorByName("white"
list(100:100 400:400)
myNode "myTool" "xxxHelpSymbol")
```

#### Additional Information

To close a dagTool, call `hiCloseWindow`.

```
hiCloseWindow(tool->window)
```

If you have specified a *closeProc* property on the tool, that procedure is called just before the tool or the window for it is destroyed. The user may close a dag tool by using a window manager *Quit* or *Close* button, which will also cause the *closeProc* to be called.

In *closeProc* you should be sure to destroy all the nodes and arcs you created for this graph that you no longer need. For example, if the nodes in this browser tool are used only in this browser and you have no nodes pushed on the starting object stack, you could call

```
dagDestroyNode(tool->startingObject)
```

#### Related Topics

[dag Functions](#)

## **dagPopTool**

```
dagPopTool(
 w_dagTool
)
=> t / nil
```

### **Description**

Pops the tool stack specified by the *w\_dagTool* argument.

The *dagTool* can have a stack of starting nodes. You can push new starting nodes onto the stack and pop them off. This is similar to the Layout Editor Push and Return commands. There is no limit to the number of nodes you can have on the stack. Each time that you push or pop the stack, the tool is redisplayed, so there is no need to call `dagDisplayTool`.

### **Argument**

|                  |                                                                  |
|------------------|------------------------------------------------------------------|
| <i>w_dagTool</i> | The tool for which the starting node stack is being manipulated. |
|------------------|------------------------------------------------------------------|

### **Value Returned**

|     |                              |
|-----|------------------------------|
| t   | The function was successful. |
| nil | There was an error.          |

### ***Related Topics***

[dag Functions](#)



## dagPushTool

```
dagPushTool(
 w_dagTool
 w_newStartingNode
)
=> t / nil
```

### Description

Pushes a new starting object specified by the *w\_newStartingNode* argument onto the stack of the tool specified by *w\_dagTool*.

The *dagTool* can have a stack of starting nodes. You can push new starting nodes onto the stack and pop them off. This is similar to the Layout Editor Push and Return commands. There is no limit to the number of nodes you can have on the stack.

### Arguments

|                          |                                                                    |
|--------------------------|--------------------------------------------------------------------|
| <i>w_dagTool</i>         | The tool for which the starting object stack is being manipulated. |
| <i>w_newStartingNode</i> | The new starting node to put on the <i>dagTool</i> 's stack.       |

### Value Returned

|     |                              |
|-----|------------------------------|
| t   | The function was successful. |
| nil | There was an error.          |

### Related Topics

[dag Functions](#)

## **dagRefreshObject**

```
dagRefreshObject(
 w_dagNode
)
=> t / nil
```

### **Description**

Redraws the node specified by *w\_dagNode* in all the tools it is currently displayed.

*dagRefreshObject* should be called to ensure that the display is up to date if

- The node's display list has been changed (either by changing the node's display list directly or by changing the display list inherited from the class)
- The display list for the node contains a SKILL object which needs to be reevaluated
- The width, height, or display list properties inherited from the class have changed

### **Argument**

|                  |                           |
|------------------|---------------------------|
| <i>w_dagNode</i> | The node to be refreshed. |
|------------------|---------------------------|

### **Value Returned**

|            |                              |
|------------|------------------------------|
| <i>t</i>   | The function was successful. |
| <i>nil</i> | There was an error.          |

### ***Related Topics***

[dag Functions](#)

## **dagSetActionStatus**

```
dagSetActionStatus(
 t_actionName
 w_dagObject
 g_status
)
=> t / nil
```

### **Description**

Sets the status of the specified action in the action list of the class or node. An action that has been turned off is grayed-out in the pop-up menu and not available through its action accelerator, if any.

### **Arguments**

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| <i>t_actionName</i> | Name of the action.                                 |
| <i>w_dagObject</i>  | Node or class the action is associated with.        |
| <i>g_status</i>     | [t or nil] t turns on the action; nil turns it off. |

### **Values Returned**

|     |                              |
|-----|------------------------------|
| t   | The function was successful. |
| nil | There was an error.          |

### ***Related Topics***

[dag Functions](#)

## **dagSetCurrentTool**

```
dagSetCurrentTool(
 w_dagTool
)
=> t / nil
```

### **Description**

Sets the current *dagTool* to the one specified by the *w\_dagTool* argument.

### **Argument**

|                  |                           |
|------------------|---------------------------|
| <i>w_dagTool</i> | The tool to make current. |
|------------------|---------------------------|

### **Value Returned**

|     |                              |
|-----|------------------------------|
| t   | The function was successful. |
| nil | There was an error.          |

### ***Related Topics***

[dag Functions](#)

## **dagSetExpandedActionStatus**

```
dagSetExpandedActionStatus(
 t_actionName
 w_dagObject
 g_status
)
=> t / nil
```

### **Description**

Sets the status of the specified action in the expanded-action list of a node. Setting the status to `nil` makes the action name appear gray in the pop-up menu and makes the action accelerator unavailable. Once a *dagNode* has been placed into a tool, the actions from the *dagClass* are concatenated to make a full list of actions for this node. This is known as the expanded-action list.

### **Arguments**

|                     |                                     |
|---------------------|-------------------------------------|
| <i>t_actionName</i> | Name of the action.                 |
| <i>w_dagObject</i>  | Node the action is associated with. |
| <i>g_status</i>     | Enables the action.                 |

### **Value Returned**

|                  |                              |
|------------------|------------------------------|
| <code>t</code>   | The function was successful. |
| <code>nil</code> | There was an error.          |

### ***Related Topics***

[dag Functions](#)

## **dagSetFont**

```
dagSetFont(
 t_XFontName
)
=> t / nil
```

### **Description**

Modifies the font used in text mode in any *dag* tool.

### **Argument**

|                    |                                             |
|--------------------|---------------------------------------------|
| <i>t_XFontName</i> | Name of the X font to be used in text mode. |
|--------------------|---------------------------------------------|

### **Values Returned**

|            |                                 |
|------------|---------------------------------|
| <i>t</i>   | Specified font is found.        |
| <i>nil</i> | Specified font cannot be found. |

### ***Related Topics***

[dag Functions](#)

## dagUnlinkParentFromChild

```
dagUnlinkParentFromChild(
 w_arc
)
=> t / nil
```

### Description

Unlinks a parent node from a child node. It takes, as its only argument, the *dagArc* that connects the parent to the child. The arc is destroyed; however, neither the parent node nor the child node is destroyed. This can be useful if you want to change the parent of a node.

### Argument

|              |                                          |
|--------------|------------------------------------------|
| <i>w_arc</i> | The arc connecting the parent and child. |
|--------------|------------------------------------------|

### Values Returned

|            |                          |
|------------|--------------------------|
| <i>t</i>   | The nodes were unlinked. |
| <i>nil</i> | There was an error.      |

### Examples

Use this example to visualize a graph by using DAG functions.

```
myClass= dagCreateClass("myClass") ; create class
myClass->displayList = dlMakeDisplayList(2)
dlAddBox(myClass->displayList 12 10:10 100:100)
dlAddBox(myClass->displayList 14 30:30 80:80)
myClass->pen = 1

person1= dagCreateNode("Person1" myClass) ; create nodes
person2= dagCreateNode("Person2" myClass)
person3= dagCreateNode("Person3" myClass)

person1->textColor = hivForegroundColor ; avoid erase in rubberbanding
person2->textColor = hivForegroundColor
person3->textColor = hivForegroundColor
```

## Cadence User Interface SKILL Reference

### Graph Browser

---

```
person1->label = "p1label"
person2->label = list("p1l1" "p1l2")

dagLinkParentToChild(person1 person2) ; link nodes together
dagLinkParentToChild(person1 person3)

procedure(main()
 tool = dagOpenTool(hivEditorBackgroundColor
 list(180:150 450:550) ; x1,y1 x2,y2
 person1 ; starting node
 "DAG example" ; win title
 "dagHelpSymbol") ; help symbol

 tool->textOnly = nil ; use text mode only; display list core dumps
 tool->showLabels = t
 tool->labelJustification = 'lowerLeft

 dagAddActionToObject(; add accelerator a on nodes
 ("Print_Hello" "println" "\"Hi, I am in func hello\"" "a" t)
 myClass
)
 dagAddActionToObject(; add accelerator b on nodes
 ("Print_object_name" "printObj" "dagGetCurrentObject()" "b" t)
 myClass
)
 dagAddActionToObject(; add accelerator c on nodes
 ("Print_M1" "println" "\"message 1\"" "c" t)
 myClass
)
 dagAddActionToObject(; add accelerator d on nodes
 ("Print_M2" "println" "\"message 2\"" "d" t)
 myClass
)
 dagAddActionToObject(; add accelerator e on nodes
 ("Print_M3" "println" "\"message 3\"" "e" t)
 myClass
)
 dagAddActionToObject(; add accelerator f on nodes
 ("Print_M4" "println" "\"message 4\"" "f" t)
 myClass
)
)
```



## Cadence User Interface SKILL Reference

### Graph Browser

---

```
dagAddActionToObject(; add accelerator g on nodes
 ("Print_M5" "println" "\"message 5\" \"g\" t)
 myClass
)
dagAddActionToObject(; add accelerator MMB on nodes
 ("Print_MMB" "println" "\"click MMB\" \"mouseMiddle\" t)
 myClass
)
dagAddActionToObject(; add accelerator MMB2 on nodes
 ("Print_MMB2" "println" "\"double click MMB\" \"mouseMiddle2\" t)
 myClass
)
dagAddActionToObject(; add accelerator RMB on nodes
 ("Print_RMB" "println" "\"click RMB\" \"mouseRight\" t)
 myClass
)
dagAddActionToObject(; add accelerator RMB2 on nodes
 ("Print_RMB2" "println" "\"double click RMB\" \"mouseRight2\" t)
 myClass
)
)

procedure(printHello(buf) ; called from accelerator a on nodes
 printf("got message -%s-\n", buf)
)
procedure(printObj(node) ; called from accelerator b on nodes
 printf("got node -%s-\n", node->name)
)

prefix= "DAG example"
hiRegisterBindKeyPrefix(prefix "browser") ; register bind key table

hiSetBindKey(prefix "None<Btn1Down>" "startBox()")
hiSetBindKey(prefix "None<Btn1Down> EF" "getBoxPoint()")
hiSetBindKey(prefix "None<Btn2Down>" "keyFunc2()") ; do bind keys
hiSetBindKey(prefix "None<Btn3Down>" "keyFunc3()")

; called when left mouse button is pressed inside the enter function
procedure(getBoxPoint()
 addPoint(hiGetCommandPoint())
)
```

## Cadence User Interface SKILL Reference

### Graph Browser

---

```
procedure(boxDone(w done pts) ; called when enter function for box is done
 if(done then
 printf("box entered was %L \n" pts)
 else
 println("Box entry aborted\n")
)
)
; called when left mouse button is pressed and not in enter function
procedure(startBox()
 enterBox(?doneProc "boxDone"
 ?prompts '("enter 1st point for box" "enter 2nd point for box")
)
)

procedure(keyFunc2() ; called when middle mouse button is pressed
 println("got middle mouse button")
)
procedure(keyFunc3() ; called when right mouse button is pressed
 println("got right mouse button")
)
main()
```

### ***Related Topics***

[dag Functions](#)

---

## Interprocess Communication

---

`hi` interprocess communication functions have been replaced by `ipc` functions and are no longer available, except for `hiSleep`, which is an alias for `ipcSleep`.

For information about `ipc` functions, see [Cadence Interprocess Communication SKILL Reference](#).

### **hiSleep**

The `hiSleep()` function is an alias for the [ipcSleep](#) function.

## **Cadence User Interface SKILL Reference**

### Interprocess Communication

---

---

## Stroke Functions

---

This topic provides a list of Cadence® SKILL functions associated with strokes.

A stroke is a unique shape that users draw in a window by pressing and holding a mouse button, typically the right button. Each stroke opens a command. Files of predefined strokes are shipped with the Cadence software and must be installed.

Strokes are most commonly used by the Virtuoso layout editor and the Preview floorplanning tool. For more information, see [Layout Editor Basics](#).

Only the functions listed here are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- [hiGetStrokeBBox](#)
- [hiGetStrokeFirstPt](#)
- [hiGetStrokeLastPt](#)
- [hiStroke](#)

## Cadence User Interface SKILL Reference

### Stroke Functions

---

#### hiGetStrokeBBox

```
hiGetStrokeBBox()
=> l_boundingBox / nil
```

#### Description

Returns the bounding box of the stroke when within a callback initiated by a stroke.

#### Arguments

None

#### Values Returned

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <i>l_boundingBox</i> | Bounding box of the stroke in user units.                                              |
| <i>nil</i>           | There is no current window or no valid application associated with the current window. |

#### ***Related Topics***

[Stroke Functions](#)

[hiGetStrokeFirstPt](#)

[hiGetStrokeLastPt](#)

[hiStroke](#)

## hiGetStrokeFirstPt

```
hiGetStrokeFirstPt ()
=> l_point / nil
```

### Description

Returns the point location of the first pixel of a stroke when within a callback initiated by a stroke.

### Arguments

None

### Values Returned

|                |                                                                                        |
|----------------|----------------------------------------------------------------------------------------|
| <i>l_point</i> | First pixel of the stroke, expressed in user units.                                    |
| <i>nil</i>     | There is no current window or no valid application associated with the current window. |

### ***Related Topics***

[Stroke Functions](#)

[hiGetStrokeBBox](#)

[hiGetStrokeLastPt](#)

[hiStroke](#)

## hiGetStrokeLastPt

```
hiGetStrokeLastPt ()
=> l_point / nil
```

### Description

Returns the point location of the last pixel of a stroke when within a callback initiated by a stroke.

### Argument

None

### Values Returned

|                |                                                                                        |
|----------------|----------------------------------------------------------------------------------------|
| <i>l_point</i> | Last pixel of the stroke, expressed in user units.                                     |
| <i>nil</i>     | There is no current window or no valid application associated with the current window. |

### ***Related Topics***

[Stroke Functions](#)

[hiGetStrokeBBox](#)

[hiGetStrokeFirstPt](#)

[hiStroke](#)



## Cadence User Interface SKILL Reference

### Stroke Functions

---

#### hiStroke

```
hiStroke(
 [l_point]
)
=> t / nil
```

#### Description

Mouse callback that initiates a stroke command.

This command is bound to the right mouse button when the stroke files are loaded, see the [\*Virtuoso Layout Suite L User Guide\*](#). This command only works if the strokes files are loaded.

#### Argument

|                |                                |
|----------------|--------------------------------|
| <i>l_point</i> | Starting point for the stroke. |
|----------------|--------------------------------|

#### Values Returned

|     |                                                  |
|-----|--------------------------------------------------|
| t   | Stroke is not canceled.                          |
| nil | Stroke is canceled. No current window available. |

#### ***Related Topics***

[Stroke Functions](#)

[hiGetStrokeBBox](#)

[hiGetStrokeFirstPt](#)

[hiGetStrokeLastPt](#)