IBM VisualAge for C++ for Windows

S33H-5035-00

Visual Builder Parts Reference

Version 3.5

IBM

IBM VisualAge for C++ for Windows

S33H-5035-00

Visual Builder Parts Reference

Version 3.5

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xxv

First Edition (February 1996)

This edition applies to Version 3.5 of IBM VisualAge for C++ for Windows (33H4979, 33H4980) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

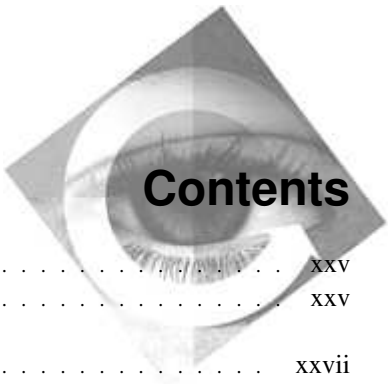
A form for readers’ comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 Eglinton Avenue East
North York, Ontario, Canada. M3C 1H7

You can also send your comments by facsimile (attention: RCF Coordinator), or you can send your comments electronically to IBM. See “Communicating Your Comments to IBM” for a description of the methods. This page immediately precedes the Readers’ Comment Form at the back of this publication.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1995, 1996. All rights reserved. Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.



Contents

Notices xxv
Trademarks xxv

About this Book xxvii
Who Should Use This Book xxvii
How to Use This Book xxvii
How to Use Part Information xxviii
Part Description xxix
Portability xxx
Related Parts xxx
Building Guidelines xxx
Preferred Features xxx
Features xxxi
How to Use Feature Information xxxii
Actions xxxii
Attributes xxxiii
Events xxxiv
How to Get Help xxxiv
Getting Help Inside VisualAge for C++ xxxv
Getting Help from the Command Line xxxvi
Getting Help for a Keyword or Construct xxxvi
Online Documents Available in VisualAge for C++ xxxvi

Part 1. Basic Concepts 1

Chapter 1. The Part Interface 3
Features 3
Connections 3

Chapter 2. The Notification Framework 5
Events, Notifiers, and Observers 5
Attribute Notifications 5
Event Notifications 6

Chapter 3. Portability 7
Portability — vbbase.vbb 7
Portability — vbmm.vbb 34

Chapter 4. Part List 37

© Copyright IBM Corp. 1995, 1996

Part 2. Parts in vbbase.vbb 47

Chapter 5. I0String 59
Preferred Features — I0String 59
Features — I0String 59

Chapter 6. I3StateCheckBox 61
Portability — I3StateCheckBox 61
Related Parts — I3StateCheckBox 62
Building Guidelines — I3StateCheckBox 62
Preferred Features — I3StateCheckBox 64
Features — I3StateCheckBox 64

Chapter 7. IAccelerator 67
Portability — IAccelerator 67
Preferred Features — IAccelerator 67
Features — IAccelerator 67

Chapter 8. IAcceleratorKey 69
Preferred Features — IAcceleratorKey 69
Features — IAcceleratorKey 69

Chapter 9. IAcceleratorTable 71
Preferred Features — IAcceleratorTable 71
Features — IAcceleratorTable 71

Chapter 10. IAnimatedButton 73
Portability — IAnimatedButton 73
Related Parts — IAnimatedButton 74
Building Guidelines — IAnimatedButton 74
Preferred Features — IAnimatedButton 77
Features — IAnimatedButton 77

Chapter 11. IBitmapControl 79
Portability — IBitmapControl 79
Building Guidelines — IBitmapControl 80
Preferred Features — IBitmapControl 81
Features — IBitmapControl 81

Chapter 12. ICanvas 83
Portability — ICanvas 84
Related Parts — ICanvas 84
Building Guidelines — ICanvas 85

Preferred Features — ICanvas 86
Features — ICanvas 86

Chapter 13. ICheckBox 89
Portability — ICheckBox 89
Related Parts — ICheckBox 90
Building Guidelines — ICheckBox 90
Preferred Features — ICheckBox 92
Features — ICheckBox 92

Chapter 14. ICircularSlider 95
Portability — ICircularSlider 95
Related Parts — ICircularSlider 96
Building Guidelines — ICircularSlider 97
Preferred Features — ICircularSlider 98
Features — ICircularSlider 98

Chapter 15. ICLibErrorInfo 101
Preferred Features — ICLibErrorInfo 101
Features — ICLibErrorInfo 101

Chapter 16. IClipboard 103
Preferred Features — IClipboard 103
Features — IClipboard 103

Chapter 17. ICollectionViewComboBox 105
Portability — ICollectionViewComboBox 106
Related Parts — ICollectionViewComboBox 107
Building Guidelines — ICollectionViewComboBox 107
Preferred Features — ICollectionViewComboBox 110
Features — ICollectionViewComboBox 111

Chapter 18. ICollectionViewListBox 113
Portability — ICollectionViewListBox 113
Related Parts — ICollectionViewListBox 114
Building Guidelines — ICollectionViewListBox 115
Preferred Features — ICollectionViewListBox 117
Features — ICollectionViewListBox 118

Chapter 19. IColor 121
Preferred Features — IColor 121
Features — IColor 121

Chapter 20. IComboBox 123
Portability — IComboBox 124
Related Parts — IComboBox 125
Building Guidelines — IComboBox 125
Preferred Features — IComboBox 127
Features — IComboBox 127

Chapter 21. IContainerColumn 131
Related Parts — IContainerColumn 131
Building Guidelines — IContainerColumn 132
Preferred Features — IContainerColumn 133
Features — IContainerColumn 134

Chapter 22. IContainerObject 135
Related Parts — IContainerObject 135
Preferred Features — IContainerObject 135
Features — IContainerObject 135

Chapter 23. ICustomButton 137
Portability — ICustomButton 137
Related Parts — ICustomButton 138
Building Guidelines — ICustomButton 138
Preferred Features — ICustomButton 140
Features — ICustomButton 140

Chapter 24. IDate 143
Portability — IDate 143
Preferred Features — IDate 143
Features — IDate 143

Chapter 25. IDrawingCanvas 145
Portability — IDrawingCanvas 145
Related Parts — IDrawingCanvas 146
Building Guidelines — IDrawingCanvas 147
Preferred Features — IDrawingCanvas 148
Features — IDrawingCanvas 148

Chapter 26. IDynamicLinkLibrary 151
Preferred Features — IDynamicLinkLibrary 151
Features — IDynamicLinkLibrary 151

Chapter 27. IEntryField 153
Portability — IEntryField 153

Related Parts — IEntryField 154
Building Guidelines — IEntryField 154
Preferred Features — IEntryField 156
Features — IEntryField 156

Chapter 28. IFlyOverHelpHandler 159
Preferred Features — IFlyOverHelpHandler 159
Features — IFlyOverHelpHandler 159

Chapter 29. IFlyText 161
Portability — IFlyText 161
Preferred Features — IFlyText 162
Features — IFlyText 162

Chapter 30. IFont 165
Portability — IFont 165
Preferred Features — IFont 165
Features — IFont 166

Chapter 31. IFrameWindow 167
Portability — IFrameWindow 168
Related Parts — IFrameWindow 169
Building Guidelines — IFrameWindow 169
Preferred Features — IFrameWindow 172
Features — IFrameWindow 173

Chapter 32. IGraphicPushButton 175
Portability — IGraphicPushButton 175
Related Parts — IGraphicPushButton 176
Building Guidelines — IGraphicPushButton 176
Preferred Features — IGraphicPushButton 177
Features — IGraphicPushButton 178

Chapter 33. IGroupBox 181
Portability — IGroupBox 181
Related Parts — IGroupBox 182
Building Guidelines — IGroupBox 182
Preferred Features — IGroupBox 183
Features — IGroupBox 183

Chapter 34. IGUIErrorInfo 187
Preferred Features — IGUIErrorInfo 187
Features — IGUIErrorInfo 187

Chapter 35. IHelpWindow 189
Portability — IHelpWindow 189
Related Parts — IHelpWindow 190
Building Guidelines — IHelpWindow 190
Preferred Features — IHelpWindow 191
Features — IHelpWindow 192

Chapter 36. IIconControl 195
Portability — IIconControl 195
Building Guidelines — IIconControl 196
Preferred Features — IIconControl 196
Features — IIconControl 196

Chapter 37. IListBox 199
Portability — IListBox 199
Related Parts — IListBox 200
Building Guidelines — IListBox 201
Preferred Features — IListBox 202
Features — IListBox 203

Chapter 38. IMenu 205
Portability — IMenu 206
Related Parts — IMenu 206
Building Guidelines — IMenu 207
Preferred Features — IMenu 208
Features — IMenu 208

Chapter 39. IMenuCascade 211
Portability — IMenuCascade 211
Related Parts — IMenuCascade 211
Building Guidelines — IMenuCascade 211
Preferred Features — IMenuCascade 213
Features — IMenuCascade 213

Chapter 40. IMenuItem 215
Portability — IMenuItem 215
Related Parts — IMenuItem 215
Building Guidelines — IMenuItem 215
Preferred Features — IMenuItem 217
Features — IMenuItem 218

Chapter 41. IMenuSeparator 219
Portability — IMenuSeparator 219

Related Parts — IMenuSeparator 219
Building Guidelines — IMenuSeparator 219
Preferred Features — IMenuSeparator 220
Features — IMenuSeparator 220

Chapter 42. IMessageBox 221
Related Parts — IMessageBox 221
Building Guidelines — IMessageBox 221
Preferred Features — IMessageBox 223
Features — IMessageBox 223

Chapter 43. IMultiCellCanvas 225
Portability — IMultiCellCanvas 226
Related Parts — IMultiCellCanvas 226
Building Guidelines — IMultiCellCanvas 227
Preferred Features — IMultiCellCanvas 229
Features — IMultiCellCanvas 229

Chapter 44. IMultiLineEdit 231
Portability — IMultiLineEdit 231
Related Parts — IMultiLineEdit 232
Building Guidelines — IMultiLineEdit 232
Preferred Features — IMultiLineEdit 234
Features — IMultiLineEdit 234

Chapter 45. INotebook 237
Portability — INotebook 237
Related Parts — INotebook 238
Building Guidelines — INotebook 239
Preferred Features — INotebook 240
Features — INotebook 241

Chapter 46. INumericSpinButton 243
Portability — INumericSpinButton 243
Related Parts — INumericSpinButton 244
Building Guidelines — INumericSpinButton 245
Preferred Features — INumericSpinButton 246
Features — INumericSpinButton 246

Chapter 47. IOutlineBox 249
Portability — IOutlineBox 249
Related Parts — IOutlineBox 250
Building Guidelines — IOutlineBox 250

Preferred Features — IOutlineBox 251
Features — IOutlineBox 251

Chapter 48. IPair 253
Preferred Features — IPair 253
Features — IPair 253

Chapter 49. IPoint 255
Preferred Features — IPoint 255
Features — IPoint 255

Chapter 50. IPointArray 257
Preferred Features — IPointArray 257
Features — IPointArray 257

Chapter 51. IProfile 259
Portability — IProfile 259
Preferred Features — IProfile 259
Features — IProfile 259

Chapter 52. IProgressIndicator 261
Portability — IProgressIndicator 261
Related Parts — IProgressIndicator 262
Building Guidelines — IProgressIndicator 262
Preferred Features — IProgressIndicator 264
Features — IProgressIndicator 265

Chapter 53. IPushButton 267
Portability — IPushButton 267
Related Parts — IPushButton 268
Building Guidelines — IPushButton 268
Preferred Features — IPushButton 270
Features — IPushButton 270

Chapter 54. IRadioButton 273
Portability — IRadioButton 273
Related Parts — IRadioButton 274
Building Guidelines — IRadioButton 275
Preferred Features — IRadioButton 277
Features — IRadioButton 277

Chapter 55. IRange 279
Preferred Features — IRange 279

Features — IRange 279

Chapter 56. IRectangle 281
Preferred Features — IRectangle 281
Features — IRectangle 282

Chapter 57. IResourceId 283
Preferred Features — IResourceId 283
Features — IResourceId 283

Chapter 58. IResourceLibrary 285
Preferred Features — IResourceLibrary 285
Features — IResourceLibrary 285

Chapter 59. IScrollBar 287
Portability — IScrollBar 287
Related Parts — IScrollBar 288
Building Guidelines — IScrollBar 289
Preferred Features — IScrollBar 290
Features — IScrollBar 290

Chapter 60. ISetCanvas 293
Portability — ISetCanvas 294
Related Parts — ISetCanvas 294
Building Guidelines — ISetCanvas 295
Preferred Features — ISetCanvas 296
Features — ISetCanvas 297

Chapter 61. ISize 299
Preferred Features — ISize 299
Features — ISize 299

Chapter 62. ISlider 301
Portability — ISlider 301
Related Parts — ISlider 302
Building Guidelines — ISlider 302
Preferred Features — ISlider 305
Features — ISlider 305

Chapter 63. ISplitCanvas 307
Portability — ISplitCanvas 308
Related Parts — ISplitCanvas 308
Building Guidelines — ISplitCanvas 309

Preferred Features — ISplitCanvas 310
Features — ISplitCanvas 310

Chapter 64. IStandardNotifier 313
Preferred Features — IStandardNotifier 313
Features — IStandardNotifier 313

Chapter 65. IStaticText 315
Portability — IStaticText 315
Related Parts — IStaticText 316
Building Guidelines — IStaticText 316
Preferred Features — IStaticText 317
Features — IStaticText 318

Chapter 66. IString 321
Preferred Features — IString 321
Features — IString 321

Chapter 67. IStringGenerator 325
Features — IStringGenerator 325

Chapter 68. ISystemErrorInfo 327
Preferred Features — ISystemErrorInfo 327
Features — ISystemErrorInfo 327

Chapter 69. ITextSpinButton 329
Portability — ITextSpinButton 329
Related Parts — ITextSpinButton 330
Building Guidelines — ITextSpinButton 331
Preferred Features — ITextSpinButton 332
Features — ITextSpinButton 332

Chapter 70. ITime 335
Portability — ITime 335
Preferred Features — ITime 335
Features — ITime 335

Chapter 71. ITitle 337
Portability — ITitle 337
Related Parts — ITitle 338
Building Guidelines — ITitle 338
Preferred Features — ITitle 339
Features — ITitle 340

Chapter 72. IToolBar 343
Portability — IToolBar 343
Related Parts — IToolBar 344
Building Guidelines — IToolBar 344
Preferred Features — IToolBar 346
Features — IToolBar 346

Chapter 73. IToolBarButton 349
Portability — IToolBarButton 349
Related Parts — IToolBarButton 350
Building Guidelines — IToolBarButton 350
Preferred Features — IToolBarButton 352
Features — IToolBarButton 352

Chapter 74. ITrace 355
Preferred Features — ITrace 355
Features — ITrace 355

Chapter 75. IVBCheckMenuHandler 357
Preferred Features — IVBCheckMenuHandler 357
Features — IVBCheckMenuHandler 357

Chapter 76. IVBContainerControl 359
Portability — IVBContainerControl 360
Related Parts — IVBContainerControl 361
Building Guidelines — IVBContainerControl 361
Preferred Features — IVBContainerControl 365
Features — IVBContainerControl 365

Chapter 77. IVBDragDropHandler 369
Preferred Features — IVBDragDropHandler 369
Features — IVBDragDropHandler 369

Chapter 78. IVBFactory 371
Related Parts — IVBFactory 371
Building Guidelines — IVBFactory 371
Preferred Features — IVBFactory 372
Features — IVBFactory 372

Chapter 79. IVBFileDialog 373
Portability — IVBFileDialog 373
Related Parts — IVBFileDialog 373
Building Guidelines — IVBFileDialog 374

Preferred Features — IVBFileDialog 376
Features — IVBFileDialog 376

Chapter 80. IVBFlyText 377
Portability — IVBFlyText 377
Related Parts — IVBFlyText 378
Building Guidelines — IVBFlyText 378
Preferred Features — IVBFlyText 379
Features — IVBFlyText 379

Chapter 81. IVBFontDialog 381
Portability — IVBFontDialog 381
Related Parts — IVBFontDialog 381
Building Guidelines — IVBFontDialog 382
Preferred Features — IVBFontDialog 383
Features — IVBFontDialog 384

Chapter 82. IVBInfoArea 385
Portability — IVBInfoArea 385
Related Parts — IVBInfoArea 386
Building Guidelines — IVBInfoArea 386
Preferred Features — IVBInfoArea 387
Features — IVBInfoArea 388

Chapter 83. IVBNotebookPage 391
Related Parts — IVBNotebookPage 391
Building Guidelines — IVBNotebookPage 392
Preferred Features — IVBNotebookPage 393
Features — IVBNotebookPage 393

Chapter 84. IVBVariable 395
Related Parts — IVBVariable 395
Building Guidelines — IVBVariable 395
Preferred Features — IVBVariable 397
Features — IVBVariable 397

Chapter 85. IViewPort 399
Portability — IViewPort 399
Related Parts — IViewPort 400
Building Guidelines — IViewPort 400
Preferred Features — IViewPort 401
Features — IViewPort 401

Chapter 86. IVSequence . . . 403
Related Parts — IVSequence . . . 403
Building Guidelines — IVSequence . . . 404
Preferred Features — IVSequence . . . 405
Features — IVSequence . . . 405

Part 3. Sample Parts in vbcc.vbb . . . 407

Chapter 87. IElemPointer . . . 411
Preferred Features — IElemPointer . . . 411
Features — IElemPointer . . . 411

Chapter 88. IMngPointer . . . 413
Preferred Features — IMngPointer . . . 413
Features — IMngPointer . . . 413

Chapter 89. IVBag . . . 415
Preferred Features — IVBag . . . 415
Features — IVBag . . . 415

Chapter 90. IVBagOnBSTKeySortedSet . . . 417
Preferred Features — IVBagOnBSTKeySortedSet . . . 417
Features — IVBagOnBSTKeySortedSet . . . 417

Chapter 91. IVBagOnHashKeySet . . . 419
Preferred Features — IVBagOnHashKeySet . . . 419
Features — IVBagOnHashKeySet . . . 419

Chapter 92. IVBagOnSortedDilutedSequence . . . 421
Preferred Features — IVBagOnSortedDilutedSequence . . . 421
Features — IVBagOnSortedDilutedSequence . . . 421

Chapter 93. IVBagOnSortedLinkedSequence . . . 423
Preferred Features — IVBagOnSortedLinkedSequence . . . 423
Features — IVBagOnSortedLinkedSequence . . . 423

Chapter 94. IVBagOnSortedTabularSequence . . . 425
Preferred Features — IVBagOnSortedTabularSequence . . . 425
Features — IVBagOnSortedTabularSequence . . . 425

Chapter 95. IVDeque . . . 427
Preferred Features — IVDeque . . . 427
Features — IVDeque . . . 427

Chapter 96. IVDequeOnDilutedSequence 429
Preferred Features — IVDequeOnDilutedSequence 429
Features — IVDequeOnDilutedSequence 429

Chapter 97. IVDilutedSequence 431
Preferred Features — IVDilutedSequence 431
Features — IVDilutedSequence 431

Chapter 98. IVEqualitySequence 433
Preferred Features — IVEqualitySequence 433
Features — IVEqualitySequence 434

Chapter 99. IVEqualitySequenceOnDilutedSequence 435
Preferred Features — IVEqualitySequenceOnDilutedSequence 435
Features — IVEqualitySequenceOnDilutedSequence 436

Chapter 100. IVEqualitySequenceOnTabularSequence 437
Preferred Features — IVEqualitySequenceOnTabularSequence 437
Features — IVEqualitySequenceOnTabularSequence 438

Chapter 101. IVHeap 439
Preferred Features — IVHeap 439
Features — IVHeap 439

Chapter 102. IVHeapOnDilutedSequence 441
Preferred Features — IVHeapOnDilutedSequence 441
Features — IVHeapOnDilutedSequence 441

Chapter 103. IVLinkedSequence 443
Preferred Features — IVLinkedSequence 443
Features — IVLinkedSequence 443

Chapter 104. IVQueue 445
Preferred Features — IVQueue 445
Features — IVQueue 445

Chapter 105. IVQueueOnDilutedSequence 447
Preferred Features — IVQueueOnDilutedSequence 447
Features — IVQueueOnDilutedSequence 447

Chapter 106. IVQueueOnTabularSequence 449
Preferred Features — IVQueueOnTabularSequence 449
Features — IVQueueOnTabularSequence 449

Chapter 107. IVSet 451
Preferred Features — IVSet 451
Features — IVSet 451

Chapter 108. IVSetOnBSTKeySortedSet 453
Preferred Features — IVSetOnBSTKeySortedSet 453
Features — IVSetOnBSTKeySortedSet 453

Chapter 109. IVSetOnHashKeySet 455
Preferred Features — IVSetOnHashKeySet 455
Features — IVSetOnHashKeySet 455

Chapter 110. IVSetOnSortedDilutedSequence 457
Preferred Features — IVSetOnSortedDilutedSequence 457
Features — IVSetOnSortedDilutedSequence 457

Chapter 111. IVSetOnSortedLinkedSequence 459
Preferred Features — IVSetOnSortedLinkedSequence 459
Features — IVSetOnSortedLinkedSequence 459

Chapter 112. IVSetOnSortedTabularSequence 461
Preferred Features — IVSetOnSortedTabularSequence 461
Features — IVSetOnSortedTabularSequence 461

Chapter 113. IVSortedBag 463
Preferred Features — IVSortedBag 463
Features — IVSortedBag 463

Chapter 114. IVSortedBagOnSortedDilutedSequence 465
Preferred Features — IVSortedBagOnSortedDilutedSequence 465
Features — IVSortedBagOnSortedDilutedSequence 465

Chapter 115. IVSortedBagOnSortedLinkedSequence 467
Preferred Features — IVSortedBagOnSortedLinkedSequence 467
Features — IVSortedBagOnSortedLinkedSequence 467

Chapter 116. IVSortedBagOnSortedTabularSequence 469
Preferred Features — IVSortedBagOnSortedTabularSequence 469
Features — IVSortedBagOnSortedTabularSequence 469

Chapter 117. IVSortedSet 471
Preferred Features — IVSortedSet 471
Features — IVSortedSet 471

Chapter 118. IVSortedSetOnBSTKeySortedSet 473
Preferred Features — IVSortedSetOnBSTKeySortedSet 473
Features — IVSortedSetOnBSTKeySortedSet 473

Chapter 119. IVSortedSetOnSortedLinkedSequence 475
Preferred Features — IVSortedSetOnSortedLinkedSequence 475
Features — IVSortedSetOnSortedLinkedSequence 475

Chapter 120. IVSortedSetOnSortedTabularSequence 477
Preferred Features — IVSortedSetOnSortedTabularSequence 477
Features — IVSortedSetOnSortedTabularSequence 477

Chapter 121. IVStack 479
Preferred Features — IVStack 479
Features — IVStack 479

Chapter 122. IVStackOnTabularSequence 481
Preferred Features — IVStackOnTabularSequence 481
Features — IVStackOnTabularSequence 481

Chapter 123. IVTabularSequence 483
Preferred Features — IVTabularSequence 483
Features — IVTabularSequence 483

Part 4. Parts in vbmm.vbb 485

Chapter 124. IMM24FramesPerSecondTime 489
Preferred Features — IMM24FramesPerSecondTime 489
Features — IMM24FramesPerSecondTime 489

Chapter 125. IMM25FramesPerSecondTime 491
Preferred Features — IMM25FramesPerSecondTime 491
Features — IMM25FramesPerSecondTime 491

Chapter 126. IMM30FramesPerSecondTime 493
Preferred Features — IMM30FramesPerSecondTime 493
Features — IMM30FramesPerSecondTime 493

Chapter 127. IMMAmpMixer 495
Portability — IMMAmpMixer 495
Related Parts — IMMAmpMixer 495
Building Guidelines — IMMAmpMixer 496
Preferred Features — IMMAmpMixer 497

Features — IMMAmpMixer 497

Chapter 128. IMMAudioBuffer 499
Portability — IMMAudioBuffer 499
Preferred Features — IMMAudioBuffer 499
Features — IMMAudioBuffer 499

Chapter 129. IMMAudioCD 501
Portability — IMMAudioCD 501
Related Parts — IMMAudioCD 501
Building Guidelines — IMMAudioCD 502
Preferred Features — IMMAudioCD 503
Features — IMMAudioCD 504

Chapter 130. IMMAudioCDContents 507
Preferred Features — IMMAudioCDContents 507
Features — IMMAudioCDContents 507

Chapter 131. IMMDigitalVideo 509
Portability — IMMDigitalVideo 509
Related Parts — IMMDigitalVideo 510
Building Guidelines — IMMDigitalVideo 510
Preferred Features — IMMDigitalVideo 512
Features — IMMDigitalVideo 512

Chapter 132. IMMErrorInfo 515
Preferred Features — IMMErrorInfo 515
Features — IMMErrorInfo 515

Chapter 133. IMMHourMinSecFrameTime 517
Preferred Features — IMMHourMinSecFrameTime 517
Features — IMMHourMinSecFrameTime 517

Chapter 134. IMMHourMinSecTime 519
Preferred Features — IMMHourMinSecTime 519
Features — IMMHourMinSecTime 519

Chapter 135. IMMMasterAudio 521
Related Parts — IMMMasterAudio 521
Building Guidelines — IMMMasterAudio 522
Preferred Features — IMMMasterAudio 523
Features — IMMMasterAudio 523

Chapter 136. IMMMillisecondTime 525
Preferred Features — IMMMillisecondTime 525
Features — IMMMillisecondTime 525

Chapter 137. IMMMinSecFrameTime 527
Preferred Features — IMMMinSecFrameTime 527
Features — IMMMinSecFrameTime 527

Chapter 138. IMMPlayerPanel 529
Related Parts — IMMPlayerPanel 529
Building Guidelines — IMMPlayerPanel 530
Preferred Features — IMMPlayerPanel 531
Features — IMMPlayerPanel 531

Chapter 139. IMMSequencer 533
Portability — IMMSequencer 533
Related Parts — IMMSequencer 533
Building Guidelines — IMMSequencer 534
Preferred Features — IMMSequencer 535
Features — IMMSequencer 536

Chapter 140. IMMSpeed 537
Preferred Features — IMMSpeed 537
Features — IMMSpeed 537

Chapter 141. IMMTime 539
Preferred Features — IMMTime 539
Features — IMMTime 539

Chapter 142. IMMTrackMinSecFrameTime 541
Preferred Features — IMMTrackMinSecFrameTime 541
Features — IMMTrackMinSecFrameTime 541

Chapter 143. IMMWaveAudio 543
Portability — IMMWaveAudio 543
Related Parts — IMMWaveAudio 544
Building Guidelines — IMMWaveAudio 544
Preferred Features — IMMWaveAudio 546
Features — IMMWaveAudio 546

Part 5. Sample Parts in vbsample.vbb 549

Chapter 144. floatSamples 553

Preferred Features — floatSamples 553
Features — floatSamples 553

Chapter 145. IAddress 555
Preferred Features — IAddress 555
Features — IAddress 555

Chapter 146. ICompany 557
Preferred Features — ICompany 557
Features — ICompany 557

Chapter 147. ICustomer 559
Preferred Features — ICustomer 559
Features — ICustomer 559

Chapter 148. IOrderedRecord 561
Preferred Features — IOrderedRecord 561
Features — IOrderedRecord 561

Chapter 149. ioSamples 563
Preferred Features — ioSamples 563
Features — ioSamples 563

Chapter 150. IRecord 565
Preferred Features — IRecord 565
Features — IRecord 565

Chapter 151. IVBBooleanPart 567
Preferred Features — IVBBooleanPart 567
Features — IVBBooleanPart 567

Chapter 152. IVBDoublePart 569
Preferred Features — IVBDoublePart 569
Features — IVBDoublePart 569

Chapter 153. IVBLogicalAndPart 571
Preferred Features — IVBLogicalAndPart 571
Features — IVBLogicalAndPart 571

Chapter 154. IVBLogicalOrPart 573
Preferred Features — IVBLogicalOrPart 573
Features — IVBLogicalOrPart 573

Chapter 155. IVBLongPart 575
Preferred Features — IVBLongPart 575
Features — IVBLongPart 575

Chapter 156. IVBShortPart 577
Preferred Features — IVBShortPart 577
Features — IVBShortPart 577

Chapter 157. IVBStringPart 579
Preferred Features — IVBStringPart 579
Features — IVBStringPart 579

Chapter 158. IVBUnsignedLongPart 581
Preferred Features — IVBUnsignedLongPart 581
Features — IVBUnsignedLongPart 581

Chapter 159. IVBUnsignedShortPart 583
Preferred Features — IVBUnsignedShortPart 583
Features — IVBUnsignedShortPart 583

Chapter 160. mathSamples 585
Preferred Features — mathSamples 585
Features — mathSamples 585

Chapter 161. staticWindowSamples 587
Preferred Features — staticWindowSamples 587
Features — staticWindowSamples 587

Chapter 162. stdioSamples 589
Preferred Features — stdioSamples 589
Features — stdioSamples 589

Chapter 163. stdlibSamples 591
Preferred Features — stdlibSamples 591
Features — stdlibSamples 591

Part 6. Part Interface Features 595

Chapter 164. Actions 653

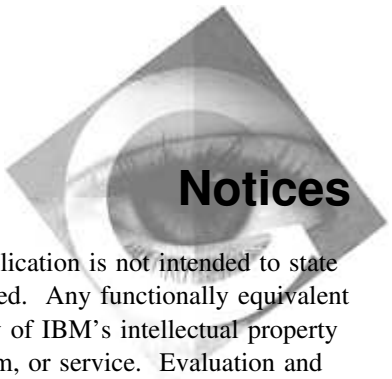
Chapter 165. Attributes 811

Chapter 166. Events 959

Glossary 973

Bibliography 981
The IBM VisualAge for C++ Library 981
C and C++ Related Publications 981

Index 983



Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights can be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the

IBM Director of Licensing,
IBM Corporation,
500 Columbus Avenue,
Thornwood, NY 10594,
USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Canada Ltd., Department 071, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, Canada. Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:

AIX
BookManager
Common User Access
CUA
IBM

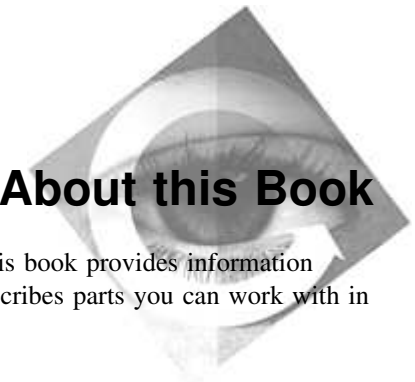
© Copyright IBM Corp. 1995, 1996

IBMLink
Library Reader
OS/2
OS/2 Warp
PROFS
QuickBrowse
SAA
System Object Model
VisualAge
WorkFrame
Workplace Shell

Windows is a trademark of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks of others.

IBM's VisualAge products and services are not associated with or sponsored by Visual Edge Software, Ltd.



About this Book

Welcome to the *Visual Builder Parts Reference*. This book provides information about parts you can use to build applications. It describes parts you can work with in the Composition Editor.

Who Should Use This Book

Programmers who want to develop C++ applications using Visual Builder should read this book. It is written primarily for the programmer building applications from Visual Builder parts using the Composition Editor.

How to Use This Book

Use this book to find detailed information about Visual Builder parts.

Visual Builder concepts

If you are not familiar with parts, features, and connections, read "Basic Concepts" on page 1.

Part information

If you need information about a part that you want to use in the Composition Editor, see the chapter for that part.

Part chapters are grouped by part files. This book covers parts from the following part files:

- vbbase.vbb
- vbcc.vbb
- vbmm.vbb
- vbsample.vbb

For each part file, part chapters are organized alphabetically by part name.

See "How to Use Part Information" on page xxviii for an explanation of the information available in a part chapter.

Feature information

© Copyright IBM Corp. 1995, 1996

Actions, attributes, and events are part interface features. If you need information about an action, attribute, or event, find the feature description in the Part Interface Features chapter for the feature type.

See "How to Use Feature Information" on page xxxii for an explanation of the information available in a feature chapter.

Class information

If you need information about the class that supports a part or on a class member that implements a feature, refer to the *IBM Open Class Library Reference*.

Highlighting conventions

Key terms are emphasized as follows:

- When a conceptual term is introduced and defined, it appears in italics.
- User interface elements appear in bold type.
- Part features (actions, attributes, and events) appear in italics when they are referred to in text.
- Code segments appear in a Courier font.

How to Use Part Information

If you want to use a part you are not familiar with, read the part description first. Then, if building guidelines are provided, read the guidelines to learn how to use the part in your own composite part.

Part chapters can contain several topics. The extent of information varies by part file and part. Supported parts in vbbase.vbb and vbmm.vbb are more extensively documented than sample parts are. Generally, parts on the parts palette are the most fully documented.

All part chapters include the following topics:

- "Part Description" on page xxix
- "Preferred Features" on page xxx
- "Features" on page xxxi

Some part chapters also include the following topics:

- "Portability" on page xxx
- "Related Parts" on page xxx
- "Building Guidelines" on page xxx

Part Description

All part chapters include a description of the part. This description can include the following part information:

- **Description**: This short description appears in the status area of the Composition Editor when the part is selected on the parts palette.
- **Part type**: The part type is determined by the following characteristics of the part:
  - Is it a class?
  - Can it notify other parts about events or value changes?
  - Does it appear in the Composition Editor as an instance of the class?

  Depending on these characteristics, the part type is one of the following:
  - **Visual**. A visual part is a notifier class that appears in the Composition Editor as an instance of the class. It provides a visual control in your application's user interface.
  - **Nonvisual**. A nonvisual part is a notifier class that is represented in the Composition Editor by an icon. Generally, a nonvisual part does not appear in your application's user interface.
  - **Class**. A class interface part is a class that is not a notifier. It is represented in the Composition Editor by an icon. Generally, a class interface part does not appear in your application's user interface.
  - **Function group**. A function group part is a group of related functions, not a class. It is represented in the Composition Editor by an icon.

  Note that most, but not all, components of your user interface are visual parts. An IHelpWindow part is an example of a notifier that is a user interface component but is represented by an icon in the Composition Editor. Therefore, it is a nonvisual part. An IMessageBox part is an example of a user interface component that is not a notifier. Therefore, it is a class interface part, and is represented in the Composition Editor by an icon.
- **Part file**: Identifies the data file containing the part definition. You need to load this part file before you can use the part. Visual Builder automatically loads the vbbase.vbb part file for you.
- **Header file**: Identifies the C++ header file for the class supporting the part.
- **Derivation**: Identifies the class hierarchy for the part. Abstract parts are shown in italics. If you need more information on the class, refer to the *IBM Open Class Library Reference*.
- A graphic representation of the part as it appears in the Composition Editor.
- A documentary text description covering the following points:

– What to use the part for in your applications
– What the user can do with the part

Portability

Most part interface features are supported across operating system platforms. This topic, when present, identifies portability limitations for the part.

Portability limitations are categorized as follows:

- **Features not defined for a platform:** These features are not recognized on the specified platform.
- **Features not active on a platform:** These features are recognized but ignored on the specified platform.
- **Features active with restriction on a platform:** These features are active but restricted on the specified platform. They are attributes that can be queried but not set.

For a master list of portability limitations for all parts, see “Portability” on page 7.

Related Parts

Several groups of related parts are listed for parts on the parts palette. These lists provide references to the parts.

Building Guidelines

Building guidelines help you with part construction in the Composition Editor. These guidelines give you information about using Visual Builder part settings and features for some common tasks related to building composite parts. The guidelines describe the use of many attributes, actions, events, and settings in support of these tasks.

The building guidelines are intended to be representative rather than comprehensive. They focus on common building tasks particularly applicable to each part. You can generally perform many tasks with a part that are not described in the building guidelines for the part. For example, enabling and disabling actions are common to controls in general but are described only for buttons, check boxes, and menu items.

Preferred Features

A list of preferred features identifies the most commonly used features of the part. You can read a short description of each preferred feature provided by Visual Builder.

Features

A table of part features lists all actions, attributes, and events of the part. The table includes features defined in the part and features inherited from other parts. Features are listed in alphabetical order by feature type.

You can find feature implementation information, including member function signatures, in the Part Interface Features chapters of this book. See "How to Use Feature Information" on page xxxii for information about feature chapters.

You can also find part feature information in Visual Builder's **Feature Implementation Browser**. To view this information, open the part contextual menu. Then, select **Browse part features** on the menu. In the **Feature Implementation Browser**, you can select features to view implementation information. You can also click on the **Reference** push button to open the online reference chapter for the part.

Most parts have attributes and actions, but some do not have events. Class interface parts do not support notification and, therefore, have no events. If the part has events, the table lists actions, attributes, and events. If the part does not have events, the table lists actions, attributes, and attribute data types. For function list parts, the table lists only actions and functions.

Actions

The Action column gives the name of each action. Text emphasis and symbols have the following meanings:

featureName	The action is inherited.
featureName	The action is defined by this part.
■	The action has parameters.

The Function column, if present, gives the function signature for function list actions.

Attributes

The Attribute column gives the name of each attribute. Text emphasis and symbols have the following meanings:

featureName	The attribute is inherited.
featureName	The attribute is defined by this part.
■	The attribute has parameters. This means it can be set.
►	The attribute can notify observers when its value changes. This means that you can add a connection that gets the new value whenever the value changes.

The Type column, if present, gives the data type of each attribute.

Events

The Event column gives the name of each event. Text emphasis and symbols have the following meanings:

featureName	The event is inherited.
featureName	The event is defined by this part.
featureName	The event is an attribute event that is inherited.
featureName	The event is an attribute event that is defined by this part.
∎	The event provides a parameter. This means that the event sends event data with the notification. If you connect the event to an action with a corresponding parameter, the event data provides the parameter value for the action.

How to Use Feature Information

To find information about a feature, you need to know the feature name and the feature type. For features defined by more than one part, you also need to know the defining part name.

For example, if you need information about the *enable* action of the IPushButton part, look in the chapter on actions. Find the *enable* action in the alphabetically ordered list of actions. You will see that the *enable* action is defined by more than one part. IPushButton's *enable* action is defined by IWindow, which IPushButton is derived from, so the **enable — IWindow** description is the one you are interested in. None of the other *enable* actions are defined by parts in IPushButton's derivation hierarchy.

Each feature heading shows the feature name followed by the name of the part that defines the feature. If the defining part is from a part file other than vbbase.vbb, the part file name follows the part name.

The feature type chapters are described in the following topics:

- "Actions"
- "Attributes" on page xxxiii
- "Events" on page xxxiv

Actions

The actions chapter describes actions of all parts covered by this book. The actions are listed in alphabetical order.

Action descriptions provide the following information:

- A text **Description**
- **Member** information

The **Description** topic shows the action's brief description from the part file.

The **Member** topic shows the signature of the *action member function* that performs the action. Text emphasis has the following meanings:

`returnValue` You can retrieve the return value with a connection.
`returnValue` You cannot connect to the return value.
`parameterName` You must provide a value for the parameter.
`parameterName``=defaultValue`
You can either provide a value for the parameter or use the default value.

Attributes

The attributes chapter describes attributes of all parts covered by this book. The attributes are listed in alphabetical order.

Attribute descriptions provide the following information:

- A text **Description**
- **Get** member information
- **Set** member information
- Notification **ID**

Some attributes do not have **Set** or **ID** topics.

The **Description** topic shows the attribute's brief description from the part file.

The **Get** topic shows the signature of the *get member function* that retrieves the attribute value. Text emphasis has the following meanings:

`returnValue` You can retrieve the return value with a connection.
`parameterName``=defaultValue`
The attribute either provides a value for the parameter or uses the default value.

The **Set** topic shows the signature of the *set member function* that changes the attribute value. Text emphasis has the following meanings:

`returnValue` You cannot connect to the return value.
`parameterName` You must provide a value for the parameter.
`parameterName``=defaultValue`
You can either provide a value for the parameter or use the default value.

The **ID** topic shows the *notification identifier* that is sent with value change notifications. It identifies the notification as a value change for the particular attribute.

The attribute sends the its new value as event data with the notification.

Events

The events chapter describes events of all parts covered by this book. The events are listed in alphabetical order.

Event descriptions provide the following information:

- A text **Description**
- Notification **ID**
- Notification **Parameter**

Some events do not have **Parameter** topic.

The **Description** topic shows the event's brief description from the part file.

The **ID** topic shows the *notification identifier* that is sent with event notifications. It identifies the notification as an occurrence of the particular event.

The **Parameter** topic shows event data that the event sends with the notification. The parameter name appears first. The parameter type, in parentheses, follows the name.

How to Get Help

There are three kinds of online information available to you while you are using VisualAge for C++:

Online documents

These are complete documents, like the one you are reading now, presented online. These documents contain detailed information on the different aspects of VisualAge for C++. For your convenience, the online documents are presented in:

- Standard format (`.INF` files). See "Getting Help Inside VisualAge for C++" on page xxxv for instructions on opening standard format documents from inside VisualAge for C++. See "Getting Help from the Command Line" on page xxxvi for instructions on opening standard format documents from the command line. For a list of the VisualAge for C++ documents that are available in standard format, see "Online Documents Available in VisualAge for C++" on page xxxvi.

Contextual help

Contextual help is available throughout VisualAge for C++. This help tells you all about the elements that you see in the interface, including menus, entry fields, and pushbuttons.

How Do I **help**

Many of the common tasks that you want to perform with VisualAge for C++ are described in *How Do I* help. The *How Do I* help for a task gives you step-by-step instructions for completing the task. There is overall *How Do I* help for VisualAge for C++, as well as individual task lists for each of its components.

Getting Help Inside VisualAge for C++

All three kinds of help are available directly within the VisualAge for C++ interface:

- To get general contextual help for the component of VisualAge for C++ that you are using, press F1 anywhere in the window.
- To get contextual help on a particular menu, menu item, or button, highlight the element and press F1.
- To get access to all of the help information that is available to you in a particular window, click on **Help** in the menu bar at the top of the window. This menu includes the following selections:
  - **Help Index**, an alphabetical list of all of the help topics that are available from this window
  - **General Help**, overall help for the window
  - **Using Help**, general information about the help facility
  - **How Do I...**, the How Do I help for the component
  - **Product Information**, a dialog that shows the level of VisualAge for C++ being used

  In addition, there are selections that let you open all of online documents that are available in VisualAge for C++.
- To get detailed information, open the **Online Information** notebook in the VisualAge for C++ folder. In this notebook you will find tabs for **Guides**, **References**, and **How Do I** help. Each page in the notebook lists a variety of online documents that describe, in detail, the different aspects of VisualAge for C++. To open a particular online document, select the radio button for the document, and click on the **View** pushbutton.

Getting Help from the Command Line

If you want, you can look at the online documents by issuing the `iview` command. The installation routine stores the online document files in the `\IBMCPPW\HELP` directory. To view the Language Reference, for example, make `C:\IBMCPPW\HELP` your current directory (substituting the drive where you installed VisualAge for C++ for `C:`) and enter the following command:

```
IVIEW CPPLNG.INF
```

If you want to get information on a specific topic, you can specify a word or a series of words after the file name. If the words appear in an entry in the table of contents or the index, the online document is opened to the associated section. For example, if you want to read the section on operator precedence in the Language Reference, you can enter the following command:

```
IVIEW CPPLNG.INF OPERATOR PRECEDENCE
```

Getting Help for a Keyword or Construct

If you are editing a file using the Editor, you can get help for a keyword or construct by moving the cursor to the word and pressing Ctrl+H. In the other tools, you can get help for a keyword or construct by highlighting the word and pressing Ctrl+H.

Online Documents Available in VisualAge for C++

The following documents are available in standard format:

Building VisualAge for C++ Parts for Fun and Profit	Open Class Library Reference
C Library Reference	Open Class Library User's Guide
Editor Command Reference	Programming Guide
Frequently Asked Questions	SOM Programming Guide
Installation Guide & Product Overview	SOM Programming Reference
IPF User's Guide	User's Guide
IPF Guide and Reference	Visual Builder User's Guide
Language Reference	Visual Builder Parts Reference

Part 1. Basic Concepts

The following chapters introduce concepts related to the part interface and the notification framework.

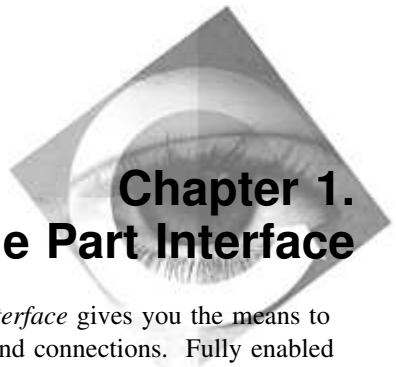
Chapter 1. The Part Interface 3
Features 3
Connections 3

Chapter 2. The Notification Framework 5
Events, Notifiers, and Observers 5
Attribute Notifications 5
Event Notifications 6

Chapter 3. Portability 7
Portability — vbbase.vbb 7
Portability — vbmm.vbb 34

Chapter 4. Part List 37

© Copyright IBM Corp. 1995, 1996



Chapter 1. The Part Interface

Visual Builder packages classes as parts. The *part interface* gives you the means to work visually with classes through standard features and connections. Fully enabled parts can notify each other of value or state changes that occur in them. Class interface parts provide limited features and support connections but cannot send notifications.

For information on implementing parts and features, refer to *Building VisualAge for C++ Parts for Fun and Profit*.

Features

Part interface *features* organize class members as elements of attributes, actions, and events. These features have the following characteristics:

- An *attribute* is a property of a part. Class member functions enable you to get and set the property. Most attributes can be set as well as queried. Many attributes also have notification IDs that are used to report attribute changes.
- An *action* is a behavior of a part. A class member function performs the part behavior.
- An *event* informs other parts of an occurrence in a part. A notification ID identifies the event.

For more information on features, refer to the *Visual Builder User's Guide*.

Connections

Part connections give you the ability to visually relate part features in the following ways:

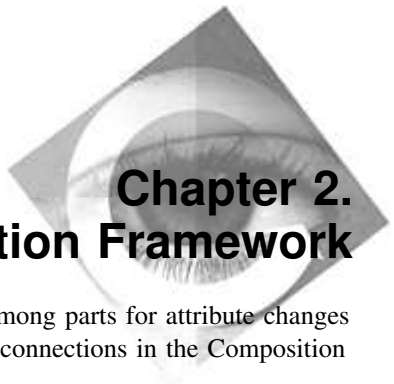
- An *attribute-to-action connection* maps an attribute to an action. When the attribute value is changed, the action is performed.
- An *attribute-to-attribute connection* maps an attribute to another attribute. When the source attribute value is changed, the other attribute is updated with the same value.
- An *event-to-action connection* maps an event to an action. When the event occurs, the action is performed.
- An *event-to-attribute connection* maps an event to an attribute. When the event occurs, the attribute is updated with a new value.

© Copyright IBM Corp. 1995, 1996

- An *event-to-member-function connection* maps an event to a class member function. When the event occurs, the member function is called.
- A *parameter connection* maps a value to a parameter on another connection. The parameter connection can be made between the parameter and any of the following:
  - An attribute
  - An action
  - A member function
  - Custom logic

For more information on connections, refer to the *Visual Builder User's Guide*.

"The Notification Framework" on page 5 describes class support for these connections.



Chapter 2. The Notification Framework

The *notification framework* supports communication among parts for attribute changes and other events. This communication enables visual connections in the Composition Editor.

As a visual application builder, you need not be concerned with implementation details of the notification framework. Visual Builder provides this support for visual connections. Be aware, however, that you can make connections only from attributes with notification IDs and from events. All events have notification IDs.

For more information on implementing notifications, refer to *Building VisualAge for C++ Parts for Fun and Profit*.

Events, Notifiers, and Observers

In this section, "event" is used in the general sense. It represents either an attribute-change event or an event feature of the part interface.

A part in which an event occurs can notify other parts about the event. The reporting part is called the *notifier*. Each part to be notified of an event is registered as an *observer* of the event. The addObserver action performs this function.

The notifier informs observers of an event by using the notifyObservers action. A notification ID identifies the event, and notification data, if any, provides additional information.

A notifier can enable and disable notification of observers. The enableNotification action turns on notification for a part, and disableNotification turns it off.

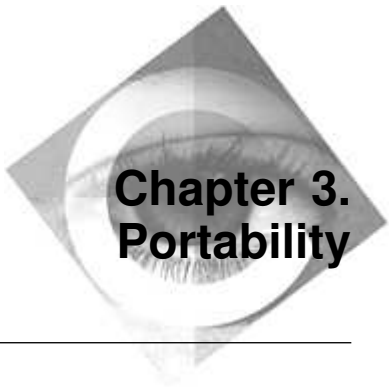
Attribute Notifications

Notifications are commonly used to inform observers when an attribute value changes. A notifier sends this notification when an attribute is changed and the attribute has a notification ID. The notifier sends the new attribute value as notification data.

© Copyright IBM Corp. 1995, 1996

Event Notifications

Notifications are used to inform observers when an event defined by the part interface occurs. The notifier can send information about some events as notification data.



Chapter 3. Portability

Portability — vbbase.vbb

The following portability restrictions apply to parts in the vbbase.vbb part file.

I3StateCheckBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IAccelerator

Features not defined for Windows:

- unset

© Copyright IBM Corp. 1995, 1996

Portability

IAnimatedButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IBitmapControl

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor

- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ICanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ICheckBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

Portability

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ICircularSlider

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ICollectionViewComboBox

Features not defined for Windows:

- readOnly
- messageQueue

Features not active on Windows:

- disableAutoScroll
- disableAutoTab
- disableCommand
- disableInsertMode
- disableMargin
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment
- charType
- autoScroll
- autoTab
- command
- insertMode
- margin

ICollectionViewListBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- disableExtendedSelect
- disableMultipleSelect
- activeColor
- borderColor

Portability

- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- extendedSelect
- multipleSelect

ICComboBox

Features not defined for Windows:

- readOnly
- messageQueue

Features not active on Windows:

- disableAutoScroll
- disableAutoTab
- disableCommand
- disableInsertMode
- disableMargin
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor

- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment
- charType
- autoScroll
- autoTab
- command
- insertMode
- margin

ICustomButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Portability

IDate

Features not defined for Windows:

- asCDATE

IDrawingCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IEntryField

Features not defined for Windows:

- readOnly
- messageQueue

Features not active on Windows:

- disableAutoScroll
- disableAutoTab
- disableCommand
- disableInsertMode
- disableMargin
- activeColor
- borderColor

- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment
- charType
- autoScroll
- autoTab
- command
- insertMode
- margin

IFlyText

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor

Portability

- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IFont

Features not defined for Windows:

- avgLowercase
- avgUppercase
- fattrs
- fontmetrics
- maxLowercaseAscender
- maxLowercaseDescender
- subscriptOffset
- subscriptSize
- superscriptOffset
- superscriptSize

IFrameWindow

Features not defined for Windows:

- shareParentDBCSStatus
- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IGraphicPushButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- border
- removeBorder
- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IGroupBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor

Portability

- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IHelpWindow

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IIconControl

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IListBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- disableExtendedSelect
- disableMultipleSelect
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- extendedSelect
- multipleSelect

Portability

IMenu

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IMenuCascade

Features active with restriction on Windows:

- framed
- noDismiss

IMenuItem

Features active with restriction on Windows:

- framed
- noDismiss

IMenuSeparator

Features active with restriction on Windows:

- framed
- noDismiss

IMultiCellCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IMultiLineEdit

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- disableWordWrap
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor

Portability

- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- wordWrap

INotebook

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

INumericSpinButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment

IOutlineBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Portability

IProfile

Features not defined for Windows:

- handle

IProgressIndicator

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IPushButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- border
- removeBorder
- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor

- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IRadioButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- cursorSelect
- disableCursorSelect
- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Portability

IScrollBar

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ISetCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor

- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ISlider

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ISplitCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor

Portability

- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IStaticText

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

ITextSpinButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment

ITime

Features not defined for Windows:

- asCTIME

ITitle

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeTextBackgroundColor
- activeTextForegroundColor
- inactiveTextBackgroundColor
- inactiveTextForegroundColor
- resetActiveTextBackgroundColor
- resetActiveTextForegroundColor
- resetInactiveTextBackgroundColor
- resetInactiveTextForegroundColor
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor

Portability

- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IToolBar

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IToolBarButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor

- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IVBContainerControl

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Portability

IVBFileDialog

Features not defined for Windows:

- saveAsEAType

IVBFlyText

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IVBFontDialog

Features not defined for Windows:

- emHeight
- externalLeading
- fontFamily
- fontWeight
- fontWidth
- nominalPointSize
- xHeight

IVBInfoArea

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

IViewPort

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor

Portability

- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Portability — vbmm.vbb

The following portability restrictions apply to parts in the vbmm.vbb part file.

IMMAmpMixer

Features not defined for Windows:

- prerollTime
- prerollType

IMMAudioBuffer

Features not defined for Windows:

- bitsPerSample
- blockAlignment
- bytesPerSecond
- channels
- contentType
- data
- format
- headerData
- length
- mediaType
- samplesPerSecond
- setData

IMMAudioCD

Features not defined for Windows:

- lockDoor
- unlockDoor
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

IMMCDXA

Features not defined for Windows:

- upc
- lockDoor
- unlockDoor
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

IMMDigitalVideo

Features not defined for Windows:

- fastSpeed
- monitoringEnabled
- monitorHandle
- slowSpeed
- disableMonitoring
- refresh
- setMonitorWindow
- useDefaultMonitorWindow
- canRedo
- canUndo
- redo
- undo
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

IMMSequencer

Features not defined for Windows:

- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

Portability

IMMWaveAudio

Features not defined for Windows:

- cutCopyBufferSize
- copyFromBuffer
- copyToBuffer
- cutToBuffer
- pasteFromBuffer
- pasteToBuffer
- supportsWaveFormat
- canRedo
- canUndo
- redo
- undo
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

Chapter 4. Part List

The part list table provides an alphabetical list of parts. For each part, the following information is provided:

- **Name**. Abstract part names are shown in italics.
- **Page**. For documented parts, the first page of the part chapter is given.
- **Header file**. This is the C++ header file for the part.
- **Part file**. This is the part file that contains the part definition. You need to load this part file before you can use the part. If no part file name is given, the part is in the vbbase.vbb part file. Visual Builder automatically loads the vbbase.vbb part file for you.

Name	Page	Header file	Part file
floatSamples	553	float.h	vbsample
I0String	59	i0string.hpp	
I3StateCheckBox	61	i3statbx.hpp	
IABag		iabag.h	vbcc
IAccelerator	67	iaccel.hpp	
IAcceleratorKey	69	iaccelky.hpp	
IAcceleratorTable	71	iacceltb.hpp	
IACollection		iacllct.h	
IAddress	555	iadd.hpp	vbsample
IADeque		iadeque.h	vbcc
IAEqualityCollection		iaequal.h	vbcc
IAEqualityKeyCollection		iaeqkey.h	vbcc
IAEqualityKeySortedCollection		iaeqksrt.h	vbcc
IAEqualitySequence		iaeqseq.h	vbcc
IAEqualitySortedCollection		iaeqsrt.h	vbcc
IAHeap		iaheap.h	vbcc
IAKeyBag		iakeybag.h	vbcc
IAKeyCollection		iakey.h	vbcc
IAKeySet		iakeyset.h	vbcc
IAKeySortedBag		iaksbag.h	vbcc
IAKeySortedCollection		iaksrt.h	vbcc
IAKeySortedSet		iaksset.h	vbcc
IAMap		iamap.h	vbcc
IAnimatedButton	73	ianimbut.hpp	
IAOrderedCollection		iaorder.h	
IApplication		iapp.hpp	
IAPriorityQueue		iaprioqu.h	vbcc
IAQueue		iaqueue.h	vbcc
IARelation		iarel.h	vbcc

© Copyright IBM Corp. 1995, 1996

Part List

Name	Page	Header file	Part file
IASequence		iaseq.h	
IASequentialCollection		iasqntl.h	
IASet		iaset.h	vbcc
IASortedBag		iasrtbag.h	vbcc
IASortedCollection		iasrt.h	vbcc
IASortedMap		iasrtmap.h	vbcc
IASortedRelation		iasrtrel.h	vbcc
IASortedSet		iasrtset.h	vbcc
IAStack		iastack.h	vbcc
IBase		ibase.hpp	
IBaseComboBox		icombobs.hpp	
IBaseListBox		ilistbas.hpp	
IBaseSpinButton		ispinbas.hpp	
IBitFlag		ibitflag.hpp	
IBitmapControl	79	ibmpctl.hpp	
IButton		ibutton.hpp	
ICanvas	83	icanvas.hpp	
ICheckBox	89	icheckbx.hpp	
ICircularSlider	95	icslider.hpp	
ICLibErrorInfo	101	iexcept.hpp	
IClipboard	103	iclipbrd.hpp	
ICnrEditHandler		icnrehdr.hpp	
ICollectionViewComboBox	105	icombovw.hpp	
ICollectionViewListBox	113	ilistcvw.hpp	
IColor	121	icolor.hpp	
IComboBox	123	icombobx.hpp	
ICompany	557	icompany.hpp	vbsample
IContainerColumn	131	icnrcol.hpp	
IContainerControl		icnrctl.hpp	
IContainerObject	135	icnrobj.hpp	
IControl		icontrol.hpp	
ICurrentApplication		iapp.hpp	
ICurrentThread		ithread.hpp	
ICustomButton	137	icustbut.hpp	
ICustomer	559	icust.hpp	vbsample
IDate	143	idate.hpp	
IDrawingCanvas	145	idrawcv.hpp	
IDynamicLinkLibrary	151	ireslib.hpp	
IElemPointer	411	iptr.h	vbcc
IEntryField	153	ientryfd.hpp	
IErrorInfo		iexcept.hpp	
IFileDialog		ifiledlg.hpp	
IFlyOverHelpHandler	159	iflyhhdr.hpp	
IFlyText	161	iflytext.hpp	
IFont	165	ifont.hpp	
IFontDialog		ifontdlg.hpp	
IFrameWindow	167	iframe.hpp	

Part List

Name	Page	Header file	Part file
IGraphicPushButton	175	igraphbt.hpp	
IGroupBox	181	igroupbx.hpp	
IGUIErrorInfo	187	iexcept.hpp	
IHandle		ihandle.hpp	
IHandler		ihandler.hpp	
IHelpWindow	189	ihelp.hpp	
IIconControl	195	iiconctl.hpp	
IInfoArea		iinfoa.hpp	
IListBox	199	ilistbox.hpp	
IMenu	205	imenu.hpp	
IMenuBar		imenubar.hpp	
IMenuCascade	211	imnitem.hpp	
IMenuHandler		imenuhdr.hpp	
IMenuItem	215	imnitem.hpp	
IMenuSeparator	219	imnitem.hpp	
IMessageBox	221	imsgbox.hpp	
IMM24FramesPerSecondTime	489	immtime.hpp	vbmm
IMM25FramesPerSecondTime	491	immtime.hpp	vbmm
IMM30FramesPerSecondTime	493	immtime.hpp	vbmm
IMMAmpMixer	495	immamix.hpp	vbmm
IMMAudioBuffer	499	immabuf.hpp	vbmm
IMMAudioCD	501	immcdda.hpp	vbmm
IMMAudioCDContents	507	immcdda.hpp	vbmm
IMMConfigurableAudio		immaud.hpp	vbmm
IMMDevice		immdev.hpp	vbmm
IMMDigitalVideo	509	immdigvd.hpp	vbmm
IMMErrorInfo	515	immexcpt.hpp	vbmm
IMMFileMedia		immfilem.hpp	vbmm
IMMHourMinSecFrameTime	517	immtime.hpp	vbmm
IMMHourMinSecTime	519	immtime.hpp	vbmm
IMMMasterAudio	521	immmaud.hpp	vbmm
IMMMillisecondTime	525	immtime.hpp	vbmm
IMMMinSecFrameTime	527	immtime.hpp	vbmm
IMMPlayableDevice		immplayd.hpp	vbmm
IMMPlayerPanel	529	immplypn.hpp	vbmm
IMMRecordable		immrecrd.hpp	vbmm
IMMRemovableMedia		immremed.hpp	vbmm
IMMSequencer	533	immsequ.hpp	vbmm
IMMSpeed	537	immspeed.hpp	vbmm
IMMTime	539	immtime.hpp	vbmm
IMMTrackMinSecFrameTime	541	immttime.hpp	vbmm
IMMWaveAudio	543	immwave.hpp	vbmm
IMngPointer	413	iptr.h	vbcc
IMultiCellCanvas	225	imcelcv.hpp	
IMultiLineEdit	231	imle.hpp	
INotebook	237	inotebk.hpp	
INotifier		inotify.hpp	

Part List

Name	Page	Header file	Part file
INumericSpinButton	243	ispinnum.hpp	
IObserver		iobservr.hpp	
IOrderedRecord	561	iordrrec.hpp	vbsample
ioSamples	563	io.h	vbsample
IOutlineBox	249	ioutlbox.hpp	
IPair	253	ipoint.hpp	
IPartOrderedCollection		ipartccl.h	vbcc
IPoint	255	ipoint.hpp	
IPointArray	257	iptarray.hpp	
IPopUpMenu		ipopmenu.hpp	
IProfile	259	iprofile.hpp	
IProgressIndicator	261	islider.hpp	
IPushButton	267	ipushbut.hpp	
IRadioButton	273	iradiobt.hpp	
IRange	279	ipoint.hpp	
IRBag		irbag.h	vbcc
IRDeque		irdeque.h	vbcc
IRecord	565	irecord.hpp	vbsample
IRectangle	281	irect.hpp	
IRefCounted		irefcnt.hpp	
IREqualitySequence		ireqseq.h	vbcc
IResource		ireslock.hpp	
IResourceId	283	ireslib.hpp	
IResourceLibrary	285	ireslib.hpp	
IRHeap		irheap.h	vbcc
IRKeyBag		irkeybag.h	vbcc
IRKeySet		irkeyset.h	vbcc
IRKeySortedBag		irksbag.h	vbcc
IRKeySortedSet		irksset.h	vbcc
IRMap		irmap.h	vbcc
IRPriorityQueue		irprioqu.h	vbcc
IRQueue		irqueue.h	vbcc
IRRelation		irrel.h	vbcc
IRSequence		irseq.h	
IRSet		irset.h	vbcc
IRSortedBag		irsrtbag.h	vbcc
IRSortedMap		irsrtmap.h	vbcc
IRSortedRelation		irsrtrel.h	vbcc
IRSortedSet		irsrtset.h	vbcc
IRStack		irstack.h	vbcc
IScrollBar	287	iscroll.hpp	
ISetCanvas	293	isetcv.hpp	
ISettingButton		isetbut.hpp	
ISize	299	ipoint.hpp	
ISlider	301	islider.hpp	
ISplitCanvas	307	isplitcv.hpp	
IStandardNotifier	313	istdntfy.hpp	

Part List

Name	Page	Header file	Part file
IStaticText	315	istattxt.hpp	
IString	321	istring.hpp	
IStringGenerator	325	istrgen.hpp	
ISubmenu		isubmenu.hpp	
ISystemErrorInfo	327	iexcept.hpp	
ITextControl		itextctl.hpp	
ITextSpinButton	329	ispintxt.hpp	
ITime	335	itime.hpp	
ITitle	337	ititle.hpp	
IToolBar	343	itbar.hpp	
IToolBarButton	349	itbarbut.hpp	
ITrace	355	itrace.hpp	
IVAvlKeySortedSet		ivksset.h	vbcc
IVBag	415	ivbag.h	vbcc
IVBagOnBase		ivbag.h	vbcc
IVBagOnBSTKeySortedSet	417	ivbag.h	vbcc
IVBagOnHashKeySet	419	ivbag.h	vbcc
IVBagOnSortedDilutedSequence	421	ivbag.h	vbcc
IVBagOnSortedLinkedSequence	423	ivbag.h	vbcc
IVBagOnSortedTabularSequence	425	ivbag.h	vbcc
IVBase		ivbase.hpp	
IVBBooleanPart	567	ivbbool.hpp	vbsample
IVBCheckMenuHandler	357	ivbmenuh.hpp	
IVBCnrEditHandler		ivbcehdr.hpp	
IVBContainerControl	359	ivbcnr.h	
IVBDataTypePart		ivbdtype.hpp	vbsample
IVBDoublePart	569	ivbdbl.hpp	vbsample
IVBDragDropHandler	369	ivbdragh.hpp	
IVBFactory	371		
IVBFileDialog	373	ivbfiled.hpp	
IVBFlyText	377	ivbfly.hpp	
IVBFontDialog	381	ivbfontd.hpp	
IVBInfoArea	385	ivbinfoa.hpp	
IVBLogicalAndPart	571	ivbland.hpp	vbsample
IVBLogicalOrPart	573	ivblor.hpp	vbsample
IVBLongPart	575	ivblong.hpp	vbsample
IVBMinSizeViewPortHandler		ivbvpmsz.hpp	
IVBNotebookPage	391		
IVBShortPart	577	ivbshort.hpp	vbsample
IVBSTKeySortedSet		ivksset.h	vbcc
IVBStringPart	579	ivbstrng.hpp	vbsample
IVBUnsignedLongPart	581	ivbulong.hpp	vbsample
IVBUnsignedShortPart	583	ivbushrt.hpp	vbsample
IVBVariable	395		
IVDeque	427	ivdeque.h	vbcc
IVDequeOnBase		ivdeque.h	vbcc
IVDequeOnDilutedSequence	429	ivdeque.h	vbcc

Part List

Name	Page	Header file	Part file
IVDilutedSequence	431	ivseq.h	vbcc
IVEqualitySequence	433	iveqseq.h	vbcc
IVEqualitySequenceOnBase		iveqseq.h	vbcc
IVEqualitySequenceOnDilutedSequence	435	iveqseq.h	vbcc
IVEqualitySequenceOnTabularSequence	437	iveqseq.h	vbcc
IVGAvlKeySortedSet		ivksset.h	vbcc
IVGBag		ivbag.h	vbcc
IVGBagOnBSTKeySortedSet		ivbag.h	vbcc
IVGBagOnHashKeySet		ivbag.h	vbcc
IVGBagOnSortedDilutedSequence		ivbag.h	vbcc
IVGBagOnSortedLinkedSequence		ivbag.h	vbcc
IVGBagOnSortedTabularSequence		ivbag.h	vbcc
IVGBSTKeySortedSet		ivksset.h	vbcc
IVGDeque		ivdeque.h	vbcc
IVGDequeOnDilutedSequence		ivdeque.h	vbcc
IVGDilutedSequence		ivseq.h	vbcc
IVGEqualitySequence		iveqseq.h	vbcc
IVGEqualitySequenceOnDilutedSequence		iveqseq.h	vbcc
IVGEqualitySequenceOnTabularSequence		iveqseq.h	vbcc
IVGHashKeyBag		ivkeybag.h	vbcc
IVGHashKeySet		ivkeyset.h	vbcc
IVGHeap		ivheap.h	vbcc
IVGHeapOnDilutedSequence		ivheap.h	vbcc
IVGKeyBag		ivkeybag.h	vbcc
IVGKeySet		ivkeyset.h	vbcc
IVGKeySetOnBSTKeySortedSet		ivkeyset.h	vbcc
IVGKeySetOnSortedDilutedSequence		ivkeyset.h	vbcc
IVGKeySetOnSortedLinkedSequence		ivkeyset.h	vbcc
IVGKeySetOnSortedTabularSequence		ivkeyset.h	vbcc
IVGKeySortedBag		ivksbag.h	vbcc
IVGKeySortedBagOnSortedDilutedSequence		ivksbag.h	vbcc
IVGKeySortedBagOnSortedTabularSequence		ivksbag.h	vbcc
IVGKeySortedSet		ivksset.h	vbcc
IVGKeySortedSetOnSortedDilutedSequence		ivksset.h	vbcc
IVGKeySortedSetOnSortedLinkedSequence		ivksset.h	vbcc
IVGKeySortedSetOnSortedTabularSequence		ivksset.h	vbcc
IVGLinkedSequence		ivseq.h	vbcc
IVGMap		ivmap.h	vbcc
IVGMapOnBSTKeySortedMap		ivsrtmap.h	vbcc
IVGMapOnBSTKeySortedSet		ivmap.h	vbcc
IVGMapOnHashKeySet		ivmap.h	vbcc
IVGMapOnSortedDilutedSequence		ivmap.h	vbcc
IVGMapOnSortedLinkedSequence		ivmap.h	vbcc
IVGMapOnSortedTabularSequence		ivmap.h	vbcc
IVGPriorityQueue		ivprioqu.h	vbcc
IVGQueue		ivqueue.h	vbcc
IVGQueueOnDilutedSequence		ivqueue.h	vbcc

Part List

Name	Page	Header file	Part file
IVGQueueOnTabularSequence		ivqueue.h	vbcc
IVGRelation		ivrel.h	vbcc
IVGSequence		ivseq.h	
IVGSet		ivset.h	vbcc
IVGSetOnBSTKeySortedSet		ivset.h	vbcc
IVGSetOnHashKeySet		ivset.h	vbcc
IVGSetOnSortedDilutedSequence		ivset.h	vbcc
IVGSetOnSortedLinkedSequence		ivset.h	vbcc
IVGSetOnSortedTabularSequence		ivset.h	vbcc
IVGSortedBag		ivsrtbag.h	vbcc
IVGSortedBagOnSortedDilutedSequence		ivsrtbag.h	vbcc
IVGSortedBagOnSortedLinkedSequence		ivsrtbag.h	vbcc
IVGSortedBagOnSortedTabularSequence		ivsrtbag.h	vbcc
IVGSortedMap		ivsrtmap.h	vbcc
IVGSortedMapOnSortedDilutedSequence		ivsrtmap.h	vbcc
IVGSortedMapOnSortedLinkedSequence		ivsrtmap.h	vbcc
IVGSortedMapOnSortedTabularSequence		ivsrtmap.h	vbcc
IVGSortedRelation		ivsrtrel.h	vbcc
IVGSortedRelationOnSortedDilutedSequence		ivsrtrel.h	vbcc
IVGSortedRelationOnSortedTabularSequence		ivsrtrel.h	vbcc
IVGSortedSet		ivsrtset.h	vbcc
IVGSortedSetOnBSTKeySortedSet		ivsrtset.h	vbcc
IVGSortedSetOnSortedDilutedSequence		ivsrtset.h	vbcc
IVGSortedSetOnSortedLinkedSequence		ivsrtset.h	vbcc
IVGSortedSetOnSortedTabularSequence		ivsrtset.h	vbcc
IVGStack		ivstack.h	vbcc
IVGStackOnDilutedSequence		ivstack.h	vbcc
IVGStackOnTabularSequence		ivstack.h	vbcc
IVGTabularSequence		ivseq.h	vbcc
IVHashKeyBag		ivkeybag.h	vbcc
IVHashKeySet		ivkeyset.h	vbcc
IVHeap	439	ivheap.h	vbcc
IVHeapOnBase		ivheap.h	vbcc
IVHeapOnDilutedSequence	441	ivheap.h	vbcc
IViewPort	399	ivport.hpp	
IVKeyBag		ivkeybag.h	vbcc
IVKeyBagOnBase		ivkeybag.h	vbcc
IVKeySet		ivkeyset.h	vbcc
IVKeySetOnBase		ivkeyset.h	vbcc
IVKeySetOnBSTKeySortedSet		ivkeyset.h	vbcc
IVKeySetOnSortedDilutedSequence		ivkeyset.h	vbcc
IVKeySetOnSortedLinkedSequence		ivkeyset.h	vbcc
IVKeySetOnSortedTabularSequence		ivkeyset.h	vbcc
IVKeySortedBag		ivksbag.h	vbcc
IVKeySortedBagOnBase		ivksbag.h	vbcc
IVKeySortedBagOnSortedDilutedSequence		ivksbag.h	vbcc
IVKeySortedBagOnSortedTabularSequence		ivksbag.h	vbcc

Part List

Name	Page	Header file	Part file
IVKeySortedSet		ivksset.h	vbcc
IVKeySortedSetOnBase		ivksset.h	vbcc
IVKeySortedSetOnSortedLinkedSequence		ivksset.h	vbcc
IVKeySortedSetOnSortedTabularSequence		ivksset.h	vbcc
IVLinkedSequence	443	ivseq.h	vbcc
IVMap		ivmap.h	vbcc
IVMapOnBase		ivmap.h	vbcc
IVMapOnBSTKeySortedSet		ivmap.h	vbcc
IVMapOnHashKeySet		ivmap.h	vbcc
IVMapOnSortedDilutedSequence		ivmap.h	vbcc
IVMapOnSortedLinkedSequence		ivmap.h	vbcc
IVMapOnSortedTabularSequence		ivmap.h	vbcc
IVPriorityQueue		ivprioqu.h	vbcc
IVPriorityQueueOnBase		ivprioqu.h	vbcc
IVQueue	445	ivqueue.h	vbcc
IVQueueOnBase		ivqueue.h	vbcc
IVQueueOnDilutedSequence	447	ivqueue.h	vbcc
IVQueueOnTabularSequence	449	ivqueue.h	vbcc
IVRelation		ivrel.h	vbcc
IVRelationOnBase		ivrel.h	vbcc
IVSequence	403	ivseq.h	
IVSequenceOnBase		ivseq.h	
IVSet	451	ivset.h	vbcc
IVSetOnBase		ivset.h	vbcc
IVSetOnBSTKeySortedSet	453	ivset.h	vbcc
IVSetOnHashKeySet	455	ivset.h	vbcc
IVSetOnSortedDilutedSequence	457	ivset.h	vbcc
IVSetOnSortedLinkedSequence	459	ivset.h	vbcc
IVSetOnSortedTabularSequence	461	ivset.h	vbcc
IVSortedBag	463	ivsrtbag.h	vbcc
IVSortedBagOnBase		ivsrtbag.h	vbcc
IVSortedBagOnSortedDilutedSequence	465	ivsrtbag.h	vbcc
IVSortedBagOnSortedLinkedSequence	467	ivsrtbag.h	vbcc
IVSortedBagOnSortedTabularSequence	469	ivsrtbag.h	vbcc
IVSortedMap		ivsrtmap.h	vbcc
IVSortedMapOnBase		ivsrtmap.h	vbcc
IVSortedMapOnSortedDilutedSequence		ivsrtmap.h	vbcc
IVSortedMapOnSortedLinkedSequence		ivsrtmap.h	vbcc
IVSortedMapOnSortedTabularSequence		ivsrtmap.h	vbcc
IVSortedRelation		ivsrtrel.h	vbcc
IVSortedRelationOnBase		ivsrtrel.h	vbcc
IVSortedRelationOnSortedDilutedSequence		ivsrtrel.h	vbcc
IVSortedRelationOnSortedTabularSequence		ivsrtrel.h	vbcc
IVSortedSet	471	ivsrtset.h	vbcc
IVSortedSetOnBase		ivsrtset.h	vbcc
IVSortedSetOnBSTKeySortedSet	473	ivsrtset.h	vbcc
IVSortedSetOnSortedLinkedSequence	475	ivsrtset.h	vbcc

Part List

Name	Page	Header file	Part file
IVSortedSetOnSortedTabularSequence	477	ivsrtset.h	vbcc
IVStack	479	ivstack.h	vbcc
IVStackOnBase		ivstack.h	vbcc
IVStackOnTabularSequence	481	ivstack.h	vbcc
IVTabularSequence	483	ivseq.h	vbcc
IWindow		iwindow.hpp	
mathSamples	585	math.h	vbsample
staticWindowSamples	587	iwindow.hpp	vbsample
stdioSamples	589	stdio.h	vbsample
stdlibSamples	591	stdlib.h	vbsample

Part List

Part 2. Parts in vbbase.vbb

The following chapters provide information on Visual Builder parts that you can use in your applications. These parts are defined in the vbbase.vbb part file. Visual Builder automatically loads the vbbase.vbb part file for you.

Chapter 5. I0String . . . 59
Preferred Features — I0String . . . 59
Features — I0String . . . 59

Chapter 6. I3StateCheckBox . . . 61
Portability — I3StateCheckBox . . . 61
Related Parts — I3StateCheckBox . . . 62
Building Guidelines — I3StateCheckBox . . . 62
Preferred Features — I3StateCheckBox . . . 64
Features — I3StateCheckBox . . . 64

Chapter 7. IAccelerator . . . 67
Portability — IAccelerator . . . 67
Preferred Features — IAccelerator . . . 67
Features — IAccelerator . . . 67

Chapter 8. IAcceleratorKey . . . 69
Preferred Features — IAcceleratorKey . . . 69
Features — IAcceleratorKey . . . 69

Chapter 9. IAcceleratorTable . . . 71
Preferred Features — IAcceleratorTable . . . 71
Features — IAcceleratorTable . . . 71

Chapter 10. IAnimatedButton . . . 73
Portability — IAnimatedButton . . . 73
Related Parts — IAnimatedButton . . . 74
Building Guidelines — IAnimatedButton . . . 74
Preferred Features — IAnimatedButton . . . 77
Features — IAnimatedButton . . . 77

Chapter 11. IBitmapControl . . . 79
Portability — IBitmapControl . . . 79
Building Guidelines — IBitmapControl . . . 80
Preferred Features — IBitmapControl . . . 81

© Copyright IBM Corp. 1995, 1996

Parts in vbbase.vbb

Features — IBitmapControl 81

Chapter 12. ICanvas 83
Portability — ICanvas 84
Related Parts — ICanvas 84
Building Guidelines — ICanvas 85
Preferred Features — ICanvas 86
Features — ICanvas 86

Chapter 13. ICheckBox 89
Portability — ICheckBox 89
Related Parts — ICheckBox 90
Building Guidelines — ICheckBox 90
Preferred Features — ICheckBox 92
Features — ICheckBox 92

Chapter 14. ICircularSlider 95
Portability — ICircularSlider 95
Related Parts — ICircularSlider 96
Building Guidelines — ICircularSlider 97
Preferred Features — ICircularSlider 98
Features — ICircularSlider 98

Chapter 15. ICLibErrorInfo 101
Preferred Features — ICLibErrorInfo 101
Features — ICLibErrorInfo 101

Chapter 16. IClipboard 103
Preferred Features — IClipboard 103
Features — IClipboard 103

Chapter 17. ICollectionViewComboBox 105
Portability — ICollectionViewComboBox 106
Related Parts — ICollectionViewComboBox 107
Building Guidelines — ICollectionViewComboBox 107
Preferred Features — ICollectionViewComboBox 110
Features — ICollectionViewComboBox 111

Chapter 18. ICollectionViewListBox 113
Portability — ICollectionViewListBox 113
Related Parts — ICollectionViewListBox 114
Building Guidelines — ICollectionViewListBox 115
Preferred Features — ICollectionViewListBox 117

Features — ICollectionViewListBox 118

Chapter 19. IColor 121
Preferred Features — IColor 121
Features — IColor 121

Chapter 20. IComboBox 123
Portability — IComboBox 124
Related Parts — IComboBox 125
Building Guidelines — IComboBox 125
Preferred Features — IComboBox 127
Features — IComboBox 127

Chapter 21. IContainerColumn 131
Related Parts — IContainerColumn 131
Building Guidelines — IContainerColumn 132
Preferred Features — IContainerColumn 133
Features — IContainerColumn 134

Chapter 22. IContainerObject 135
Related Parts — IContainerObject 135
Preferred Features — IContainerObject 135
Features — IContainerObject 135

Chapter 23. ICustomButton 137
Portability — ICustomButton 137
Related Parts — ICustomButton 138
Building Guidelines — ICustomButton 138
Preferred Features — ICustomButton 140
Features — ICustomButton 140

Chapter 24. IDate 143
Portability — IDate 143
Preferred Features — IDate 143
Features — IDate 143

Chapter 25. IDrawingCanvas 145
Portability — IDrawingCanvas 145
Related Parts — IDrawingCanvas 146
Building Guidelines — IDrawingCanvas 147
Preferred Features — IDrawingCanvas 148
Features — IDrawingCanvas 148

Parts in vbbase.vbb

Chapter 26. IDynamicLinkLibrary 151
Preferred Features — IDynamicLinkLibrary 151
Features — IDynamicLinkLibrary 151

Chapter 27. IEntryField 153
Portability — IEntryField 153
Related Parts — IEntryField 154
Building Guidelines — IEntryField 154
Preferred Features — IEntryField 156
Features — IEntryField 156

Chapter 28. IFlyOverHelpHandler 159
Preferred Features — IFlyOverHelpHandler 159
Features — IFlyOverHelpHandler 159

Chapter 29. IFlyText 161
Portability — IFlyText 161
Preferred Features — IFlyText 162
Features — IFlyText 162

Chapter 30. IFont 165
Portability — IFont 165
Preferred Features — IFont 165
Features — IFont 166

Chapter 31. IFrameWindow 167
Portability — IFrameWindow 168
Related Parts — IFrameWindow 169
Building Guidelines — IFrameWindow 169
Preferred Features — IFrameWindow 172
Features — IFrameWindow 173

Chapter 32. IGraphicPushButton 175
Portability — IGraphicPushButton 175
Related Parts — IGraphicPushButton 176
Building Guidelines — IGraphicPushButton 176
Preferred Features — IGraphicPushButton 177
Features — IGraphicPushButton 178

Chapter 33. IGroupBox 181
Portability — IGroupBox 181
Related Parts — IGroupBox 182
Building Guidelines — IGroupBox 182

Preferred Features — IGroupBox 183
Features — IGroupBox 183

Chapter 34. IGUIErrorInfo 187
Preferred Features — IGUIErrorInfo 187
Features — IGUIErrorInfo 187

Chapter 35. IHelpWindow 189
Portability — IHelpWindow 189
Related Parts — IHelpWindow 190
Building Guidelines — IHelpWindow 190
Preferred Features — IHelpWindow 191
Features — IHelpWindow 192

Chapter 36. IIconControl 195
Portability — IIconControl 195
Building Guidelines — IIconControl 196
Preferred Features — IIconControl 196
Features — IIconControl 196

Chapter 37. IListBox 199
Portability — IListBox 199
Related Parts — IListBox 200
Building Guidelines — IListBox 201
Preferred Features — IListBox 202
Features — IListBox 203

Chapter 38. IMenu 205
Portability — IMenu 206
Related Parts — IMenu 206
Building Guidelines — IMenu 207
Preferred Features — IMenu 208
Features — IMenu 208

Chapter 39. IMenuCascade 211
Portability — IMenuCascade 211
Related Parts — IMenuCascade 211
Building Guidelines — IMenuCascade 211
Preferred Features — IMenuCascade 213
Features — IMenuCascade 213

Chapter 40. IMenuItem 215
Portability — IMenuItem 215

Parts in vbbase.vbb

Related Parts — IMenuItem 215
Building Guidelines — IMenuItem 215
Preferred Features — IMenuItem 217
Features — IMenuItem 218

Chapter 41. IMenuSeparator 219
Portability — IMenuSeparator 219
Related Parts — IMenuSeparator 219
Building Guidelines — IMenuSeparator 219
Preferred Features — IMenuSeparator 220
Features — IMenuSeparator 220

Chapter 42. IMessageBox 221
Related Parts — IMessageBox 221
Building Guidelines — IMessageBox 221
Preferred Features — IMessageBox 223
Features — IMessageBox 223

Chapter 43. IMultiCellCanvas 225
Portability — IMultiCellCanvas 226
Related Parts — IMultiCellCanvas 226
Building Guidelines — IMultiCellCanvas 227
Preferred Features — IMultiCellCanvas 229
Features — IMultiCellCanvas 229

Chapter 44. IMultiLineEdit 231
Portability — IMultiLineEdit 231
Related Parts — IMultiLineEdit 232
Building Guidelines — IMultiLineEdit 232
Preferred Features — IMultiLineEdit 234
Features — IMultiLineEdit 234

Chapter 45. INotebook 237
Portability — INotebook 237
Related Parts — INotebook 238
Building Guidelines — INotebook 239
Preferred Features — INotebook 240
Features — INotebook 241

Chapter 46. INumericSpinButton 243
Portability — INumericSpinButton 243
Related Parts — INumericSpinButton 244
Building Guidelines — INumericSpinButton 245

Preferred Features — INumericSpinButton 246
Features — INumericSpinButton 246

Chapter 47. IOutlineBox 249
Portability — IOutlineBox 249
Related Parts — IOutlineBox 250
Building Guidelines — IOutlineBox 250
Preferred Features — IOutlineBox 251
Features — IOutlineBox 251

Chapter 48. IPair 253
Preferred Features — IPair 253
Features — IPair 253

Chapter 49. IPoint 255
Preferred Features — IPoint 255
Features — IPoint 255

Chapter 50. IPointArray 257
Preferred Features — IPointArray 257
Features — IPointArray 257

Chapter 51. IProfile 259
Portability — IProfile 259
Preferred Features — IProfile 259
Features — IProfile 259

Chapter 52. IProgressIndicator 261
Portability — IProgressIndicator 261
Related Parts — IProgressIndicator 262
Building Guidelines — IProgressIndicator 262
Preferred Features — IProgressIndicator 264
Features — IProgressIndicator 265

Chapter 53. IPushButton 267
Portability — IPushButton 267
Related Parts — IPushButton 268
Building Guidelines — IPushButton 268
Preferred Features — IPushButton 270
Features — IPushButton 270

Chapter 54. IRadioButton 273
Portability — IRadioButton 273

Parts in vbbase.vbb

Related Parts — IRadioButton 274
Building Guidelines — IRadioButton 275
Preferred Features — IRadioButton 277
Features — IRadioButton 277

Chapter 55. IRange 279
Preferred Features — IRange 279
Features — IRange 279

Chapter 56. IRectangle 281
Preferred Features — IRectangle 281
Features — IRectangle 282

Chapter 57. IResourceId 283
Preferred Features — IResourceId 283
Features — IResourceId 283

Chapter 58. IResourceLibrary 285
Preferred Features — IResourceLibrary 285
Features — IResourceLibrary 285

Chapter 59. IScrollBar 287
Portability — IScrollBar 287
Related Parts — IScrollBar 288
Building Guidelines — IScrollBar 289
Preferred Features — IScrollBar 290
Features — IScrollBar 290

Chapter 60. ISetCanvas 293
Portability — ISetCanvas 294
Related Parts — ISetCanvas 294
Building Guidelines — ISetCanvas 295
Preferred Features — ISetCanvas 296
Features — ISetCanvas 297

Chapter 61. ISize 299
Preferred Features — ISize 299
Features — ISize 299

Chapter 62. ISlider 301
Portability — ISlider 301
Related Parts — ISlider 302
Building Guidelines — ISlider 302

Preferred Features — ISlider . . . 305
Features — ISlider . . . 305

Chapter 63. ISplitCanvas . . . 307
Portability — ISplitCanvas . . . 308
Related Parts — ISplitCanvas . . . 308
Building Guidelines — ISplitCanvas . . . 309
Preferred Features — ISplitCanvas . . . 310
Features — ISplitCanvas . . . 310

Chapter 64. IStandardNotifier . . . 313
Preferred Features — IStandardNotifier . . . 313
Features — IStandardNotifier . . . 313

Chapter 65. IStaticText . . . 315
Portability — IStaticText . . . 315
Related Parts — IStaticText . . . 316
Building Guidelines — IStaticText . . . 316
Preferred Features — IStaticText . . . 317
Features — IStaticText . . . 318

Chapter 66. IString . . . 321
Preferred Features — IString . . . 321
Features — IString . . . 321

Chapter 67. IStringGenerator . . . 325
Features — IStringGenerator . . . 325

Chapter 68. ISystemErrorInfo . . . 327
Preferred Features — ISystemErrorInfo . . . 327
Features — ISystemErrorInfo . . . 327

Chapter 69. ITextSpinButton . . . 329
Portability — ITextSpinButton . . . 329
Related Parts — ITextSpinButton . . . 330
Building Guidelines — ITextSpinButton . . . 331
Preferred Features — ITextSpinButton . . . 332
Features — ITextSpinButton . . . 332

Chapter 70. ITime . . . 335
Portability — ITime . . . 335
Preferred Features — ITime . . . 335
Features — ITime . . . 335

Parts in vbbase.vbb

Chapter 71. ITitle 337
Portability — ITitle 337
Related Parts — ITitle 338
Building Guidelines — ITitle 338
Preferred Features — ITitle 339
Features — ITitle 340

Chapter 72. IToolBar 343
Portability — IToolBar 343
Related Parts — IToolBar 344
Building Guidelines — IToolBar 344
Preferred Features — IToolBar 346
Features — IToolBar 346

Chapter 73. IToolBarButton 349
Portability — IToolBarButton 349
Related Parts — IToolBarButton 350
Building Guidelines — IToolBarButton 350
Preferred Features — IToolBarButton 352
Features — IToolBarButton 352

Chapter 74. ITrace 355
Preferred Features — ITrace 355
Features — ITrace 355

Chapter 75. IVBCheckMenuHandler 357
Preferred Features — IVBCheckMenuHandler 357
Features — IVBCheckMenuHandler 357

Chapter 76. IVBContainerControl 359
Portability — IVBContainerControl 360
Related Parts — IVBContainerControl 361
Building Guidelines — IVBContainerControl 361
Preferred Features — IVBContainerControl 365
Features — IVBContainerControl 365

Chapter 77. IVBDragDropHandler 369
Preferred Features — IVBDragDropHandler 369
Features — IVBDragDropHandler 369

Chapter 78. IVBFactory 371
Related Parts — IVBFactory 371
Building Guidelines — IVBFactory 371

Preferred Features — IVBFactory 372
Features — IVBFactory 372

Chapter 79. IVBFileDialog 373
Portability — IVBFileDialog 373
Related Parts — IVBFileDialog 373
Building Guidelines — IVBFileDialog 374
Preferred Features — IVBFileDialog 376
Features — IVBFileDialog 376

Chapter 80. IVBFlyText 377
Portability — IVBFlyText 377
Related Parts — IVBFlyText 378
Building Guidelines — IVBFlyText 378
Preferred Features — IVBFlyText 379
Features — IVBFlyText 379

Chapter 81. IVBFontDialog 381
Portability — IVBFontDialog 381
Related Parts — IVBFontDialog 381
Building Guidelines — IVBFontDialog 382
Preferred Features — IVBFontDialog 383
Features — IVBFontDialog 384

Chapter 82. IVBInfoArea 385
Portability — IVBInfoArea 385
Related Parts — IVBInfoArea 386
Building Guidelines — IVBInfoArea 386
Preferred Features — IVBInfoArea 387
Features — IVBInfoArea 388

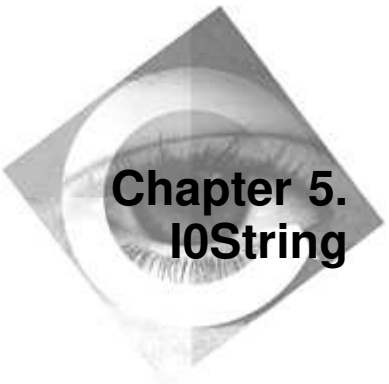
Chapter 83. IVBNotebookPage 391
Related Parts — IVBNotebookPage 391
Building Guidelines — IVBNotebookPage 392
Preferred Features — IVBNotebookPage 393
Features — IVBNotebookPage 393

Chapter 84. IVBVariable 395
Related Parts — IVBVariable 395
Building Guidelines — IVBVariable 395
Preferred Features — IVBVariable 397
Features — IVBVariable 397

Parts in vbbase.vbb

Chapter 85. IViewPort 399
Portability — IViewPort 399
Related Parts — IViewPort 400
Building Guidelines — IViewPort 400
Preferred Features — IViewPort 401
Features — IViewPort 401

Chapter 86. IVSequence 403
Related Parts — IVSequence 403
Building Guidelines — IVSequence 404
Preferred Features — IVSequence 405
Features — IVSequence 405



Chapter 5. I0String

Description: IBM string class, indexed starting at 0
Part type: Class
Part file: vbbase
Header file: i0string.hpp

Derivation:
IBase
IString
I0String

Preferred Features — I0String

copy	Replaces the receiver's contents with a specified number of replications of itself.
remove	Deletes the specified portion of a string from the receiver.
this	A Pointer to the instance.

Features — I0String

Action	Attribute	Type
b2c ■	alphabetic	`Boolean`
b2d ■	alphanumeric	`Boolean`
b2x ■	asDebugInfo	`IString`
c2b	asDouble	`double`
c2d	asInt	`long`
c2x ■	asString	`IString`
center ■	asUnsigned	`unsigned long`
change ■	binaryDigits	`Boolean`
charType ■	contents ■	`IString`
copy ■	control	`Boolean`
d2b ■	digits	`Boolean`
d2c	graphics	`Boolean`
d2x	hexDigits	`Boolean`
includes ■	includesDBCS	`Boolean`
indexOf ■	includesMBCS	`Boolean`
indexOfAnyBut ■	includesSBCS	`Boolean`
indexOfAnyOf ■	isASCII	`Boolean`
indexOfPhrase ■	isDBCS	`Boolean`
indexOfWord ■	isLowerCase	`Boolean`
insert ■	isMBCS	`Boolean`

© Copyright IBM Corp. 1995, 1996

I0String

Action	Attribute	Type
isAbbreviationFor ■	isSBCS	Boolean
isLike ■	isUpperCase	Boolean
lastIndexOf ■	length	unsigned
lastIndexOfAnyBut ■	numWords	unsigned
lastIndexOfAnyOf ■	printable	Boolean
leftJustify ■	punctuation	Boolean
lengthOfWord ■	size	unsigned
lineFrom ■	**this**	I0String*
lowerCase ■	validDBCS	Boolean
occurrencesOf ■	validMBCS	Boolean
operator & ■	whiteSpace	Boolean
operator &= ■		
operator + ■		
operator += ■		
operator = ■		
operator char *		
operator signed char *		
operator unsigned char *		
operator [] ■		
operator ^ ■		
operator ^= ■		
operator | ■		
operator |= ■		
operator ~		
overlayWith ■		
remove ■		
removeWords ■		
reverse ■		
rightJustify ■		
space ■		
strip ■		
stripBlanks ■		
stripLeading ■		
stripLeadingBlanks ■		
stripTrailing ■		
stripTrailingBlanks ■		
subString ■		
translate ■		
upperCase ■		
word ■		
wordIndexOfPhrase ■		
words ■		
x2b		
x2c ■		
x2d ■		

Chapter 6. I3StateCheckBox

Description: IBM 3-state check-box control
Part type: Visual
Part file: vbbase
Header file: i3statbx.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
ISettingButton
I3StateCheckBox

Use an I3StateCheckBox* part to provide a settings choice that has either three distinct states or two distinct states and an undetermined state. A mark in the check box indicates that the choice is selected. Shading in the check box indicates that the choice is undetermined. Use check boxes in a group to provide multiple choices that are not mutually exclusive.

Alternatively, use an ICheckBox* part to provide a settings choice that has two distinct states. Use a group of IRadioButton* parts for a set of mutually exclusive choices.

The user can alternate among the three states by clicking on the check box.

Portability — I3StateCheckBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — I3StateCheckBox

Setting choice parts:

- "I3StateCheckBox" on page 61
- "ICheckBox" on page 89
- "IRadioButton" on page 273

Building Guidelines — I3StateCheckBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Setting the Initial State
- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Disabling Another Part
- Enabling Another Part
- Disabling the Part
- Enabling the Part

Adding the Part

To add an I3StateCheckBox* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.

- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Defining Text for the Part

To set the text for the check box, enter the text you want in the **Text** field on the I3StateCheckBox* part's **General** settings page.

Setting the Initial State

To set the initial state of the check box, set the **Selected** field on the I3StateCheckBox* part's **General** settings page.

To initially select the check box, make sure the **Selected** field check box is checked. If it is not checked, select it.

To initially not select the check box, explicitly deselect the **Selected** field check box. If it is checked, select it to uncheck it. If it is not checked, select it; then select it again to uncheck it.

Enabling Tab Key Access

To allow the user to reach the check box by using a tab key, set the **tabStop** field to **On** on the I3StateCheckBox* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the I3StateCheckBox* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Disabling Another Part

To disable another control when the user selects the check box, connect the I3StateCheckBox* part's *selected* attribute to the *value* attribute of an IVBBooleanPart* part. Then, connect the IVBBooleanPart* part's *notValue* attribute to the *enabled* attribute of the other control. When the user deselects the check box, the other control is enabled.

IVBBooleanPart is available in the vbsample.vbb part file.

Enabling Another Part

To enable another control when the user selects the check box, connect the I3StateCheckBox* part's *selected* attribute to the *enabled* attribute of the other control. When the user deselects the check box, the other control is disabled.

Disabling the Part

To disable the check box when an event occurs, connect the event to the I3StateCheckBox* part's *disable* action.

Enabling the Part

To enable the check box when an event occurs, do the following:

1. Connect the event to the I3StateCheckBox* part's *enable* action.
2. The check box is enabled by default. If you have changed this default, open settings for the connection, select **Set parameters**, and check the **Enabled** setting.

Preferred Features — I3StateCheckBox

disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
menu	Popup menu link.
selected	If the button is selected, `true` is returned.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

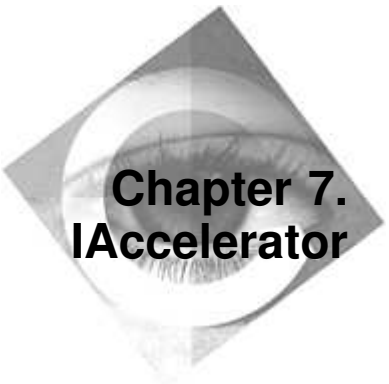
Features — I3StateCheckBox

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
asGUIStyle ■	allowsMouseClickFocus ■	anyEvent
capturePointer ■	asDebugInfo	*backgroundColor*
click	asString	*borderColor*
convertToGUIStyle ■	autoDeleteObject ■	buttonClickEvent
deselect	autoDestroyWindow ■	commandEvent
disable	**autoSelect** ■	*disabledBackgroundColor*
disableAutoSelect	backgroundColor ■ ►	*disabledForegroundColor*
disableGroup	bidiSupported	*enabled*
disableMouseClickFocus	borderColor ■ ►	*focus*
disableNotification	characterSize	*font*
disableTabStop	clipboardHasTextFormat	*foregroundColor*
disableUpdate	defaultPushButton	gotFocusEvent ■

I3StateCheckBox

Action	Attribute	Event
dispatchRemainingHandlers ■	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
enable ■	disabledForegroundColor ■ ►	*hiliteForegroundColor*
enableUpdate ■	displaySize	*inactiveColor*
handleException ■	enabled ■ ►	inputDisabledEvent ■
hide	enabledForNotification ■	inputEnabledEvent ■
hideSourceEmphasis	focus ►	*isDigits*
highlight ■	font ■ ►	lostFocusEvent ■
isLayoutDistorted ■	foregroundColor ■ ►	*position*
matchForMnemonic ■	frameWindow	*selected*
notifyObservers ■	group ■	*shadowColor*
positionBehindSibling ■	**halftone**	*size*
positionBehindSiblings	handle	systemCommandEvent
positionOnSiblings	helpId ■	*text*
postEvent ■	highlighted	*textLength*
refresh ■	hiliteBackgroundColor ■ ►	*valueAsDouble*
releasePointer	hiliteForegroundColor ■ ►	*valueAsInt*
releasePresSpace ■	id ■	*valueAsUnsigned*
resetActiveColor	inactiveColor ■ ►	visibilityDisabledEvent ■
resetBackgroundColor	isDigits ►	visibilityEnabledEvent ■
resetBorderColor	itemProvider ■	*visible*
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
selectHalftone	presSpace	
sendEvent ■	rect ■	
setFocus	selected ■ ►	
setLayoutDistorted ■	shadowColor ■ ►	
show ■	showing	
showSourceEmphasis ■	size ■ ►	
unhighlight	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

I3StateCheckBox



Chapter 7. IAccelerator

Description: IBM class to access shortcut-key table
Part type: Class
Part file: vbbase
Header file: iaccel.hpp

Derivation:
IBase
IAccelerator

IAccelerator* parts support accelerators you define with IMenuItem* and IMenuCascade* parts. You do not need to use these parts directly.

An IAccelerator* part loads an accelerator table from a resource file.

Portability — IAccelerator

Features not defined for Windows:

- unset

Preferred Features — IAccelerator

remove	Removes the accelerator without restoring the one that was previously in effect.
reset	Removes the accelerator and restores the one that was previously in effect.
set	Changes to the new accelerator table.
this	A Pointer to the instance.

Features — IAccelerator

Action	Attribute	Type
remove	asDebugInfo	`IString`
reset	asString	`IString`
set ■	**handle**	`IAccelTblHandle`
	isSet	`Boolean`
	owner	`IWindow*`
	this	`IAccelerator*`

© Copyright IBM Corp. 1995, 1996

IAccelerator

Chapter 8. IAcceleratorKey

Description: IBM accelerator key class
Part type: Class
Part file: vbbase
Header file: iaccelky.hpp

Derivation:
IBase
IAcceleratorKey

IAcceleratorKey* parts support accelerators you define with IMenuItem* and IMenuCascade* parts. You do not need to use these parts directly.

An IAcceleratorKey* part is used with an IAcceleratorTable* part to define an accelerator key.

Preferred Features — IAcceleratorKey

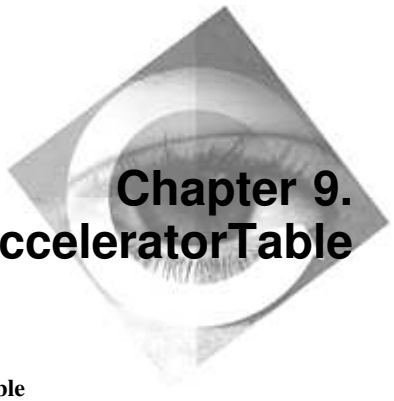
asACCEL	Converts an accelerator key object into a data structure defined by the presentation system for use with its APIs.
character	Returns the data key defined for this accelerator key.
commandId	If the accelerator key runs an application or system command, this function returns the identifier of the command.
setKey	Assigns the specified keystroke to the accelerator key.
this	A Pointer to the instance.

Features — IAcceleratorKey

Action	Attribute	Type
isLike ■	**actionType**	`ICommand::ActionType`
operator != ■	**asACCEL**	`ACCEL`
operator = ■	asDebugInfo	`IString`
operator == ■	asString	`IString`
setHelpKey	**character**	`IString`
setKey ■	**commandId** ■	`ICommand::CommandId`
	keyModifier	`IKey::KeyModifier`
	systemCommand ■	`ICommand::CommandId`
	this	`IAcceleratorKey*`
	virtualKey	`IKey::VirtualKey`

© Copyright IBM Corp. 1995, 1996

IAcceleratorKey



Chapter 9. IAcceleratorTable

Description: IBM accelerator table class
Part type: Class
Part file: vbbase
Header file: iacceltb.hpp

Derivation:
IBase
IAcceleratorTable

IAcceleratorTable* parts support accelerators you define with IMenuItem* and IMenuCascade* parts. You do not need to use these parts directly.

An IAcceleratorTable* part dynamically creates an accelerator table. Accelerator keys can be added or deleted at run time.

Preferred Features — IAcceleratorTable

addKey Adds the specified accelerator key to the table.
addOrReplaceKey
Adds the specified accelerator key to the table, or replaces an existing entry.
keyCount Returns the number of accelerator keys contained in the accelerator table.
removeAllKeys Empties the accelerator table so that it contains no keys.
this A Pointer to the instance.
update Replaces the accelerator keys used by the specified frame window with the entries in the accelerator table object.

Features — IAcceleratorTable

Action	Attribute	Type
addKey ■	asDebugInfo	`IString`
addOrReplaceKey ■	asString	`IString`
containsKeyLike ■	**keyCount**	`unsigned long`
operator = ■	**this**	`IAcceleratorTable*`
removeAllKeys		
removeKeyLike ■		
update ■		

© Copyright IBM Corp. 1995, 1996

IAcceleratorTable

Chapter 10. IAnimatedButton

Description: IBM animated button control
Part type: Visual
Part file: vbbase
Header file: ianimbut.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
ICustomButton
IAnimatedButton

Use an IAnimatedButton* part to give the user an action choice that shows animation. A multimedia play button with an arrow that moves is an example of an animated button.

The user can click the animated button to perform the indicated function. When the action is selected, the graphic on the button is replaced by a cyclic series of graphics. This gives the appearance of animation on the button while the requested action is performed.

Portability — IAnimatedButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IAnimatedButton

Button parts:

- "IAnimatedButton" on page 73
- "ICustomButton" on page 137
- "IGraphicPushButton" on page 175
- "INumericSpinButton" on page 243
- "IPushButton" on page 267
- "IRadioButton" on page 273
- "ITextSpinButton" on page 329
- "IToolBarButton" on page 349

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — IAnimatedButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Assigning Animation Bitmaps
- Setting the Animation Rate
- Enabling Button Latching

- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting and Stopping Animation
- Starting the Requested Action
- Disabling the Part
- Enabling the Part

Adding the Part

To add an IAnimatedButton* part to a composite part, select an IAnimatedButton* part from the parts palette. Then, drop the part on a canvas.

Assigning Animation Bitmaps

To assign animation bitmaps to the button, set the **Bitmap** settings on the IAnimatedButton* part's **General** settings page. You can do either of the following:

- Select a bitmap identifier, such as the play identifier, from the **Identifier list** field.
- Specify a starting bitmap resource and bitmap count by setting the following fields:
  - **DLL name** to identify the DLL containing the bitmaps
  - **Resource ID** to identify the ID of the starting bitmap
  - **Bitmap count** to indicate the number of bitmaps in the series

Setting the Animation Rate

To determine the animation rate for the graphic sequence on the button, set the **Animation rate** field on the IAnimatedButton* part's **General** settings page. Specify how long you want each bitmap displayed, in thousandths of a second.

Enabling Button Latching

To enable latching the button down when it is pressed, set the **latchable** field on the IAnimatedButton* part's **Styles** settings page.

To automatically run animation while the button is latched down, set the **animateWhenLatched** field on the IAnimatedButton* part's **Styles** settings page.

Enabling Tab Key Access

To allow the user to reach the button by using a tab key, set the **tabStop** field to **On** on the IAnimatedButton* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the IAnimatedButton* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Starting and Stopping Animation

If you want animation to start automatically when the button becomes latched down and stop when the button becomes unlatched, you can specify this behavior when you enable latching.

To start animation when an event occurs, connect the event to the IAnimatedButton* part's *startAnimation* action.

To stop animation when an event occurs, do one of the following:

- Connect the event to the IAnimatedButton* part's *stopAnimation* action
- Connect the event to the IAnimatedButton* part's *startAnimation* action and provide a parameter value of false

Starting the Requested Action

To start the action represented by the button, connect the IAnimatedButton* part's *buttonClickEvent* feature to the action on the target part. For example, to start the *play* action of an IMMAudioCD* part, connect the IAnimatedButton* part's *buttonClickEvent* feature to the IMMAudioCD* part's *play* action.

Disabling the Part

To disable the button when an event occurs, connect the event to the IAnimatedButton* part's *disable* action.

Enabling the Part

To enable the button when an event occurs, do the following:

1. Connect the event to the IAnimatedButton* part's *enable* action.
2. The button is enabled by default. If you have changed this default, open settings for the connection, select **Set parameters**, and check the **Enabled** setting.

Preferred Features — IAnimatedButton

bitmap Returns the handle of the specifed bitmap.

buttonClickEvent
Notification identifier provided to observers when the button control is clicked by the user.

disable Prevents keyboard and mouse input from being sent to the window.

enable Enables the window to accept keyboard and mouse input.

enabled If the window is sent mouse and keyboard input, `true` is returned.

latched Returns `true` if the button is in a latched state.

menu Popup menu link.

setFocus Sets the input focus to the window.

text Returns the control window's text.

this A Pointer to the instance.

Features — IAnimatedButton

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	allowsMouseClickFocus ■	anyEvent
click	**animatedWhenLatched** ■	*backgroundColor*
convertToGUIStyle ■	**animationRate** ■	*borderColor*
disable	**animationStarted**	buttonClickEvent
disableAnimateWhenLatched	asDebugInfo	commandEvent
disableAutoLatch	asString	*disabledBackgroundColor*
disableGroup	autoDeleteObject ■	*disabledForegroundColor*
disableLatching	autoDestroyWindow ■	*enabled*
disableMouseClickFocus	autoLatchEnabled ■	*focus*
disableNotification	backgroundColor ■ ►	*font*
disableTabStop	bidiSupported	*foregroundColor*
disableUpdate	**bitmap**	gotFocusEvent ■
dispatchRemainingHandlers ■	**bitmapCount**	*hiliteBackgroundColor*
enable ■	borderColor ■ ►	*hiliteForegroundColor*
enableUpdate ■	characterSize	*inactiveColor*
handleException ■	clipboardHasTextFormat	inputDisabledEvent ■
hide	**currentBitmapIndex** ■	inputEnabledEvent ■
hideSourceEmphasis	defaultPushButton	*isDigits*
highlight ■	disabledBackgroundColor ■ ►	*latched*
isLayoutDistorted ■	disabledForegroundColor ■ ►	lostFocusEvent ■
matchForMnemonic ■	displaySize	*position*
notifyObservers ■	enabled ■ ►	*shadowColor*
positionBehindSibling ■	enabledForNotification ■	*size*
positionBehindSiblings	focus ►	systemCommandEvent
positionOnSiblings	font ■ ►	*text*
postEvent ■	foregroundColor ■ ►	*textLength*
refresh ■	frameWindow	*valueAsDouble*

IAnimatedButton

Action	Attribute	Event
releasePointer	group ■	*valueAsInt*
releasePresSpace ■	handle	*valueAsUnsigned*
resetActiveColor	helpId ■	visibilityDisabledEvent ■
resetBackgroundColor	highlighted	visibilityEnabledEvent ■
resetBorderColor	hiliteBackgroundColor ■ ►	*visible*
resetDisabledBackgroundColor	hiliteForegroundColor ■ ►	
resetDisabledForegroundColor	id ■	
resetFont	inactiveColor ■ ►	
resetForegroundColor	isDigits ►	
resetHiliteBackgroundColor	itemProvider ■	
resetHiliteForegroundColor	latched ■ ►	
resetInactiveColor	latchedBackgroundColor ■	
resetLatchedBackgroundColor	latchedBackgroundColorHalftone	
resetLatchedForegroundColor	latchedForegroundColor ■	
resetMinimumSize	latchingEnabled ■	
resetShadowColor	layoutAdjustment	
sendEvent ■	menu	
setBitmaps ■	minimumSize ■	
setFocus	nativeRect	
setLayoutDistorted ■	owner ■	
show ■	parent ■	
showSourceEmphasis ■	parentSize	
startAnimation ■	pointerCaptured	
stopAnimation	position ■ ►	
unhighlight	presSpace	
unlatch	rect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	userData ■	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

Chapter 11. IBitmapControl

Description: IBM bitmap control
Part type: Visual
Part file: vbbase
Header file: ibmpctl.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IStaticText
IBitmapControl

Use an IBitmapControl* part to provide a picture for graphical application data. Be aware that a bitmap is resized when you place it on either a set canvas or an expandable row or column of a multicell canvas.

Portability — IBitmapControl

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor

© Copyright IBM Corp. 1995, 1996

- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Building Guidelines — IBitmapControl

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Assigning a Bitmap

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Hiding the Bitmap
- Showing the Bitmap

Adding the Part

To add an IBitmapControl* part to a composite part, select an IBitmapControl* part from the parts palette. Then, drop the part on a canvas.

Assigning a Bitmap

To assign a bitmap to the control, set the **DLL name** and **Resource ID** fields on the IBitmapControl* part's **General** settings page.

Hiding the Bitmap

To hide the bitmap when an event occurs, connect the event to the IBitmapControl* part's *disable* action.

Showing the Bitmap

To show the bitmap when an event occurs, do the following:

1. Connect the event to the IBitmapControl* part's *show* action.
2. Open settings for the connection, press the **Set parameters** push button, and check the **visible** setting.

Preferred Features — IBitmapControl

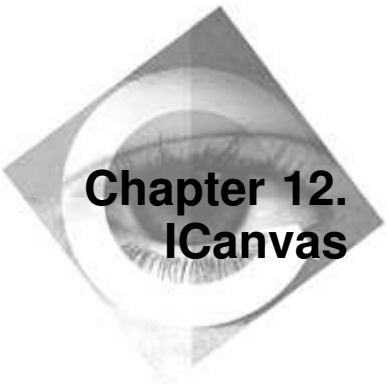
bitmap	Returns the handle to the bitmap.
limit	Returns the number of characters set by setLimit.
menu	Popup menu link.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

Features — IBitmapControl

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	alignment ■	anyEvent
convertToGUIStyle ■	asDebugInfo	*backgroundColor*
disable	asString	*borderColor*
disableFillBackground	autoDeleteObject ■	commandEvent
disableGroup	autoDestroyWindow ■	*disabledBackgroundColor*
disableHalftone	backgroundColor ■ ►	*disabledForegroundColor*
disableNotification	bidiSupported	*enabled*
disableStrikeout	**bitmap** ■	*fillBackground*
disableTabStop	borderColor ■ ►	*fillColor*
disableUnderscore	characterSize	*focus*
disableUpdate	clipboardHasTextFormat	*font*
dispatchRemainingHandlers ■	defaultPushButton	*foregroundColor*
enable ■	disabledBackgroundColor ■ ►	gotFocusEvent ■
enableUpdate ■	disabledForegroundColor ■ ►	*halftone*
handleException ■	displaySize	*hiliteBackgroundColor*
hide	enabled ■ ►	*hiliteForegroundColor*
hideSourceEmphasis	enabledForNotification ■	*inactiveColor*
isLayoutDistorted ■	fillBackground ■ ►	inputDisabledEvent ■
matchForMnemonic ■	fillColor ■ ►	inputEnabledEvent ■
notifyObservers ■	focus ►	*isDigits*
positionBehindSibling ■	font ■ ►	*limit*
positionBehindSiblings	foregroundColor ■ ►	lostFocusEvent ■
positionOnSiblings	frameWindow	*position*
postEvent ■	group ■	*shadowColor*
refresh ■	halftone ■ ►	*size*
releasePointer	handle	*strikeout*
releasePresSpace ■	helpId ■	systemCommandEvent
resetActiveColor	hiliteBackgroundColor ■ ►	*text*
resetBackgroundColor	hiliteForegroundColor ■ ►	*textLength*
resetBorderColor	id ■	*underscore*
resetDisabledBackgroundColor	inactiveColor ■ ►	*valueAsDouble*
resetDisabledForegroundColor	isDigits ►	*valueAsInt*

IBitmapControl

Action	Attribute	Event
resetFillColor	itemProvider ■	*valueAsUnsigned*
resetFont	layoutAdjustment	visibilityDisabledEvent ■
resetForegroundColor	limit ■ ►	visibilityEnabledEvent ■
resetHiliteBackgroundColor	menu	*visible*
resetHiliteForegroundColor	minimumSize ■	
resetInactiveColor	nativeRect	
resetMinimumSize	owner ■	
resetShadowColor	parent ■	
sendEvent ■	parentSize	
setFocus	pointerCaptured	
setLayoutDistorted ■	position ■ ►	
show ■	presSpace	
showSourceEmphasis ■	rect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	strikeout ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	underscore ■ ►	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	



Chapter 12. ICanvas

Description: IBM canvas control
Part type: Visual
Part file: vbbase
Header file: icanvas.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas

Use an ICanvas* part as a composition board for the fixed placement of control parts. Control size and position are not adjusted for different display resolutions or for language translation. If you do not need fixed control placement, we recommend that you use IMultiCellCanvas* and ISetCanvas* parts.

You can use an ICanvas* part in another composer part, such as the following:

- IFrameWindow* client area
- IViewPort*
- IVBNotebookPage*

Then, you can add controls to the canvas within the other composer part.

You can also use a canvas as the base part for a composite part. Then, you can add the canvas with its controls as a subpart in other composite parts.

Alternatively, use one of the following canvas parts:

- IDrawingCanvas* for graphic objects
- IMultiCellCanvas* to provide freely arranged canvas cells
- ISetCanvas* for canvas cells in uniform rows or columns
- ISplitCanvas* for split windows

© Copyright IBM Corp. 1995, 1996

The user interacts with controls placed on a canvas, not with the canvas itself.

Portability — ICanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ICanvas

Canvas parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

Client parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “IMultiLineEdit” on page 231
- “INotebook” on page 237
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — ICanvas

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding Controls
- Adding a Handler for the Part
- Selecting a Font
- Selecting a Color

Adding the Part

To add an ICanvas* part to a composite part, select an ICanvas* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- An IViewPort* part
- An IVBNotebookPage* part
- A canvas part
- The free-form surface

Adding Controls

To add controls to the canvas, select the control parts from the parts palette and drop them on the ICanvas* part.

Position and size the parts as you want them on the canvas. You can use the Composition Editor alignment buttons for positioning and relative sizing of controls on an ICanvas* part. These buttons are applicable only for ICanvas* parts. Parts placed on the ICanvas* part are not automatically sized or moved based on context or content.

Adding a Handler for the Part

To specify a handler for the canvas, do the following:

- Add the handler name and constructor parameters in the **Handler list** field on the ICanvas* part's **Handlers** settings page.
- If the handler is a part that is loaded into Visual Builder, you do not need to specify the handler's header file name. Otherwise, type the header file name for the handler in the Class Editor **Required include files** field.

Selecting a Font

To set the font for all parts on the canvas, select a font on the ICanvas* part's **Font** settings page.

Selecting a Color

To specify a background color for the canvas, select a color from the **Colors** field or set RGB values in the **RGB values** field on the ICanvas* part's **Color** settings page.

Preferred Features — ICanvas

menu	Popup menu link.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

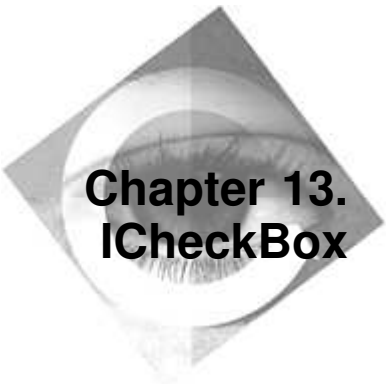
Features — ICanvas

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	defaultPushButton	*font*
enableUpdate ■	disabledBackgroundColor ■ ►	*foregroundColor*
handleException ■	disabledForegroundColor ■ ►	gotFocusEvent ■
hide	enabled ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	focus ►	*inactiveColor*
matchForMnemonic ■	font ■ ►	inputDisabledEvent ■
notifyObservers ■	foregroundColor ■ ►	inputEnabledEvent ■
positionBehindSibling ■	frameWindow	lostFocusEvent ■
positionBehindSiblings	group ■	*position*
positionOnSiblings	handle	*shadowColor*
postEvent ■	helpId ■	*size*
refresh ■	hiliteBackgroundColor ■ ►	systemCommandEvent
releasePointer	hiliteForegroundColor ■ ►	visibilityDisabledEvent ■
releasePresSpace ■	id ■	visibilityEnabledEvent ■
resetActiveColor	inactiveColor ■ ►	*visible*
resetBackgroundColor	itemProvider ■	
resetBorderColor	layoutAdjustment	
resetDisabledBackgroundColor	menu	

ICanvas

Action	Attribute	Event
resetDisabledForegroundColor	minimumSize ■	
resetFont	nativeRect	
resetForegroundColor	**origDefaultButtonHandle**	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	rect ■	
setLayoutDistorted ■	shadowColor ■ ►	
show ■	showing	
showSourceEmphasis ■	size ■ ►	
	tabStop ■	
	this	
	valid	
	visible ■ ►	
	visibleRectangle	

ICanvas



Chapter 13. ICheckBox

Description: IBM 2-state check-box control
Part type: Visual
Part file: vbbase
Header file: icheckbx.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
ISettingButton
ICheckBox

Use an ICheckBox* part to provide a settings choice that has two distinct states, for example, on and off. A mark in the check box indicates that the choice is selected. Use check boxes in a group to provide multiple choices that are not mutually exclusive.

Alternatively, use an I3StateCheckBox* part to provide a settings choice that has either three distinct states or two distinct states and an undetermined state. Use a group of IRadioButton* parts for a set of mutually exclusive choices.

The user can alternate between the two states by clicking on the check box.

Portability — ICheckBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ICheckBox

Setting choice parts:

- "I3StateCheckBox" on page 61
- "ICheckBox" on page 89
- "IRadioButton" on page 273

Building Guidelines — ICheckBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Setting the Initial State
- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Disabling Another Part
- Enabling Another Part
- Disabling the Part
- Enabling the Part

Adding the Part

To add an ICheckBox* part to a composite part, select an ICheckBox* part from the parts palette. Then, drop the part on a canvas.

Defining Text for the Part

To set the text for the check box, enter the text you want in the **Text** field on the ICheckBox* part's **General** settings page.

Setting the Initial State

To set the initial state of the check box, select or deselect the **Selected** field on the ICheckBox* part's **General** settings page.

To initially select the check box, make sure the **Selected** field check box is checked. If it is not checked, select it.

To initially not select the check box, explicitly deselect the **Selected** field check box. If it is checked, select it to uncheck it. If it is not checked, select it; then select it again to uncheck it.

Enabling Tab Key Access

To allow the user to reach the check box by using a tab key, set the **tabStop** field to **On** on the ICheckBox* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the ICheckBox* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Disabling Another Part

To disable another control when the user selects the check box, connect the ICheckBox* part's *selected* attribute to the *value* attribute of an IVBBooleanPart* part. Then, connect the IVBBooleanPart* part's *notValue* attribute to the *enabled* attribute of the other control. When the user deselects the check box, the other control is enabled.

IVBBooleanPart is available in the vbsample.vbb part file.

Enabling Another Part

To enable another control when the user selects the check box, connect the ICheckBox* part's *selected* attribute to the *enabled* attribute of the other control. When the user deselects the check box, the other control is disabled.

Disabling the Part

To disable the check box when an event occurs, connect the event to the ICheckBox* part's *disable* action.

Enabling the Part

To enable the check box when an event occurs, do the following:

1. Connect the event to the ICheckBox* part's *enable* action.
2. The check box is enabled by default. If you have changed this default, open settings for the connection, select **Set parameters**, and check the **Enabled** setting.

Preferred Features — ICheckBox

buttonClickEvent
Notification identifier provided to observers when the button control is clicked by the user.

disable Prevents keyboard and mouse input from being sent to the window.

enable Enables the window to accept keyboard and mouse input.

enabled If the window is sent mouse and keyboard input, `true` is returned.

menu Popup menu link.

selected If the button is selected, `true` is returned.

setFocus Sets the input focus to the window.

text Returns the control window's text.

textLength Returns the current length of the control window's text in bytes.

this A Pointer to the instance.

Features — ICheckBox

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	allowsMouseClickFocus ■	anyEvent
click	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
deselect	autoDeleteObject ■	buttonClickEvent
disable	autoDestroyWindow ■	commandEvent
disableAutoSelect	**autoSelect** ■	*disabledBackgroundColor*
disableGroup	backgroundColor ■ ►	*disabledForegroundColor*
disableMouseClickFocus	bidiSupported	*enabled*
disableNotification	borderColor ■ ►	*focus*
disableTabStop	characterSize	*font*
disableUpdate	clipboardHasTextFormat	*foregroundColor*
dispatchRemainingHandlers ■	defaultPushButton	gotFocusEvent ■
enable ■	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*

ICheckBox

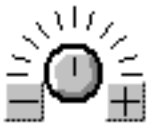
Action	Attribute	Event
enableUpdate ■	disabledForegroundColor ■ ►	*hiliteForegroundColor*
handleException ■	displaySize	*inactiveColor*
hide	enabled ■ ►	inputDisabledEvent ■
hideSourceEmphasis	enabledForNotification ■	inputEnabledEvent ■
highlight ■	focus ►	*isDigits*
isLayoutDistorted ■	font ■ ►	lostFocusEvent ■
matchForMnemonic ■	foregroundColor ■ ►	*position*
notifyObservers ■	frameWindow	*selected*
positionBehindSibling ■	group ■	*shadowColor*
positionBehindSiblings	handle	*size*
positionOnSiblings	helpId ■	systemCommandEvent
postEvent ■	highlighted	*text*
refresh ■	hiliteBackgroundColor ■ ►	*textLength*
releasePointer	hiliteForegroundColor ■ ►	*valueAsDouble*
releasePresSpace ■	id ■	*valueAsInt*
resetActiveColor	inactiveColor ■ ►	*valueAsUnsigned*
resetBackgroundColor	isDigits ►	visibilityDisabledEvent ■
resetBorderColor	itemProvider ■	visibilityEnabledEvent ■
resetDisabledBackgroundColor	layoutAdjustment	*visible*
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	rect ■	
setLayoutDistorted ■	selected ■ ►	
show ■	shadowColor ■ ►	
showSourceEmphasis ■	showing	
unhighlight	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

ICheckBox

Chapter 14. ICircularSlider

Description: IBM circular slider control
Part type: Visual
Part file: vbbase
Header file: icslider.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
ICircularSlider



Use an ICircularSlider* part to provide a range of seemingly nondiscrete selection values, such as degrees of brightness or loudness. A circular dial represents the range of values. A slider *arm* marks the selected value.

To use an ICircularSlider* part, ensure that you have installed multimedia support on your system. In Windows, this support is installed by default.

The user moves the arm around the dial to change the selected value. The user can move the arm in any of the following ways:

- Drag it with the mouse.
- Click or press a *slider button* to move the arm in the direction indicated on the button. This moves the arm by rotation increments defined for the slider.
- Click on a *tick mark* representing a value in the slider range to move the arm directly to the tick mark.

Portability — ICircularSlider

Features not defined for Windows:

- messageQueue

© Copyright IBM Corp. 1995, 1996

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ICircularSlider

Slider parts:

- "ICircularSlider" on page 95
- "IProgressIndicator" on page 261
- "IScrollBar" on page 287
- "ISlider" on page 301

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — ICircularSlider

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Value Range
- Setting the Increment Value
- Setting the Initial Value

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Value

Adding the Part

To add an ICircularSlider* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on a canvas.

Setting the Value Range

To define the value range for the circular slider, enter **Arm range** values in the **Lower** and **Upper** fields on the ICircularSlider* part's **General** settings page.

Setting the Increment Value

To determine the rotation increment for arm movement, enter an increment value in the **Rotation increment** field on the ICircularSlider* part's **General** settings page.

Setting the Initial Value

To define an initial value for the circular slider, enter a value in the **Current value** field on the ICircularSlider* part's **General** settings page. The arm is initially positioned at this value.

Getting the Selected Value

To get the selected value for an attribute or for a connection parameter, connect the ICircularSlider* part's *value* attribute to the target attribute or parameter.

Preferred Features — ICircularSlider

armRange	Returns the range of values over which the arm can be moved.
disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
enabled	If the window is sent mouse and keyboard input, `true` is returned.
menu	Popup menu link.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
this	A Pointer to the instance.
value	Returns the current value of the circular slider.

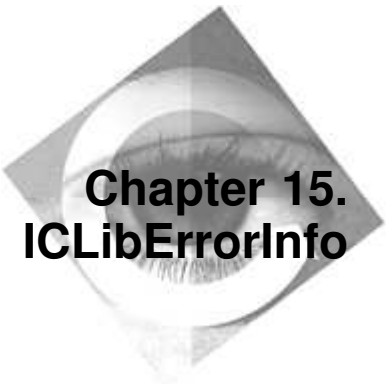
Features — ICircularSlider

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**armRange** ■	anyEvent
convertToGUIStyle ■	asDebugInfo	*backgroundColor*
disable	asString	*borderColor*
disableGroup	autoDeleteObject ■	commandEvent
disableNotification	autoDestroyWindow ■	*disabledBackgroundColor*
disableTabStop	backgroundColor ■ ►	*disabledForegroundColor*
disableUpdate	bidiSupported	*enabled*
dispatchRemainingHandlers ■	borderColor ■ ►	*focus*
enable ■	characterSize	*font*
enableUpdate ■	clipboardHasTextFormat	*foregroundColor*
handleException ■	defaultPushButton	gotFocusEvent ■
hide	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	disabledForegroundColor ■ ►	*hiliteForegroundColor*
isLayoutDistorted ■	displaySize	*inactiveColor*
matchForMnemonic ■	enabled ■ ►	inputDisabledEvent ■
notifyObservers ■	enabledForNotification ■	inputEnabledEvent ■
positionBehindSibling ■	focus ►	*isDigits*
positionBehindSiblings	font ■ ►	lostFocusEvent ■
positionOnSiblings	foregroundColor ■ ►	*position*
postEvent ■	frameWindow	*shadowColor*
refresh ■	group ■	*size*
releasePointer	handle	systemCommandEvent
releasePresSpace ■	helpId ■	*text*
resetActiveColor	hiliteBackgroundColor ■ ►	*textLength*
resetBackgroundColor	hiliteForegroundColor ■ ►	***value***
resetBorderColor	id ■	*valueAsDouble*
resetDisabledBackgroundColor	inactiveColor ■ ►	*valueAsInt*
resetDisabledForegroundColor	isDigits ►	*valueAsUnsigned*
resetFont	itemProvider ■	visibilityDisabledEvent ■
resetForegroundColor	layoutAdjustment	visibilityEnabledEvent ■

ICircularSlider

Action	Attribute	Event
resetHiliteBackgroundColor	menu	*visible*
resetHiliteForegroundColor	minimumSize ■	
resetInactiveColor	nativeRect	
resetMinimumSize	owner ■	
resetShadowColor	parent ■	
sendEvent ■	parentSize	
setDecrementBitmaps ■	pointerCaptured	
setFocus	position ■ ►	
setIncrementBitmaps ■	presSpace	
setLayoutDistorted ■	**radius**	
show ■	rect ■	
showSourceEmphasis ■	**rotationIncrement** ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	tickSpacing ■	
	valid	
	value ■ ►	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

ICircularSlider



Chapter 15. ICLibErrorInfo

Description: IBM access to CLib error information
Part type: Class
Part file: vbbase
Header file: iexcept.hpp

Derivation:
IBase
IVBase
IErrorInfo
ICLibErrorInfo

Preferred Features — ICLibErrorInfo

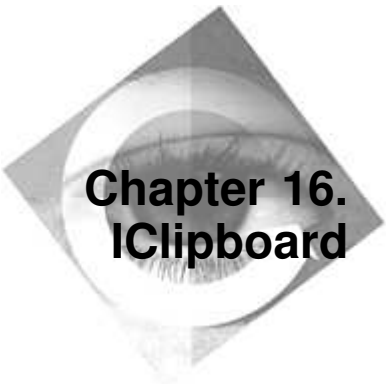
text	Returns the error text.
this	A Pointer to the instance.

Features — ICLibErrorInfo

Action	Attribute	Type
operator const char *	asDebugInfo	`IString`
throwCLibError ■	asString	`IString`
throwError ■	**available**	`Boolean`
	errorId	`unsigned long`
	text	`const char*`
	this	`ICLibErrorInfo*`

© Copyright IBM Corp. 1995, 1996

ICLibErrorInfo



Chapter 16. IClipboard

Description: IBM clipboard class
Part type: Class
Part file: vbbase
Header file: iclipbrd.hpp

Derivation:
IBase
IVBase
IClipboard

Use an IClipboard* part to provide clipboard operations for other parts.

Preferred Features — IClipboard

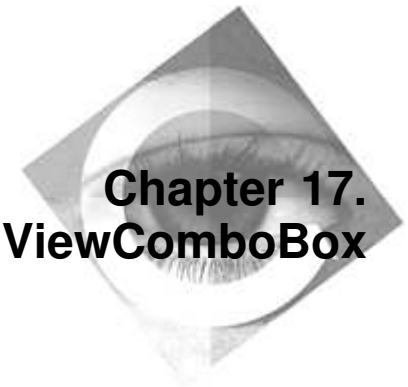
bitmap	If data of the format `IClipboard::bitmapFormat` exists on the clipboard, this function creates a copy of the bitmap and returns its IBitmapHandle.
close	Closes the clipboard.
data	Returns a void* value.
empty	Clears the clipboard of all formats of data and establishes the window provided on the constructor as the clipboard owner.
isOpen	Returns `true` if this `IClipboard` object has opened the clipboard.
open	Opens the clipboard.
text	Returns textFormat data as an `IString` object.
this	A Pointer to the instance.

Features — IClipboard

Action	Attribute	Type
close	asDebugInfo	`IString`
data ■	asString	`IString`
empty	**bitmap** ■	`IBitmapHandle`
format ■	**formatCount**	`unsigned long`
formatAsHandle ■	**hasBitmap**	`Boolean`
hasData ■	**hasText**	`Boolean`
open	**isOpen**	`Boolean`
registerFormat ■	**owner**	`IWindowHandle`
setData ■	**primaryFormat**	`IString`
setHandle ■	**text** ■	`IString`
	this	`IClipboard*`

© Copyright IBM Corp. 1995, 1996

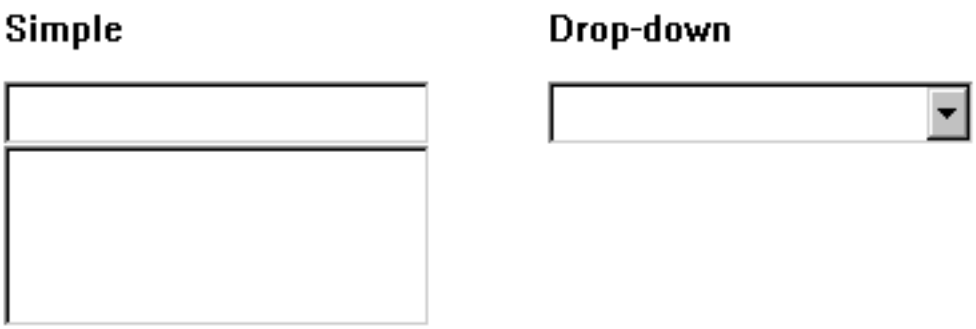
IClipboard



Chapter 17. ICollectionViewComboBox

Description: IBM collection combination-box control
Part type: Visual
Part file: vbbase
Header file: icombovw.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IEntryField
IBaseComboBox
ICollectionViewComboBox



Use an ICollectionViewComboBox* part to give the user a selection list and entry field for collection object choices. The combination box provides a view of the object collection. An IComboBox* part, by contrast, provides a set of string choices.

Use one of the following ICollectionViewComboBox* part types for selection choices:

Drop-down
Allows the user to type a choice or to select a choice from a drop-down list.

Read-only drop-down
Allows the user only to select a choice from a drop-down list.

Simple
Allows the user to type a choice or to select a choice from a list that is always displayed.

© Copyright IBM Corp. 1995, 1996

Alternatively, for object selection only from a list that is always displayed, use an ICollectionViewListBox* part. You can provide either single or multiple selection with a list box part.

For selection of strings rather than objects in a collection, use an IComboBox* part or an IListBox* part.

With a collection view combination box, the user can select an object from the list. With a simple collection view combination box, the user can type the beginning of a selection that is in the list. The closest match from the list is used. With a simple or drop-down collection view combination box, but not with a read-only drop-down combination box, the user can type a selection that is not in the list.

If you define the combination box as a drop-down or read-only drop-down type, the user does not initially see the list box but can click a drop-down button to open it.

Portability — ICollectionViewComboBox

Features not defined for Windows:

- readOnly
- messageQueue

Features not active on Windows:

- disableAutoScroll
- disableAutoTab
- disableCommand
- disableInsertMode
- disableMargin
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor

- resetShadowColor

Features active with restriction on Windows:

- alignment
- charType
- autoScroll
- autoTab
- command
- insertMode
- margin

Related Parts — ICollectionViewComboBox

Selection parts:

- "ICollectionViewComboBox" on page 105
- "ICollectionViewListBox" on page 113
- "IComboBox" on page 123
- "IListBox" on page 199
- "INumericSpinButton" on page 243
- "ITextSpinButton" on page 329

Building Guidelines — ICollectionViewComboBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Combination Box Type
- Setting the Object Type
- Setting the Collection Type
- Defining Strings for Objects
- Setting Initial Contents
- Setting the Initial Choice
- Enabling Tab Key Access

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Choice
- Adding a Choice
- Removing a Choice

ICollectionViewComboBox

Adding the Part

To add an ICollectionViewComboBox* part to a composite part, select an ICollectionViewComboBox* part from the parts palette. Then, drop the part on a canvas.

Setting the Combination Box Type

To determine the combination box type, select one of the following choices in the **Combination box type** field on the ICollectionViewComboBox* part's **General** settings page:

- **Simple** for a combination box with the list always open
- **Drop-down** for a drop-down combination box with an editable entry field
- **Read-only drop-down** for a drop-down combination box with a read-only entry field

Setting the Object Type

To define the type of object supported by the combination box, enter a pointer to the part type (part type and *) in the **Item type** field on the ICollectionViewComboBox* part's **General** settings page.

Setting the Collection Type

To define the type of collection containing the objects, specify the type in the **Collection type** field on the ICollectionViewComboBox* part's **General** settings page. The default collection type is IVSequence. If you are providing support for another collection type that you want to use, enter the type in this field. The vbcc.vbb part file contains examples of other collection types you could use.

Defining Strings for Objects

To define the strings that are displayed for objects in the combination box, identify a function or class that provides the object strings. Do this in either of the following ways:

- Provide an asString override function that produces the text you want for the collection object class. The function could retrieve and return a key text attribute for the object.
- Provide a string generator class that produces the text you want for the collection object class. The string generator could retrieve and return a key text attribute for the object. The generator class must be derived from the collection object class and must be a template class.

  If you use a string generator, enter the name of the generator class in the **String generator** field on the ICollectionViewComboBox* part's **General** settings page.

If this field is left empty, the default string generator calls asString for each collection object.

Setting Initial Contents

To define initial selection choices for the combination box, connect the *this* attribute of a collection part containing the choices to the ICollectionViewComboBox* part's *items* attribute. You must use a collection part that has one template argument, such as an IVSequence* part.

The combination box provides a view of the collection. The contents of the collection determine the contents of the combination box.

Setting the Initial Choice

To define a default selection for the combination box, specify the 1-based index of the selection in the **Selection index** field on the ICollectionViewComboBox* part's **General** settings page. The corresponding choice in the collection part is the default selection.

Enabling Tab Key Access

To allow the user to reach the combination box by using a tab key, set the **tabStop** field to **On** on the ICollectionViewComboBox* part's **Styles** settings page.

Getting the Selected Choice

To copy the selected object to a target that expects an object of the type listed in the combination box, connect the ICollectionViewComboBox* part's *selectedElement* attribute to the target part's *this* attribute. For example, if you are listing ICustomer* objects, connect the ICollectionViewComboBox* part's *selectedElement* attribute to an ICustomer* attribute or parameter.

To copy the text string for the selected object to an entry field, do the following:

- Add a IVBVariable* part to the free-form surface and change its type to the object type displayed in the combination box.
- Connect the ICollectionViewComboBox* part's *selectedElement* attribute to the IVBVariable* part's *this* attribute.
- Connect the IVBVariable* part's displayed text attribute to the IEntryField* part's *text* attribute.

For example, if you are listing ICustomer* objects by customer name, connect the ICollectionViewComboBox* part's *selectedElement* attribute to an IVBVariable* part that holds an ICustomer* part. Then, connect the IVBVariable* part's *name* attribute to the IEntryField* part's *text* attribute.

Adding a Choice

To add a new object to the combination box, add the object to the collection part that contains the objects. Use an add action for the collection type.

For example, if you are using an IVSequence* collection, you can use the IVSequence* part's *add*, *addAsFirst*, *addAsLast* or *addAtPosition* action to add the object to the collection. If you want to add the object at the position of the currently selected object, connect the ICollectionViewComboBox* part's *selection* attribute to the connection or action requiring the position parameter.

Removing a Choice

To remove an object from the combination box, remove the object from the collection part that contains the objects. Use a remove action for the collection type.

For example, if you are using an IVSequence* collection, you can use the IVSequence* part's *removeFirst*, *removeLast* or *removeAtPosition* action to remove the object from the collection. If you want to remove the currently selected object, connect the ICollectionViewComboBox* part's *selection* attribute to the connection or action requiring the position parameter.

Preferred Features — ICollectionViewComboBox

copy Copies the selected text to the clipboard.

cut Removes the selected text from the entry field control and puts it in the clipboard.

items Returns the collection currently being viewed.

limit Returns the length, in bytes, of the longest text the entry field can hold.

menu Popup menu link.

paste Copies text from the clipboard to the entry field control, replacing any selected text.

selectedCollectionPosition Returns the collection position corresponding to the selected item in the combination box.

selectedElement Returns the collection element corresponding to the selected item in the combination box.

selectedText Returns the selected text string.

setFocus Sets the input focus to the window.

text Returns the control window's text.

textLength Returns the current length of the control window's text in bytes.

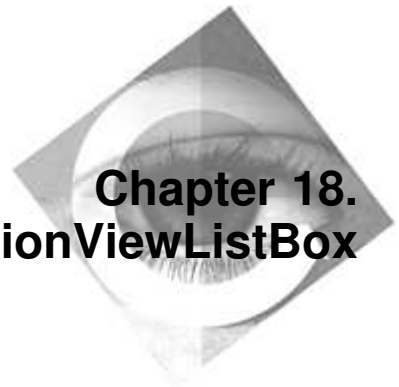
this A Pointer to the instance.

Features — ICollectionViewComboBox

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
clear ■	asString	*backgroundColor*
convertToGUIStyle ■	autoDeleteObject ■	*borderColor*
copy ■	autoDestroyWindow ■	characterTypeEvent
cut ■	autoScroll ■	*command*
deselect ■	autoTab ■	commandEvent
deselectAll	backgroundColor ■ ►	*disabledBackgroundColor*
disable	bidiSupported	*disabledForegroundColor*
disableAutoScroll	borderColor ■ ►	*enabled*
disableAutoTab	changed	enterEvent
disableCommand	characterSize	*focus*
disableDataUpdate	charType ■	*font*
disableGroup	clipboardHasTextFormat	*foregroundColor*
disableInsertMode	command ■ ►	gotFocusEvent ■
disableMargin	count	*hiliteBackgroundColor*
disableNotification	cursorPosition ■	*hiliteForegroundColor*
disableTabStop	defaultPushButton	*inactiveColor*
disableUpdate	disabledBackgroundColor ■ ►	inputDisabledEvent ■
discard	disabledForegroundColor ■ ►	inputEnabledEvent ■
dispatchRemainingHandlers ■	displaySize	*insertMode*
enable ■	empty	*isDigits*
enableUpdate ■	enabled ■ ►	**itemChangedEvent**
handleException ■	enabledForNotification ■	***items***
hide	focus ►	*limit*
hideList	font ■ ►	lostFocusEvent ■
hideSourceEmphasis	foregroundColor ■ ►	*numberOfSelections*
isLayoutDistorted ■	frameWindow	*position*
isSelected ■	group ■	***selectedCollectionPosition***
itemHandle ■	handle	***selectedElement***
itemText ■	hasSelectedText	***selection***
locateText ■	helpId ■	*shadowColor*
matchForMnemonic ■	hiliteBackgroundColor ■ ►	*size*
notifyObservers ■	hiliteForegroundColor ■ ►	systemCommandEvent
paste	horizontalScroll	*text*
positionBehindSibling ■	id ■	*textLength*
positionBehindSiblings	inactiveColor ■ ►	*valueAsDouble*
positionOnSiblings	insertMode ■ ►	*valueAsInt*
postEvent ■	isDigits ►	*valueAsUnsigned*
refresh ■	itemProvider ■	visibilityDisabledEvent ■
releasePointer	**items** ■ ►	visibilityEnabledEvent ■
releasePresSpace ■	layoutAdjustment	*visible*
removeAll	leftIndex ■	*writeable*
resetActiveColor	limit ■ ►	

ICollectionViewComboBox

Action	Attribute	Event
resetBackgroundColor	listShowing	
resetBorderColor	margin ■	
resetDisabledBackgroundColor	menu	
resetDisabledForegroundColor	minimumRows ■	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	numberOfSelections ►	
resetHiliteForegroundColor	owner ■	
resetInactiveColor	parent ■	
resetMinimumSize	parentSize	
resetShadowColor	pointerCaptured	
selectRange ■	position ■ ►	
sendEvent ■	presSpace	
setChangedFlag ■	rect ■	
setFocus	**selectedCollectionPosition** ■ ►	
setItemHandle ■	**selectedElement** ►	
setItemText ■	selectedRange	
setLayoutDistorted ■	selectedText	
show ■	selectedTextLength	
showList ■	**selection** ►	
showSourceEmphasis ■	shadowColor ■ ►	
	showing	
	size ■ ►	
	stringGenerator ■	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	top ■	
	type	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	
	writeable ■ ►	



Chapter 18. ICollectionViewListBox

Description: IBM collection list-box control
Part type: Visual
Part file: vbbase
Header file: ilistcvw.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IBaseListBox
ICollectionViewListBox

Use an ICollectionViewListBox* part to provide a modifiable list of object choices for the user. The list box provides a view of the object collection. An IListBox* part, by contrast, provides a set of string choices. Your application can add or remove choices from the list.

Alternatively, for object specification from an entry field or selection from a list, use an ICollectionViewComboBox* part. You can provide only single selection with a combination box part, but you can provide multiple selection with a list box.

For selection of strings rather than objects in a collection, use an IComboBox* part or an IListBox* part.

By default, the user can select only one choice from the list. When the user selects a choice, any previously selected choice is no longer selected. You can change the behavior of the list box to allow multiple selection.

Portability — ICollectionViewListBox

© Copyright IBM Corp. 1995, 1996

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- disableExtendedSelect
- disableMultipleSelect
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- extendedSelect
- multipleSelect

Related Parts — ICollectionViewListBox

Selection parts:

- "ICollectionViewComboBox" on page 105
- "ICollectionViewListBox" on page 113
- "IComboBox" on page 123
- "IListBox" on page 199
- "INumericSpinButton" on page 243
- "ITextSpinButton" on page 329

Building Guidelines — ICollectionViewListBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Object Type
- Setting the Collection Type
- Defining Strings for Objects
- Setting Initial Contents
- Defining Selection
- Enabling Tab Key Access

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Choice
- Adding a Choice
- Removing a Choice

Adding the Part

To add an ICollectionViewListBox* part to a composite part, select an ICollectionViewListBox* part from the parts palette. Then, drop the part on a canvas.

Setting the Object Type

To define the type of object supported by the list box, enter a pointer to the part type (part type and *) in the **Item type** field on the ICollectionViewListBox* part's **General** settings page.

Setting the Collection Type

To define the type of collection containing the objects, specify the type in the **Collection type** field on the ICollectionViewListBox* part's **General** settings page. The default collection type is IVSequence. If you are providing support for another collection type that you want to use, enter the type in this field. The vbcc.vbb part file contains examples of other collection types you could use.

Defining Strings for Objects

To define the strings that are displayed for objects in the list box, identify a function or class that provides the object strings. Do this in either of the following ways:

- Provide an asString override function that produces the text you want for the collection object class. The function could retrieve and return a key text attribute for the object.

- Provide a string generator class that produces the text you want for the collection object class. The string generator could retrieve and return a key text attribute for the object. The generator class must be derived from the collection object class and must be a template class.

  If you use a string generator, enter the name of the generator class in the **String generator** field on the ICollectionViewListBox* part's **General** settings page. If this field is left empty, the default string generator calls asString for each collection object.

Setting Initial Contents

To define initial selection choices for the list box, connect the *this* attribute of a collection part containing the choices to the ICollectionViewListBox* part's *items* attribute. You must use a collection part that has one template argument, such as an IVSequence* part.

The list box provides a view of the collection. The contents of the collection determine the contents of the list box.

Defining Selection

To allow the user to select only one choice, ensure that the **extendedSelection** field and the **multipleSelection** field on the ICollectionViewListBox* part's **Styles** settings page are both off. Set both fields to **Off** or, if you have not modified selection in the default style, to **Default**.

To allow extended selection, set the **extendedSelection** field to **On**.

To allow multiple selection, set the **multipleSelection** field to **On**.

Enabling Tab Key Access

To allow the user to reach the list box by using a tab key, set the **tabStop** field to **On** on the ICollectionViewListBox* part's **Styles** settings page.

Getting the Selected Choice

To copy the selected object to a target that expects an object of the type listed in the list box, connect the ICollectionViewListBox* part's *selectedElement* attribute to the target part's *this* attribute. For example, if you are listing ICustomer* objects, connect the ICollectionViewListBox* part's *selectedElement* attribute to an ICustomer* attribute or parameter.

To copy the text string for the selected object to an entry field, do the following:

- Add a IVBVariable* part to the free-form surface and change its type to the object type displayed in the list box.

- Connect the ICollectionViewListBox* part's *selectedElement* attribute to the IVBVariable* part's *this* attribute.
- Connect the IVBVariable* part's displayed text attribute to the IEntryField* part's *text* attribute.

For example, if you are listing ICustomer* objects by customer name, connect the ICollectionViewListBox* part's *selectedElement* attribute to an IVBVariable* part that holds an ICustomer* part. Then, connect the IVBVariable* part's *name* attribute to the IEntryField* part's *text* attribute.

Adding a Choice

To add a new object to the list box, add the object to the collection part that contains the objects. Use an add action for the collection type.

For example, if you are using an IVSequence* collection, you can use the IVSequence* part's *add*, *addAsFirst*, *addAsLast* or *addAtPosition* action to add the object to the collection. If you want to add the object at the position of the currently selected object, connect the ICollectionViewListBox* part's *selection* attribute to the connection or action requiring the position parameter.

Removing a Choice

To remove an object from the list box, remove the object from the collection part that contains the objects. Use a remove action for the collection type. For example, if you are using an IVSequence* collection, you can use the IVSequence* part's *removeFirst*, *removeLast* or *removeAtPosition* action to remove the object from the collection. If you want to remove the currently selected object, connect the ICollectionViewListBox* part's *selection* attribute to the connection or action requiring the position parameter.

Preferred Features — ICollectionViewListBox

enterEvent Notification identifier provided to observers when the user `double` clicks on an item or presses enter on an item with the cursor in a list box.

items Returns the collection currently being viewed.

menu Popup menu link.

selectedCollectionPosition Returns the collection position corresponding to the first selected item in the list box.

selectedElement Returns the collection element corresponding to the first selected item in the list box.

setFocus Sets the input focus to the window.

this A Pointer to the instance.

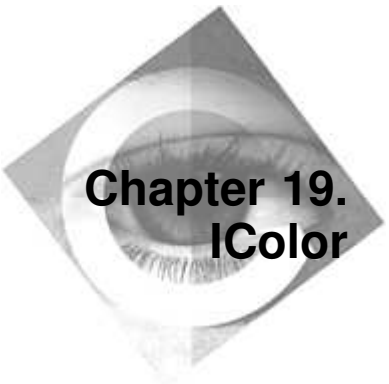
Features — ICollectionViewListBox

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
deselect ■	autoDeleteObject ■	*borderColor*
deselectAll	autoDestroyWindow ■	commandEvent
disable	backgroundColor ■ ►	*disabledBackgroundColor*
disableDrawItem	bidiSupported	*disabledForegroundColor*
disableExtendedSelect	borderColor ■ ►	*enabled*
disableGroup	characterSize	enterEvent
disableMultipleSelect	count	**extendedSelectChangedEvent**
disableNoAdjustPosition	defaultPushButton	*focus*
disableNotification	disabledBackgroundColor ■ ►	*font*
disableTabStop	disabledForegroundColor ■ ►	*foregroundColor*
disableUpdate	drawItem ■	gotFocusEvent ■
dispatchRemainingHandlers ■	empty	*hiliteBackgroundColor*
enable ■	enabled ■ ►	*hiliteForegroundColor*
enableUpdate ■	enabledForNotification ■	*inactiveColor*
handleException ■	extendedSelect ■	inputDisabledEvent ■
hide	focus ►	inputEnabledEvent ■
hideSourceEmphasis	font ■ ►	**itemChangedEvent**
isLayoutDistorted ■	foregroundColor ■ ►	***items***
isSelected ■	frameWindow	lostFocusEvent ■
itemHandle ■	group ■	*numberOfSelections*
itemText ■	handle	*position*
locateText ■	helpId ■	***selectedCollectionPosition***
matchForMnemonic ■	hiliteBackgroundColor ■ ►	***selectedElement***
notifyObservers ■	hiliteForegroundColor ■ ►	***selection***
positionBehindSibling ■	horizontalScroll	*selectionText*
positionBehindSiblings	id ■	*shadowColor*
positionOnSiblings	inactiveColor ■ ►	*size*
postEvent ■	itemProvider ■	systemCommandEvent
refresh ■	**items** ■ ►	visibilityDisabledEvent ■
releasePointer	layoutAdjustment	visibilityEnabledEvent ■
releasePresSpace ■	menu	*visible*
resetActiveColor	minimumCharacters ■	
resetBackgroundColor	minimumRows ■	
resetBorderColor	minimumSize ■	
resetDisabledBackgroundColor	multipleSelect ■	
resetDisabledForegroundColor	nativeRect	
resetFont	noAdjustPosition ■	
resetForegroundColor	numberOfSelections ►	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	

ICollectionViewListBox

Action	Attribute	Event
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
selectAll	presSpace	
selectedElements ■	rect ■	
sendEvent ■	**selectedCollectionPosition** ■ ►	
setFocus	**selectedElement** ►	
setItemHandle ■	**selection** ►	
setItemHeight ■	selectionText ►	
setItemText ■	shadowColor ■ ►	
setLayoutDistorted ■	showing	
show ■	size ■ ►	
showSourceEmphasis ■	**stringGenerator** ■	
	tabStop ■	
	this	
	top ■	
	valid	
	visible ■ ►	
	visibleRectangle	

ICollectionViewListBox



Chapter 19. IColor

Description: IBM color class
Part type: Class
Part file: vbbase
Header file: icolor.hpp

Derivation:
IBase
IVBase
IColor

Preferred Features — IColor

asRGBLong	Returns the red, green, and blue color values combined into a `long` integer.
blueMix	Returns the value of the blue component.
greenMix	Returns the value of the green component.
redMix	Returns the value of the red component.
this	A Pointer to the instance.
value	Using a specified color index, determines the Color enumerator.

Features — IColor

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator = ■	**asRGBLong**	`long`
operator == ■	asString	`IString`
	blueMix ■	`unsigned char`
	greenMix ■	`unsigned char`
	index	`long`
	redMix ■	`unsigned char`
	this	`IColor*`
	value	`IColor::Color`

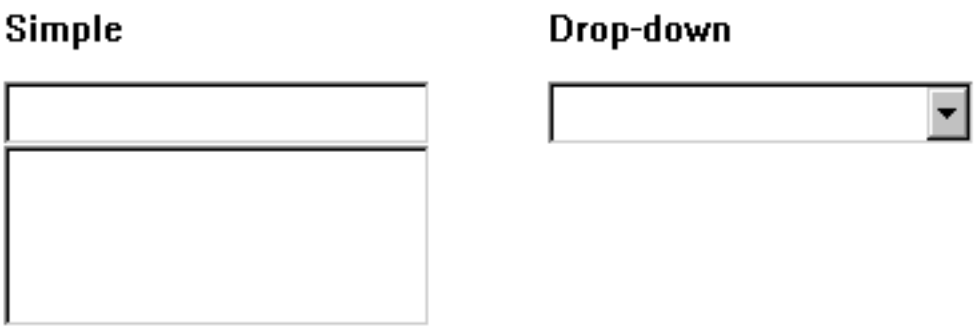
© Copyright IBM Corp. 1995, 1996

IColor

Chapter 20. IComboBox

Description: IBM string combination-box control
Part type: Visual
Part file: vbbase
Header file: icombobx.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IEntryField
IBaseComboBox
IComboBox



Use an IComboBox* part to give the user a selection list and entry field for string choices. The combination box provides a set of string choices. An ICollectionViewComboBox* part, by contrast, provides a view of an object collection.

Use one of the following IComboBox* part types for selection choices:

Drop-down
Allows the user to type a choice or to select a choice from a drop-down list.

Read-only drop-down
Allows the user only to select a choice from a drop-down list.

Simple
Allows the user to type a choice or to select a choice from a list that is always displayed.

Alternatively, for string selection only from a list that is always displayed, use an IListBox* part. You can provide either single or multiple selection with a list box part.

© Copyright IBM Corp. 1995, 1996

For selection of objects in a collection rather than strings, use an ICollectionViewComboBox* part or an ICollectionViewListBox* part.

With a string combination box, the user can select a string from the list. With a simple string combination box, the user can type the beginning of a selection that is in the list. The closest match from the list is used. With a simple or drop-down string combination box, but not with a read-only drop-down combination box, the user can type a selection that is not in the list.

If you define the combination box as a drop-down or read-only drop-down type, the user does not initially see the list box, but can click a drop-down button to open it.

Portability — IComboBox

Features not defined for Windows:

- readOnly
- messageQueue

Features not active on Windows:

- disableAutoScroll
- disableAutoTab
- disableCommand
- disableInsertMode
- disableMargin
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment
- charType
- autoScroll
- autoTab
- command
- insertMode
- margin

Related Parts — IComboBox

Selection parts:

- "ICollectionViewComboBox" on page 105
- "ICollectionViewListBox" on page 113
- "IComboBox" on page 123
- "IListBox" on page 199
- "INumericSpinButton" on page 243
- "ITextSpinButton" on page 329

Building Guidelines — IComboBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Combination Box Type
- Setting Initial Contents
- Setting the Initial Choice
- Enabling Tab Key Access

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Choice
- Adding a Choice
- Removing a Choice

Adding the Part

To add an IComboBox* part to a composite part, select an IComboBox* part from the parts palette. Then, drop the part on a canvas.

IComboBox

Setting the Combination Box Type

To determine the combination box type, select one of the following choices in the **Combination box type** field on the IComboBox* part's **General** settings page:

- **Simple** for a combination box with the list always open
- **Drop-down** for a drop-down combination box with an editable entry field
- **Read-only drop-down** for a drop-down combination box with a read-only entry field

Setting Initial Contents

To define initial selection choices for the combination box, enter the choices in the **Contents** field on the IComboBox* part's **General** settings page.

Setting the Initial Choice

To define a default selection for the combination box, specify the 0-based index of the selection in the **Selection index** field on the IComboBox* part's **General** settings page. The corresponding choice in the **Contents** field is the default selection.

Enabling Tab Key Access

To allow the user to reach the combination box by using a tab key, set the **tabStop** field to **On** on the IComboBox* part's **Styles** settings page.

Getting the Selected Choice

To copy the text selection from the combination box to an entry field, connect the IComboBox* part's *text* attribute to the *text* attribute of the IEntryField* part.

To copy the index of the selection to a numeric field, connect the IComboBox* part's *selection* attribute to the target numeric field.

Adding a Choice

Add a new choice to the combination box list from the entry field when a push button is pressed. To add the choice to the end of the list when a push button is pressed, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to the *addAsLast* action of the IComboBox* part.
2. Connect the IComboBox* part's *text* attribute to the *text* parameter of the *buttonClickEvent*-to-*addAsLast* connection. This provides the entry field text as the parameter for the *addAsLast* action.

Removing a Choice

To remove the selected choice from the combination box list when a push button is pressed, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to the *remove* action of the IComboBox* part.
2. Connect the IComboBox* part's *selection* attribute to the *index* parameter of the *buttonClickEvent*-to-*remove* connection. This provides the selection index as the parameter for the *remove* action.

Preferred Features — IComboBox

addAscending	Inserts the specified item in ascending sort order and returns the index of the item inserted.
addAsLast	Inserts a line of text at the end of the list box portion of the combination box.
copy	Copies the selected text to the clipboard.
cut	Removes the selected text from the entry field control and puts it in the clipboard.
limit	Returns the length, in bytes, of the longest text the entry field can hold.
menu	Popup menu link.
paste	Copies text from the clipboard to the entry field control, replacing any selected text.
removeAll	Deletes the entire contents of the entry field control.
selectedText	Returns the selected text string.
selection	Returns the 0-based index of the selected item.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

Features — IComboBox

Action	Attribute	Event
add ■	activeColor ■ ►	*activeColor*
addAscending ■	asDebugInfo	anyEvent
addAsFirst ■	asString	*backgroundColor*
addAsLast ■	autoDeleteObject ■	*borderColor*
addDescending ■	autoDestroyWindow ■	characterTypeEvent
applyBidiSettings ■	autoScroll ■	*command*
capturePointer ■	autoTab ■	commandEvent
clear ■	backgroundColor ■ ►	*disabledBackgroundColor*

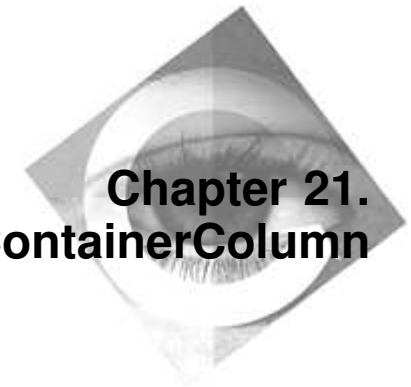
IComboBox

Action	Attribute	Event
convertToGUIStyle ■	bidiSupported	*disabledForegroundColor*
copy ■	borderColor ■ ►	*enabled*
cut ■	changed	enterEvent
deselect ■	characterSize	*focus*
deselectAll	charType ■	*font*
disable	clipboardHasTextFormat	*foregroundColor*
disableAutoScroll	command ■ ►	gotFocusEvent ■
disableAutoTab	count	*hiliteBackgroundColor*
disableCommand	cursorPosition ■	*hiliteForegroundColor*
disableDataUpdate	defaultPushButton	*inactiveColor*
disableGroup	disabledBackgroundColor ■ ►	inputDisabledEvent ■
disableInsertMode	disabledForegroundColor ■ ►	inputEnabledEvent ■
disableMargin	displaySize	*insertMode*
disableNotification	empty	*isDigits*
disableTabStop	enabled ■ ►	*limit*
disableUpdate	enabledForNotification ■	lostFocusEvent ■
discard	focus ►	*numberOfSelections*
dispatchRemainingHandlers ■	font ■ ►	*position*
enable ■	foregroundColor ■ ►	*selection*
enableUpdate ■	frameWindow	*shadowColor*
handleException ■	group ■	*size*
hide	handle	systemCommandEvent
hideList	hasSelectedText	*text*
hideSourceEmphasis	helpId ■	*textLength*
isLayoutDistorted ■	hiliteBackgroundColor ■ ►	*valueAsDouble*
isSelected ■	hiliteForegroundColor ■ ►	*valueAsInt*
itemHandle ■	horizontalScroll	*valueAsUnsigned*
itemText ■	id ■	visibilityDisabledEvent ■
locateText ■	inactiveColor ■ ►	visibilityEnabledEvent ■
matchForMnemonic ■	insertMode ■ ►	*visible*
notifyObservers ■	isDigits ►	*writeable*
paste	itemProvider ■	
positionBehindSibling ■	layoutAdjustment	
positionBehindSiblings	leftIndex ■	
positionOnSiblings	limit ■ ►	
postEvent ■	listShowing	
refresh ■	margin ■	
releasePointer	menu	
releasePresSpace ■	minimumRows ■	
remove ■	minimumSize ■	
removeAll	nativeRect	
resetActiveColor	numberOfSelections ►	
resetBackgroundColor	owner ■	
resetBorderColor	parent ■	
resetDisabledBackgroundColor	parentSize	
resetDisabledForegroundColor	pointerCaptured	
resetFont	position ■ ►	

IComboBox

Action	Attribute	Event
resetForegroundColor	presSpace	
resetHiliteBackgroundColor	rect ■	
resetHiliteForegroundColor	selectedRange	
resetInactiveColor	selectedText	
resetMinimumSize	selectedTextLength	
resetShadowColor	selection ■ ►	
selectRange ■	shadowColor ■ ►	
sendEvent ■	showing	
setChangedFlag ■	size ■ ►	
setFocus	tabStop ■	
setItemHandle ■	text ■ ►	
setItemText ■	textLength ►	
setLayoutDistorted ■	**this**	
show ■	top ■	
showList ■	type	
showSourceEmphasis ■	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	
	writeable ■ ►	

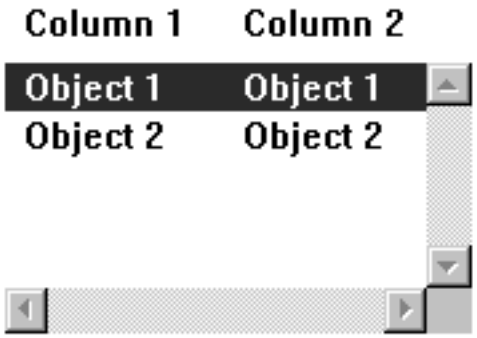
IComboBox



Chapter 21. IContainerColumn

Description: IBM column for container details view
Part type: Class
Part file: vbbase
Header file: icnrcol.hpp

Derivation:
IBase
IVBase
IContainerColumn



Use an IContainerColumn* part to define the heading and contents of a container details view column. The column contains information on a particular attribute of each container object.

The user can view the column data for container objects. With modifiable text data, the user can edit the *object* attribute.

Related Parts — IContainerColumn

Container parts:

- "IContainerColumn"
- "IContainerObject" on page 135
- "IVBContainerControl" on page 359

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IContainerColumn

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining the Column Heading
- Defining Column Data
- Reordering Columns

Adding the Part

To add an IContainerColumn* part to a composite part, select an IContainerColumn* part from the parts palette. Then, drop the part on an IVBContainerControl* part.

To gain access to the container column while building, select the **showDetailsView** choice in the **View type** field on the IVBContainerControl* part's **General** settings page.

Defining the Column Heading

To define the column heading, enter the heading in the **Heading text** field on the IContainerColumn* part's **General** settings page.

Defining Column Data

To specify what data appears in the column, set the **Column definition** field on the IContainerColumn* part's **General** settings page.

If you use the native Windows container, we strongly recommend that you define the first column to display the object icon and the second column to display the object text. This ensures that the user will see the object identification column when the details view opens.

The *object identification column* is the visual combination of the object icon and the object text. The user can select an object with the mouse only by clicking on the object identification column. If you define the first two columns to contain other data, Windows inserts the object icon and text before the columns you define and initially hides the object identification column. The user must then drag the edge of the first column you define to expose the object identification column.

To show the object icon for each container item, select the **Use Icon attribute set in the container** setting. The icon specification is set on the IVBContainerControl* part's **General** settings page.

To show the object text of each container item, select the **Use Text attribute set in the container** setting. The text presentation is set on the IVBContainerControl* part's **General** settings page.

To show an attribute of each container item, select the **Use an attribute from the part** setting. Then, select a choice from the **Attributes** list box. Only text-related attributes are available as choices.

Reordering Columns

To reorder container columns, open the IVBContainerControl* part's contextual menu and select **View parts list**. Then, in the parts list window, drag IContainerColumn* parts within the list to reorder them.

Preferred Features — IContainerColumn

date If the data in the column is a date, `true` is returned.

displayWidth Returns the displayable width of the column, in pixels.

headingIcon If the style is icon, the icon in the heading is returned.

headingIconHandle
If the data in the column heading is an icon, `true` is returned.

headingString If the data in the column heading is a character string, `true` is returned.

headingText If the style is string, the text in the heading is returned.

headingWriteable
If the data in the column heading can be edited, `true` is returned.

helpId Retrieves the help panel ID.

horizontalSeparator
If the column heading has a separator drawn under it, `true` is returned.

iconHandle If the data in the column is an icon, `true` is returned.

number If the data in the column is a number, `true` is returned.

string If the data in the column is a pointer to a character string or an `IString`, `true` is returned.

this A Pointer to the instance.

time If the data in the column is a time, `true` is returned.

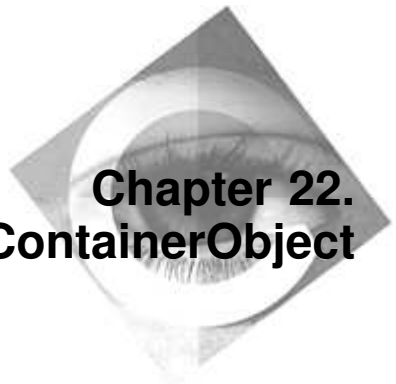
verticalSeparator
If the column heading has a vertical separator drawn after the column, `true` is returned.

visible If the column is visible in the container, `true` is returned.

writeable If the user can edit the data in the column, `true` is returned.

Features — IContainerColumn

Action	Attribute	Type
dataAsDate ■	asDebugInfo	IString
dataAsIcon ■	asString	IString
dataAsNumber ■	**date**	Boolean
dataAsString ■	**displayWidth** ■	unsigned long
dataAsTime ■	**headingIcon** ■	IPointerHandle
disableDataUpdate	**headingIconHandle**	Boolean
disableHeadingUpdate	**headingString**	Boolean
hide	**headingText** ■	IString
hideSeparators ■	**headingWriteable** ■	Boolean
justifyData ■	**helpId** ■	unsigned long
justifyHeading ■	**horizontalDataAlignment**	IContainerColumn::HorizontalAlignment
setDataOffset ■	**horizontalHeadingAlignment**	IContainerColumn::HorizontalAlignment
showSeparators ■	**horizontalSeparator**	Boolean
	iconHandle	Boolean
	number	Boolean
	string	Boolean
	this	IContainerColumn*
	time	Boolean
	verticalDataAlignment	IContainerColumn::VerticalAlignment
	verticalHeadingAlignment	IContainerColumn::VerticalAlignment
	verticalSeparator	Boolean
	visible ■	Boolean
	writeable ■	Boolean



Chapter 22. IContainerObject

Description: IBM base class for all container items
Part type: Class
Part file: vbbase
Header file: icnrobj.hpp

Derivation:
IBase
IVBase
IContainerObject

IContainerObject* parts support container objects that Visual Builder generates for you. These objects are determined by settings that you define for IVBContainerControl* and IContainerColumn* parts. You do not need to use IContainerObject* parts directly.

Related Parts — IContainerObject

Container parts:

- "IContainerColumn" on page 131
- "IContainerObject"
- "IVBContainerControl" on page 359

Preferred Features — IContainerObject

icon	Returns the icon handle of an object.
open	If this object is open, `true` is returned.
this	A Pointer to the instance.

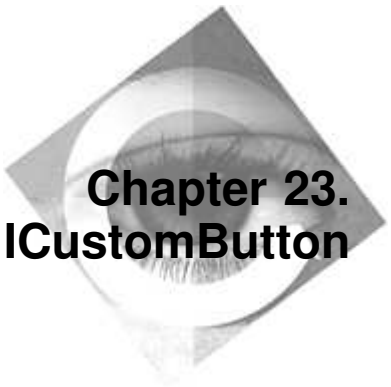
Features — IContainerObject

Action	Attribute	Type
disableDataUpdate ■	asDebugInfo	`IString`
disableDrop ■	asString	`IString`
handleCursoredChange ■	**dropOnAble** ■	`Boolean`
handleInuseChange ■	**helpId**	`unsigned long`
handleOpen ■	**icon** ■	`IPointerHandle`
handleSelectedChange ■	**iconText** ■	`IString`
handleTreeCollapse ■	**inUse** ■	`Boolean`
handleTreeExpand ■	**objectCopy**	`IContainerObject*`

© Copyright IBM Corp. 1995, 1996

IContainerObject

Action	Attribute	Type
hide ■	**open** ■	`Boolean`
operator == ■	**refreshOn** ■	`Boolean`
operator delete ■	**this**	`IContainerObject*`
operator new ■	**visible** ■	`Boolean`
refresh ■	**writeable** ■	`Boolean`
removeInUse ■		
setClosed		
setRefreshOff		



Chapter 23. ICustomButton

Description: IBM custom button control
Part type: Visual
Part file: vbbase
Header file: icustbut.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
ICustomButton

Use an ICustomButton* part to give the user an action choice button that can remain latched down when it is selected. Your application can customize a custom button. A multimedia play button is an example of a custom button.

The user can click the custom button to perform the indicated function.

Portability — ICustomButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor

© Copyright IBM Corp. 1995, 1996

- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ICustomButton

Button parts:

- "IAnimatedButton" on page 73
- "ICustomButton" on page 137
- "IGraphicPushButton" on page 175
- "INumericSpinButton" on page 243
- "IPushButton" on page 267
- "IRadioButton" on page 273
- "ITextSpinButton" on page 329
- "IToolBarButton" on page 349

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — ICustomButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Adding a Handler for the Part
- Enabling Button Latching
- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting the Requested Action

- Disabling the Part
- Enabling the Part

Adding the Part

To add an ICustomButton* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the a canvas.

Defining Text for the Part

To assign text to the custom button, set the **Text** field on the ICustomButton* part's **General** settings page.

Adding a Handler for the Part

To assign a customization handler for the custom button, specify an ICustomButtonDrawHandler part on the ICustomButton* part's **Handlers** settings page.

Enabling Button Latching

To enable latching the button down when it is pressed, set the **latchable** field on the ICustomButton* part's **Styles** settings page.

Enabling Tab Key Access

To allow the user to reach the button by using a tab key, set the **tabStop** field to **On** on the ICustomButton* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the ICustomButton* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Starting the Requested Action

To start the action represented by the button, connect the ICustomButton* part's *buttonClickEvent* feature to the action on the target part.

Disabling the Part

To disable the custom button when an event occurs, connect the event to the ICustomButton* part's *disable* action.

Enabling the Part

To enable the custom button when an event occurs, do the following:

1. Connect the event to the ICustomButton* part's *enable* action.
2. The button is enabled by default. If you have changed this default, open settings for the connection, select **Set parameters**, and check the **Enabled** setting.

Preferred Features — ICustomButton

buttonClickEvent
Notification identifier provided to observers when the button control is clicked by the user.

disable Prevents keyboard and mouse input from being sent to the window.

enable Enables the window to accept keyboard and mouse input.

enabled If the window is sent mouse and keyboard input, `true` is returned.

latched Returns `true` if the button is in a latched state.

menu Popup menu link.

setFocus Sets the input focus to the window.

text Returns the control window's text.

this A Pointer to the instance.

Features — ICustomButton

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	allowsMouseClickFocus ■	anyEvent
click	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
disable	autoDeleteObject ■	buttonClickEvent
disableAutoLatch	autoDestroyWindow ■	commandEvent
disableGroup	**autoLatchEnabled** ■	*disabledBackgroundColor*
disableLatching	backgroundColor ■ ►	*disabledForegroundColor*
disableMouseClickFocus	bidiSupported	*enabled*
disableNotification	borderColor ■ ►	*focus*
disableTabStop	characterSize	*font*
disableUpdate	clipboardHasTextFormat	*foregroundColor*
dispatchRemainingHandlers ■	defaultPushButton	gotFocusEvent ■
enable ■	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
enableUpdate ■	disabledForegroundColor ■ ►	*hiliteForegroundColor*
handleException ■	displaySize	*inactiveColor*
hide	enabled ■ ►	inputDisabledEvent ■

ICustomButton

Action	Attribute	Event
hideSourceEmphasis	enabledForNotification ■	inputEnabledEvent ■
highlight ■	focus ►	*isDigits*
isLayoutDistorted ■	font ■ ►	***latched***
matchForMnemonic ■	foregroundColor ■ ►	lostFocusEvent ■
notifyObservers ■	frameWindow	*position*
positionBehindSibling ■	group ■	*shadowColor*
positionBehindSiblings	handle	*size*
positionOnSiblings	helpId ■	systemCommandEvent
postEvent ■	highlighted	*text*
refresh ■	hiliteBackgroundColor ■ ►	*textLength*
releasePointer	hiliteForegroundColor ■ ►	*valueAsDouble*
releasePresSpace ■	id ■	*valueAsInt*
resetActiveColor	inactiveColor ■ ►	*valueAsUnsigned*
resetBackgroundColor	isDigits ►	visibilityDisabledEvent ■
resetBorderColor	itemProvider ■	visibilityEnabledEvent ■
resetDisabledBackgroundColor	**latched** ■ ►	*visible*
resetDisabledForegroundColor	**latchedBackgroundColor** ■	
resetFont	**latchedBackgroundColorHalftone**	
resetForegroundColor	**latchedForegroundColor** ■	
resetHiliteBackgroundColor	**latchingEnabled** ■	
resetHiliteForegroundColor	layoutAdjustment	
resetInactiveColor	menu	
resetLatchedBackgroundColor	minimumSize ■	
resetLatchedForegroundColor	nativeRect	
resetMinimumSize	owner ■	
resetShadowColor	parent ■	
sendEvent ■	parentSize	
setFocus	pointerCaptured	
setLayoutDistorted ■	position ■ ►	
show ■	presSpace	
showSourceEmphasis ■	rect ■	
unhighlight	shadowColor ■ ►	
unlatch	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	userData ■	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

ICustomButton

Chapter 24. IDate

Description: IBM date class
Part type: Class
Part file: vbbase
Header file: idate.hpp

Derivation:
IBase
IDate

Use an IDate* part to query and format dates.

Portability — IDate

Features not defined for Windows:

- asCDATE

Preferred Features — IDate

dayName	Returns the name of the receiver's day of the week.
dayOfMonth	Returns the day in the receiver's month as an integer from 1 to 31.
dayOfYear	Returns the day in the receiver's year as an integer from 1 to 366.
julianDate	Returns the Julian day number of the receiver `IDate`.
monthName	Returns the name of the receiver's month.
this	A Pointer to the instance.
today	Returns the current date.
year	Returns the receiver's year.

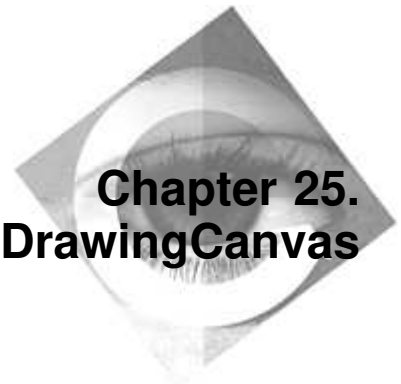
Features — IDate

Action	Attribute	Type
dayName ■	asDebugInfo	`IString`
daysInMonth ■	asString	`IString`
daysInYear ■	**dayOfMonth**	`int`
isLeapYear ■	**dayOfWeek**	`IDate::DayOfWeek`
isValid ■	**dayOfYear**	`int`
monthName ■	**julianDate**	`unsigned long`
operator != ■	**monthOfYear**	`IDate::Month`
operator + ■	**this**	`IDate*`
operator += ■	**today**	`IDate`

© Copyright IBM Corp. 1995, 1996

IDate

Action	Attribute	Type
operator - ■	**year**	`int`
operator -= ■		
operator < ■		
operator <= ■		
operator == ■		
operator > ■		
operator >= ■		



Chapter 25. IDrawingCanvas

Description: IBM drawing canvas control
Part type: Visual
Part file: vbbase
Header file: idrawcv.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
IDrawingCanvas

Use an IDrawingCanvas* part to provide a canvas for graphic composition. Your application can add graphic objects to the drawing canvas.

Alternatively, use one of the following canvas parts:

- ICanvas* for a simple canvas
- IMultiCellCanvas* to provide freely arranged canvas cells
- ISetCanvas* for canvas cells in uniform rows or columns
- ISplitCanvas* for split windows

The user can interact with graphic objects on the canvas.

Portability — IDrawingCanvas

Features not defined for Windows:

- messageQueue

© Copyright IBM Corp. 1995, 1996

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IDrawingCanvas

Canvas parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

Client parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “IMultiLineEdit” on page 231
- “INotebook” on page 237
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307
- “IVBContainerControl” on page 359
- “IViewPort” on page 399

Building Guidelines — IDrawingCanvas

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding an Object
- Adding a Handler for the Part
- Selecting a Color

Adding the Part

To add an IDrawingCanvas* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Then, drop the part on one of the following targets:
  - An IFrameWindow* part
  - An IViewPort* part
  - A canvas part

Adding an Object

To add a graphic object to the canvas, add the object to the graphic list created by the IDrawingCanvas* part. To add the object to the graphic list, write code using one of the IGList add member functions.

Adding a Handler for the Part

If the useDefaultPaintHandler style is set for the IDrawingCanvas* part, a default paint handler is provided. If you want to provide your own paint handler, do the following:

- Set the **useDefaultPaintHandler** field **Off** on the IDrawingCanvas* part's **Styles** settings page.
- Add the handler name and constructor parameters in the **Handler list** field on the IDrawingCanvas* part's **Handlers** settings page.
- If the handler is a part that is loaded into Visual Builder, you do not need to specify the handler's header file name. Otherwise, type the header file name for the handler in the Class Editor **Required include files** field.

Selecting a Color

To specify a background color for the canvas, select a color from the **Colors** field or set RGB values in the **RGB values** field on the IDrawingCanvas* part's **Color** settings page.

Preferred Features — IDrawingCanvas

menu	Popup menu link.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

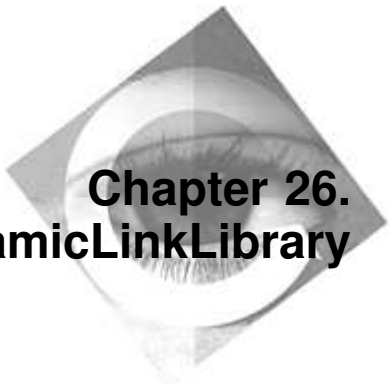
Features — IDrawingCanvas

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	defaultPushButton	*font*
enableUpdate ■	disabledBackgroundColor ■ ►	*foregroundColor*
handleException ■	disabledForegroundColor ■ ►	gotFocusEvent ■
hide	enabled ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	focus ►	*inactiveColor*
matchForMnemonic ■	font ■ ►	inputDisabledEvent ■
notifyObservers ■	foregroundColor ■ ►	inputEnabledEvent ■
positionBehindSibling ■	frameWindow	lostFocusEvent ■
positionBehindSiblings	**graphicContext** ■	*position*
positionOnSiblings	**graphicList** ■	*shadowColor*
postEvent ■	group ■	*size*
refresh ■	handle	systemCommandEvent
releasePointer	helpId ■	visibilityDisabledEvent ■
releasePresSpace ■	hiliteBackgroundColor ■ ►	visibilityEnabledEvent ■
resetActiveColor	hiliteForegroundColor ■ ►	*visible*
resetBackgroundColor	id ■	
resetBorderColor	inactiveColor ■ ►	
resetDisabledBackgroundColor	itemProvider ■	
resetDisabledForegroundColor	layoutAdjustment	
resetFont	menu	
resetForegroundColor	minimumSize ■	
resetHiliteBackgroundColor	nativeRect	

IDrawingCanvas

Action	Attribute	Event
resetHiliteForegroundColor	origDefaultButtonHandle	
resetInactiveColor	owner ■	
resetMinimumSize	parent ■	
resetShadowColor	parentSize	
sendEvent ■	pointerCaptured	
setFocus	position ■ ►	
setLayoutDistorted ■	presSpace	
show ■	rect ■	
showSourceEmphasis ■	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	this	
	valid	
	visible ■ ►	
	visibleRectangle	

IDrawingCanvas



Chapter 26. IDynamicLinkLibrary

Description: IBM support for linking to DLL
Part type: Class
Part file: vbbase
Header file: ireslib.hpp

Derivation:
IBase
IVBase
IResourceLibrary
IDynamicLinkLibrary

Preferred Features — IDynamicLinkLibrary

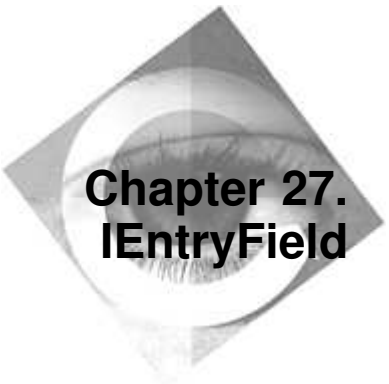
close	Closes a dynamic link library and decrements the reference count.
fileName	Returns the fully qualified file name of the resource library.
open	Opens a dynamic link library and increments the reference count.
this	A Pointer to the instance.

Features — IDynamicLinkLibrary

Action	Attribute	Type
close	asDebugInfo	`IString`
isEntryPoint32Bit ■	asString	`IString`
loadAccelTable ■	fileName	`IString`
loadBitmap ■	handle	`IModuleHandle`
loadDialog ■	**this**	`IDynamicLinkLibrary*`
loadHelpTable ■		
loadIcon ■		
loadMenu ■		
loadMessage ■		
loadPointer ■		
loadString ■		
open		
operator = ■		
procAddress ■		
tryToLoadBitmap ■		
tryToLoadIcon ■		
tryToLoadMessage ■		
tryToLoadString ■		

© Copyright IBM Corp. 1995, 1996

IDynamicLinkLibrary



Chapter 27. IEntryField

Description: IBM entry field control
Part type: Visual
Part file: vbbase
Header file: ientryfd.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IEntryField

Use an IEntryField* part to let the user enter or view a text string. Alternatively, use an IMultiLineEdit* part for multiline data.

Note: Although top and bottom margins do not appear for an IEntryField* part in the Composition Editor, these margins are present in the object created by generated code.

The user can type or place text into the entry field.

Portability — IEntryField

Features not defined for Windows:

- readOnly
- messageQueue

Features not active on Windows:

- disableAutoScroll
- disableAutoTab
- disableCommand
- disableInsertMode
- disableMargin
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment
- charType
- autoScroll
- autoTab
- command
- insertMode
- margin

Related Parts — IEntryField

Data entry parts:

- "IEntryField" on page 153
- "IMultiLineEdit" on page 231

Building Guidelines — IEntryField

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Loading Initial Text
- Preventing User Changes
- Enabling Tab Key Access

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Loading Text

- Saving Text

Adding the Part

To add an IEntryField* part to a composite part, select an IEntryField* part from the parts palette. Then, drop the part on a canvas.

Defining Text for the Part

To define constant initial text for the entry field, enter the text you want in the **Text** field on the IEntryField* part's **General** settings page.

Loading Initial Text

To load variable initial text into the entry field, connect your data source to the IEntryField* part's *text* attribute. Your application should make this connection before it shows the frame window.

Preventing User Changes

To prevent the user from entering or modifying text, select the **readOnly** field on the IEntryField* part's **Styles** settings page.

Enabling Tab Key Access

To allow the user to reach the entry field by using a tab key, set the **tabStop** field to **On** on the IEntryField* part's **Styles** settings page.

Loading Text

To load text into the entry field when the user presses a push button, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to the IEntryField* part's *text* attribute.
2. Connect the data source to the *text* parameter of the *buttonClickEvent*-to-*text* connection.

Saving Text

To copy or save the contents of the entry field when the user presses a push button, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to an action of the target part that sets text for the part.
2. Connect the IEntryField* part's *text* attribute to a text parameter of the *buttonClickEvent*-to-action connection.

Alternatively, if you want to copy the entry field text as numeric data, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to an action of the target part that sets the numeric value for the part.
2. Connect the IEntryField* part's *valueAsInt* attribute or *valueAsDouble* attribute to a numeric parameter of the *buttonClickEvent*-to-action connection.

Preferred Features — IEntryField

copy	Copies the selected text to the clipboard.
cut	Removes the selected text from the entry field control and puts it in the clipboard.
limit	Returns the length, in bytes, of the longest text the entry field can hold.
menu	Popup menu link.
paste	Copies text from the clipboard to the entry field control, replacing any selected text.
removeAll	Deletes the entire contents of the entry field control.
selectedText	Returns the selected text string.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.
valueAsInt	Return asInt from the `IString` text attribute

Features — IEntryField

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**alignment** ■	anyEvent
clear ■	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
copy ■	autoDeleteObject ■	**characterTypeEvent**
cut ■	autoDestroyWindow ■	***command***
disable	**autoScroll** ■	commandEvent
disableAutoScroll	**autoTab** ■	*disabledBackgroundColor*
disableAutoTab	backgroundColor ■ ►	*disabledForegroundColor*
disableCommand	bidiSupported	*enabled*
disableDataUpdate	borderColor ■ ►	*focus*
disableGroup	**changed**	*font*
disableInsertMode	characterSize	*foregroundColor*
disableMargin	**charType** ■	gotFocusEvent ■
disableNotification	clipboardHasTextFormat	*hiliteBackgroundColor*
disableTabStop	**command** ■ ►	*hiliteForegroundColor*
disableUpdate	**cursorPosition** ■	*inactiveColor*
discard	defaultPushButton	inputDisabledEvent ■

IEntryField

Action	Attribute	Event
dispatchRemainingHandlers ■	disabledBackgroundColor ■ ►	inputEnabledEvent ■
enable ■	disabledForegroundColor ■ ►	***insertMode***
enableUpdate ■	displaySize	*isDigits*
handleException ■	**empty**	***limit***
hide	enabled ■ ►	lostFocusEvent ■
hideSourceEmphasis	enabledForNotification ■	*position*
isLayoutDistorted ■	focus ►	*shadowColor*
matchForMnemonic ■	font ■ ►	*size*
notifyObservers ■	foregroundColor ■ ►	systemCommandEvent
paste	frameWindow	*text*
positionBehindSibling ■	group ■	*textLength*
positionBehindSiblings	handle	*valueAsDouble*
positionOnSiblings	**hasSelectedText**	*valueAsInt*
postEvent ■	helpId ■	*valueAsUnsigned*
refresh ■	hiliteBackgroundColor ■ ►	visibilityDisabledEvent ■
releasePointer	hiliteForegroundColor ■ ►	visibilityEnabledEvent ■
releasePresSpace ■	id ■	*visible*
removeAll	inactiveColor ■ ►	***writeable***
resetActiveColor	**insertMode** ■ ►	
resetBackgroundColor	isDigits ►	
resetBorderColor	itemProvider ■	
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	**leftIndex** ■	
resetFont	**limit** ■ ►	
resetForegroundColor	**margin** ■	
resetHiliteBackgroundColor	menu	
resetHiliteForegroundColor	minimumSize ■	
resetInactiveColor	nativeRect	
resetMinimumSize	owner ■	
resetShadowColor	parent ■	
selectRange ■	parentSize	
sendEvent ■	pointerCaptured	
setChangedFlag ■	position ■ ►	
setFocus	presSpace	
setLayoutDistorted ■	rect ■	
show ■	**selectedRange**	
showSourceEmphasis ■	**selectedText**	
	selectedTextLength	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	

IEntryField

Action	Attribute	Event
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	
	writeable ■ ►	

Chapter 28. IFlyOverHelpHandler

Description: IBM handler class for fly-over help
Part type: Class
Part file: vbbase
Header file: iflyhhdr.hpp

Derivation:
IBase
IVBase
IHandler
IFlyOverHelpHandler

Use an IFlyOverHelpHandler* part as a tear-off attribute for IVBFlyText* parts. An IFlyOverHelpHandler* part provides additional fly-over text features, such as initial delay time and dynamic help definition, for IVBFlyText* parts.

Preferred Features — IFlyOverHelpHandler

longTextControl
Returns a pointer to the text control that is used to display `long` help text.

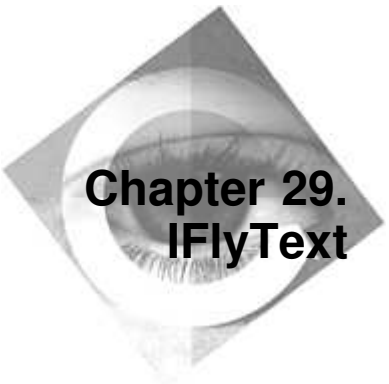
this A Pointer to the instance.

Features — IFlyOverHelpHandler

Action	Attribute	Type
disable	asDebugInfo	`IString`
flyHelpText ■	asString	`IString`
longHelpText ■	**defaultText** ■	`IString`
removeHelpText ■	**delayTime** ■	`unsigned long`
setHelpText ■	enabled ■	`Boolean`
start ■	**flyTextControl** ■	`IFlyText*`
stop ■	**flyTextStringTableOffset** ■	`long`
	longStringTableOffset ■	`long`
	longTextControl ■	`ITextControl*`
	resourceLibrary ■	`IResourceLibrary`
	this	`IFlyOverHelpHandler*`

© Copyright IBM Corp. 1995, 1996

IFlyOverHelpHandler



Chapter 29. IFlyText

Description: IBM fly-over text class
Part type: Nonvisual
Part file: vbbase
Header file: iflytext.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IFlyText

IFlyText* parts support IVBFlyText* parts that you use in your applications. You do not need to use IFlyText* parts directly.

Portability — IFlyText

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

© Copyright IBM Corp. 1995, 1996

Preferred Features — IFlyText

relativeWindowRect
This function returns the rectangle of the control which the `IFlyText` control is positioned relative to.

this A Pointer to the instance.

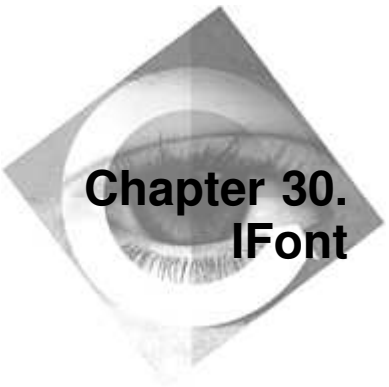
Features — IFlyText

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	clipboardHasTextFormat	*font*
enableUpdate ■	defaultPushButton	*foregroundColor*
handleException ■	disabledBackgroundColor ■ ►	gotFocusEvent ■
hide	disabledForegroundColor ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	displaySize	*hiliteForegroundColor*
isLayoutDistorted ■	enabled ■ ►	*inactiveColor*
matchForMnemonic ■	enabledForNotification ■	inputDisabledEvent ■
notifyObservers ■	focus ►	inputEnabledEvent ■
positionBehindSibling ■	font ■ ►	*isDigits*
positionBehindSiblings	foregroundColor ■ ►	lostFocusEvent ■
positionOnSiblings	frameWindow	*position*
postEvent ■	group ■	*shadowColor*
refresh ■	handle	*size*
releasePointer	helpId ■	systemCommandEvent
releasePresSpace ■	hiliteBackgroundColor ■ ►	*text*
resetActiveColor	hiliteForegroundColor ■ ►	*textLength*
resetBackgroundColor	id ■	*valueAsDouble*
resetBorderColor	inactiveColor ■ ►	*valueAsInt*
resetDisabledBackgroundColor	isDigits ►	*valueAsUnsigned*
resetDisabledForegroundColor	itemProvider ■	visibilityDisabledEvent ■
resetFont	layoutAdjustment	visibilityEnabledEvent ■
resetForegroundColor	menu	*visible*
resetHiliteBackgroundColor	minimumSize ■	
resetHiliteForegroundColor	nativeRect	
resetInactiveColor	owner ■	
resetMinimumSize	parent ■	
resetShadowColor	parentSize	

IFlyText

Action	Attribute	Event
sendEvent ■	pointerCaptured	
setFocus	position ■ ►	
setLayoutDistorted ■	presSpace	
show ■	rect ■	
showSourceEmphasis ■	**relativeWindowRect** ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

IFlyText



Chapter 30. IFont

Description: IBM class to manage fonts
Part type: Class
Part file: vbbase
Header file: ifont.hpp

Derivation:
IBase
IVBase
IFont

Portability — IFont

Features not defined for Windows:

- avgLowercase
- avgUppercase
- fattrs
- fontmetrics
- maxLowercaseAscender
- maxLowercaseDescender
- subscriptOffset
- subscriptSize
- superscriptOffset
- superscriptSize

Preferred Features — IFont

bold	If the `IFont` object uses a font that is bold, `true` is returned.
italic	If the `IFont` object uses an italicized font, `true` is returned.
name	Returns the face name of the font.
outline	If the `IFont` object uses a font that appears hollow, `true` is returned.
pointSize	Returns the point size of the font.
strikeout	If the `IFont` object uses a font with a line drawn through the characters, `true` is returned.
this	A Pointer to the instance.
underscore	If the `IFont` object uses a font with a line drawn under the characters, `true` is returned.

© Copyright IBM Corp. 1995, 1996

Features — IFont

Action	Attribute	Type
beginUsingFont ■	asDebugInfo	IString
charWidth ■	asString	IString
endUsingFont ■	**avgCharWidth**	unsigned long
minTextWidth ■	**bitmap**	Boolean
operator = ■	**bitmapOnly**	Boolean
setAllEmphasis ■	**bold** ■	Boolean
setCharHeight ■	**externalLeading**	unsigned long
setCharSize ■	**fixed**	Boolean
setCharWidth ■	**internalLeading**	unsigned long
setDirection ■	**italic** ■	Boolean
setFontAngle ■	**maxAscender**	unsigned long
setFontShear ■	**maxCharHeight**	unsigned long
setWindowFont ■	**maxDescender**	unsigned long
textLines ■	**maxSize**	ISize
textWidth ■	**maxUppercaseSize**	ISize
useBitmapOnly ■	**name** ■	IString
useNonPropOnly ■	**nonPropOnly**	Boolean
useVectorOnly ■	**outline** ■	Boolean
	pointSize ■	unsigned long
	strikeout ■	Boolean
	this	IFont*
	underscore ■	Boolean
	vectorOnly	Boolean

Chapter 31. IFrameWindow

Description: IBM window with title bar and extensions
Part type: Visual
Part file: vbbase
Header file: iframe.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IFrameWindow

Use an IFrameWindow* part as a primary or secondary window for your application. A *primary window* is the base window for your application. A *secondary window* is opened from the primary window or from another secondary window.

A frame window can contain the following *frame controls*:

- System menu
- Title bar
- Sizing buttons
- Border
- Menu bar

The portion of the frame window not occupied by frame controls is called the *client area*. Your application program uses the client area for the presentation and gathering of application data.

© Copyright IBM Corp. 1995, 1996

The client area is occupied and managed by a *client control*: You can use Visual Builder to provide one of the following client control parts:

- A canvas part for layout of controls
- An IMultiLineEdit* part for multiline text entry
- An INotebook* part for a notebook of controls
- An IViewPort* part for a scrollable view window

Your application uses these controls, or other controls within them, for most user interaction.

You can also add extensions to the frame window. Visual Builder provides the following frame extension parts:

- IMenu* parts for menu bars
- IToolBar* parts for application tool controls
- IVBInfoArea* parts for menu choice descriptions and long fly-over help

The user interacts with your application through the frame window.

Portability — IFrameWindow

Features not defined for Windows:

- shareParentDBCSStatus
- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IFrameWindow

Frame parts:

- "IFrameWindow" on page 167
- "IMenu" on page 205
- "IScrollBar" on page 287
- "ITitle" on page 337
- "IToolBar" on page 343
- "IVBInfoArea" on page 385

Client parts:

- "ICanvas" on page 83
- "IDrawingCanvas" on page 145
- "IMultiCellCanvas" on page 225
- "IMultiLineEdit" on page 231
- "INotebook" on page 237
- "ISetCanvas" on page 293
- "ISplitCanvas" on page 307
- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Model parts:

- "IVBFactory" on page 371
- "IVBVariable" on page 395
- "IVSequence" on page 403

Building Guidelines — IFrameWindow

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining the Title
- Adding a Menu
- Adding a Tool Bar
- Adding an Information Area
- Changing the Window Client
- Assigning Ownership
- Loading Data

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Opening a Secondary Window
- Closing a Window

Adding the Part

To add an IFrameWindow* part to a composite part, select an IFrameWindow* part from the parts palette. Then, drop the part on the free-form surface.

Defining the Title

To define a title for the frame window, set the **Title text** field on the IFrameWindow* part's **General** settings page.

Alternatively, use an ITitle* part for the frame window title. If you use an ITitle* part, you can change the frame window title after the window is created. The IFrameWindow* part's **Title text** field has no effect if you use the ITitle* part.

Adding a Menu

To add a menu bar to the frame window, do the following:

1. Select an IMenu* part from the parts palette and drop it on the free-form surface.
2. Connect the IMenu* part's *this* attribute to the IFrameWindow* part's *menu* attribute.

Adding a Tool Bar

To add a tool bar to the frame window, select an IToolBar* part from the parts palette and drop it on the IFrameWindow* part's title bar. The tool bar is added at the top of the client area.

Adding an Information Area

To add an information area to the frame window, select an IVBInfoArea part from the parts palette and drop it on the IFrameWindow* part's title bar. The information area is added at the bottom of the client area.

Changing the Window Client

To change the client for the frame window to an IMultiCellCanvas* part, do the following:

1. Select the current client part
2. Open the client part's contextual menu and select the **Delete** option
3. Select the IMultiCellCanvas* from the parts palette and drop it on the client area

If you have controls on a client area canvas that you want to retain for the new client, move them to the free-form surface before you delete the old client. Then, place them on the new client after it is added.

Assigning Ownership

To give the primary frame window ownership of a secondary window, do one of the following:

- For a static secondary window, connect the primary IFrameWindow* part's *this* attribute to the secondary IFrameWindow* part's *owner* attribute. The connection direction must be from the primary part to the secondary part.
- For a secondary window that you create dynamically with an IVBFactory* part, connect the primary IFrameWindow* part's *this* attribute to the IVBFactory* part's *owner* attribute.

Loading Data

To load data from your primary view into a secondary view, connect objects in the primary view composite part to corresponding attributes of the secondary view composite part. The secondary view attributes are IVBVariable* parts you have promoted to the interface of the composite part for the secondary view. These variables are connected to data fields within your secondary view composite part.

Opening a Secondary Window

To open a secondary window from a menu item or push button in the primary window, do one of the following:

- For a static secondary window, do the following:
  1. Connect the IMenuItem* part's *commandEvent* feature or the IPushButton* part's *buttonClickEvent* feature to the secondary IFrameWindow* part's *setFocus* action.
  2. Connect the *commandEvent* feature or the *buttonClickEvent* feature to the secondary IFrameWindow* part's *show* action or *showModally* action.
- For a secondary window to be created dynamically with an IVBFactory* part, do the following:
  1. Connect the IMenuItem* part's *commandEvent* feature or the IPushButton* part's *buttonClickEvent* feature to the IVBFactory* part's *new* action.
  2. Add an IVBVariable* part for the secondary view composite part to your primary view composite part.
  3. Connect the IVBFactory* part's *newEvent* feature to the IVBVariable* part's *this* attribute.
  4. Connect the IVBFactory* part's *newEvent* feature to the IVBVariable* part's *show* action or *showModally* action.
  5. Connect the IVBFactory* part's *newEvent* feature to the IVBVariable* part's *setFocus* action.

Closing a Window

To remove a window when the user presses a push button to close the window, do one of the following:

- For a static modeless window that must be available to show again, connect the IPushButton* part's *buttonClickEvent* feature to the IFrameWindow* part's *hide* action. Also, connect the primary composite part's *ready* event (from the free-form surface) to the secondary IFrameWindow* part's *removeFromWindowList* action.
- For a static modeless window that is not to be shown again, connect the IPushButton* part's *buttonClickEvent* feature to the IFrameWindow* part's *close* action. Also, connect the primary composite part's *ready* event (from the free-form surface) to the secondary IFrameWindow* part's *removeFromWindowList* action.
- For a static modal window, connect the IPushButton* part's *buttonClickEvent* feature to the IFrameWindow* part's *close* action. Also, connect the primary composite part's *ready* event (from the free-form surface) to the secondary IFrameWindow* part's *removeFromWindowList* action.
- For a dynamically created modeless window, connect the IPushButton* part's *buttonClickEvent* feature to the IFrameWindow* part's *close* action.

  To delete the window object, set the **Auto delete** field on the IVBFactory* part's **General** settings page. This represents the **General** settings of the secondary IFrameWindow* part.
- For a dynamically created modal window, connect the IPushButton* part's *buttonClickEvent* feature to the IFrameWindow* part's *close* action.

  To delete the window object, connect the IVBFactory* part's *newEvent* feature to the *deleteTarget* action of the IVBVariable* part for the secondary view composite part. Make this connection after the *newEvent*-to-*showModally* connection.

Preferred Features — IFrameWindow

close	Closes the frame window.
hide	Hides the window.
icon	Returns the pointer handle of the window icon.
menu	Menu bar link.
setFocus	Sets the input focus to the window.
showModally	Displays the frame window in application-modal mode (all other application windows are disabled).
this	A Pointer to the instance.
visible	If the window's style is set to visible, `true` is returned.

Features — IFrameWindow

Action	Attribute	Event
addExtension ■	activeColor ■ ►	**activateEvent**
addToWindowList	asDebugInfo	*activeColor*
applyBidiSettings ■	asString	anyEvent
beginFlashing	autoDeleteObject ■	*backgroundColor*
capturePointer ■	autoDestroyWindow ■	*borderColor*
clientRectFor ■	backgroundColor ■ ►	**closeEvent**
close	bidiSupported	commandEvent
convertToGUIStyle ■	borderColor ■ ►	**deactivateEvent**
disable	**borderHeight** ■	*disabledBackgroundColor*
disableNotification	**borderSize** ■	*disabledForegroundColor*
disableUpdate	**borderWidth** ■	*enabled*
dismiss ■	characterSize	*focus*
dispatchRemainingHandlers ■	**client** ■	*font*
enable ■	**clientHandle**	*foregroundColor*
enableUpdate ■	**defaultOrdering** ■	gotFocusEvent ■
endFlashing	defaultPushButton	*hiliteBackgroundColor*
frameRectFor ■	disabledBackgroundColor ■ ►	*hiliteForegroundColor*
handleException ■	disabledForegroundColor ■ ►	*inactiveColor*
handleFor ■	enabled ■ ►	inputDisabledEvent ■
hide	enabledForNotification ■	inputEnabledEvent ■
hideSourceEmphasis	**flashing**	lostFocusEvent ■
isAnExtension ■	focus ►	*position*
isLayoutDistorted ■	font ■ ►	***result***
matchForMnemonic ■	foregroundColor ■ ►	*shadowColor*
maximize	frameWindow	*size*
minimize	group	systemCommandEvent
moveSizeToClient ■	handle	visibilityDisabledEvent ■
notifyObservers ■	helpId ■	visibilityEnabledEvent ■
notifyOwner ■	hiliteBackgroundColor ■ ►	*visible*
positionBehindSibling ■	hiliteForegroundColor ■ ►	
positionBehindSiblings	**icon** ■	
positionOnSiblings	id ■	
postEvent ■	inactiveColor ■ ►	
refresh ■	itemProvider ■	
releasePointer	layoutAdjustment	
releasePresSpace ■	**maximized**	
removeExtension ■	**maximizeRect**	
removeFromWindowList	**menu**	
resetActiveColor	**minimized**	
resetBackgroundColor	**minimizeRect**	
resetBorderColor	minimumSize ■	
resetDisabledBackgroundColor	**modal**	
resetDisabledForegroundColor	**mousePointer** ■	
resetFont	nativeRect	

IFrameWindow

Action	Attribute	Event
resetForegroundColor	**nextShellRect**	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
restore	presSpace	
sendEvent ■	rect ■	
setBorderSize ■	**restoreRect** ■	
setDestroyOnClose ■	**result** ■ ►	
setExtensionSize ■	shadowColor ■ ►	
setFocus	showing	
setLayoutDistorted ■	size ■ ►	
show ■	tabStop	
showModally	**this**	
showSourceEmphasis ■	**toolBarList** ■	
start ■	**usesDialogBackground**	
update	valid	
useExtensionMinimumSize ■	visible ■ ►	
	visibleRectangle	
	willDestroyOnClose	

Chapter 32. IGraphicPushButton

Description: IBM push-button control with graphic
Part type: Visual
Part file: vbbase
Header file: igraphbt.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
IPushButton
IGraphicPushButton

Use an IGraphicPushButton* part to let the user request a function that can be represented graphically.

Alternatively, to identify the choice with text, use an IPushButton* part.

The user can click the push button to perform the indicated function.

Portability — IGraphicPushButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- border
- removeBorder
- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor

© Copyright IBM Corp. 1995, 1996

- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IGraphicPushButton

Button parts:

- "IAnimatedButton" on page 73
- "ICustomButton" on page 137
- "IGraphicPushButton" on page 175
- "INumericSpinButton" on page 243
- "IPushButton" on page 267
- "IRadioButton" on page 273
- "ITextSpinButton" on page 329
- "IToolBarButton" on page 349

Building Guidelines — IGraphicPushButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Assigning a Bitmap
- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting the Requested Action
- Disabling the Part
- Enabling the Part

Adding the Part

To add an IGraphicPushButton* part to a composite part, select an IGraphicPushButton* part from the parts palette. Then, drop the part on a canvas.

Assigning a Bitmap

To assign a bitmap to the graphic button, set the **DLL name** and the **Resource ID** fields on the IGraphicPushButton* part's **General** settings page.

Enabling Tab Key Access

To allow the user to reach the button by using a tab key, set the **tabStop** field to **On** on the IGraphicPushButton* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the IGraphicPushButton* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Starting the Requested Action

To start the action represented by the button, connecting the IGraphicPushButton* part's *buttonClickEvent* feature to the action on the target part.

Disabling the Part

To disable the button when an event occurs, connect the event to the IGraphicPushButton* part's *disable* action.

Enabling the Part

To enable the button when an event occurs, do the following:

1. Connect the event to the IGraphicPushButton* part's *enable* action.
2. Open settings for the connection, select **Set parameters**, and check the **enabled** setting.

Preferred Features — IGraphicPushButton

bitmap	Returns the handle of the currently set bit map.
buttonClickEvent	Notification identifier provided to observers when the button control is clicked by the user.
disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
enabled	If the window is sent mouse and keyboard input, `true` is returned.
icon	Returns the handle of the currently set icon.
menu	Popup menu link.

setFocus	Sets the input focus to the window.
text	Returns the control window's text.
this	A Pointer to the instance.

Features — IGraphicPushButton

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	allowsMouseClickFocus ■	anyEvent
click	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
disable	autoDeleteObject ■	buttonClickEvent
disableDefault	autoDestroyWindow ■	commandEvent
disableGroup	backgroundColor ■ ►	*disabledBackgroundColor*
disableHelp	bidiSupported	*disabledForegroundColor*
disableMouseClickFocus	**bitmap** ■	*enabled*
disableNotification	border ■	*focus*
disableSizeToGraphic	borderColor ■ ►	*font*
disableSystemCommand	characterSize	*foregroundColor*
disableTabStop	clipboardHasTextFormat	gotFocusEvent ■
disableUpdate	**currentGraphicType**	*hiliteBackgroundColor*
dispatchRemainingHandlers ■	default ■	*hiliteForegroundColor*
enable ■	defaultPushButton	*inactiveColor*
enableUpdate ■	disabledBackgroundColor ■ ►	inputDisabledEvent ■
handleException ■	disabledForegroundColor ■ ►	inputEnabledEvent ■
hide	displaySize	*isDigits*
hideSourceEmphasis	enabled ■ ►	lostFocusEvent ■
highlight ■	enabledForNotification ■	*position*
isLayoutDistorted ■	focus ►	*shadowColor*
matchForMnemonic ■	font ■ ►	*size*
notifyObservers ■	foregroundColor ■ ►	*systemCommand*
positionBehindSibling ■	frameWindow	systemCommandEvent
positionBehindSiblings	**graphicWindow**	*text*
positionOnSiblings	group ■	*textLength*
postEvent ■	handle	*valueAsDouble*
refresh ■	help ■	*valueAsInt*
releasePointer	helpId ■	*valueAsUnsigned*
releasePresSpace ■	highlighted	visibilityDisabledEvent ■
removeBorder	hiliteBackgroundColor ■ ►	visibilityEnabledEvent ■
resetActiveColor	hiliteForegroundColor ■ ►	*visible*
resetBackgroundColor	**icon** ■	
resetBorderColor	id ■	
resetDisabledBackgroundColor	inactiveColor ■ ►	
resetDisabledForegroundColor	isDigits ►	
resetFont	itemProvider ■	
resetForegroundColor	layoutAdjustment	
resetHiliteBackgroundColor	**marginSize** ■	

IGraphicPushButton

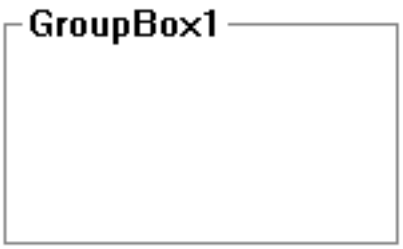
Action	Attribute	Event
resetHiliteForegroundColor	menu	
resetInactiveColor	minimumSize ■	
resetMinimumSize	nativeRect	
resetShadowColor	owner ■	
sendEvent ■	parent ■	
setFocus	parentSize	
setLayoutDistorted ■	pointerCaptured	
show ■	position ■ ►	
showSourceEmphasis ■	presSpace	
unhighlight	rect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	sizeToGraphic ■	
	systemCommand ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

IGraphicPushButton

Chapter 33. IGroupBox

Description: IBM group-box control
Part type: Visual
Part file: vbbase
Header file: igroupbx.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IGroupBox



Use an IGroupBox* part to outline and label a group of controls. Alternatively, use an IOutlineBox* part if you do not want to label the group. In an IMultiCellCanvas* part, you should use an ISetCanvas* part to represent a group box.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Portability — IGroupBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor

© Copyright IBM Corp. 1995, 1996

- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IGroupBox

Label parts:

- "IGroupBox" on page 181
- "IOutlineBox" on page 249
- "IStaticText" on page 315

Building Guidelines — IGroupBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Sizing the Part
- Adding Controls
- Reordering Controls
- Defining Text for the Part
- Defining a Control Group

Adding the Part

To add an IGroupBox* part to a composite part, select an IGroupBox* part from the parts palette. Then, drop the part on a canvas.

Sizing the Part

To resize the IGroupBox* part to the size you need, select the part. Then, drag a corner handle to size the part.

Adding Controls

To add the controls you want in the group box, select control parts from the parts palette and drop them on the IGroupBox* part.

Reordering Controls

If you add the group box after adding the controls you are enclosing, you must change the depth order of controls on the canvas to position the group box ahead of the controls within it.

Also, if you added the controls in the group box out of order, reorder the controls to reflect the logical order you want.

To reorder controls on a canvas, open the canvas part's contextual menu and select **View parts list**. Then, in the parts list window, drag parts to the new positions you want.

Defining Text for the Part

To define the text label for the group box, enter the label in the **Text** field on the IGroupBox* part's **General** settings page.

Defining a Control Group

To make the controls in the group box a control group that the user can reach with a tab key, do the following:

- Set the **group** and **tabStop** fields to **On** on the **Styles** settings page of the first control in the group box.
- Leave these fields set to **Off** for the rest of the controls in the group box.
- Ensure that the first control following the group box on your canvas has the **group** field set to **On**. This makes the last control in the group box the end of the control group.

Preferred Features — IGroupBox

menu	Popup menu link.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

Features — IGroupBox

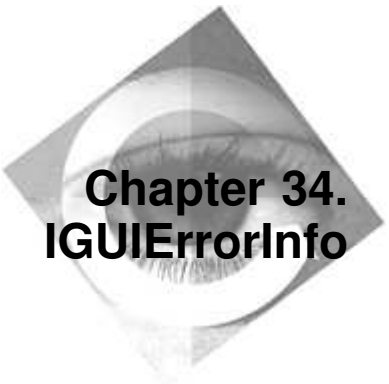
Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent

IGroupBox

Action	Attribute	Event
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	clipboardHasTextFormat	*font*
enableUpdate ■	defaultPushButton	*foregroundColor*
handleException ■	disabledBackgroundColor ■ ►	gotFocusEvent ■
hide	disabledForegroundColor ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	displaySize	*hiliteForegroundColor*
isLayoutDistorted ■	enabled ■ ►	*inactiveColor*
matchForMnemonic ■	enabledForNotification ■	inputDisabledEvent ■
notifyObservers ■	focus ►	inputEnabledEvent ■
positionBehindSibling ■	font ■ ►	*isDigits*
positionBehindSiblings	foregroundColor ■ ►	lostFocusEvent ■
positionOnSiblings	frameWindow	*position*
postEvent ■	group ■	*shadowColor*
refresh ■	handle	*size*
releasePointer	helpId ■	systemCommandEvent
releasePresSpace ■	hiliteBackgroundColor ■ ►	*text*
resetActiveColor	hiliteForegroundColor ■ ►	*textLength*
resetBackgroundColor	id ■	*valueAsDouble*
resetBorderColor	inactiveColor ■ ►	*valueAsInt*
resetDisabledBackgroundColor	isDigits ►	*valueAsUnsigned*
resetDisabledForegroundColor	itemProvider ■	visibilityDisabledEvent ■
resetFont	layoutAdjustment	visibilityEnabledEvent ■
resetForegroundColor	menu	*visible*
resetHiliteBackgroundColor	minimumSize ■	
resetHiliteForegroundColor	nativeRect	
resetInactiveColor	owner ■	
resetMinimumSize	parent ■	
resetShadowColor	parentSize	
sendEvent ■	pointerCaptured	
setFocus	position ■ ►	
setLayoutDistorted ■	presSpace	
show ■	rect ■	
showSourceEmphasis ■	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	

Action	Attribute	Event
	visibleRectangle	

IGroupBox



Chapter 34. IGUIErrorInfo

Description: IBM access to GUI error information
Part type: Class
Part file: vbbase
Header file: iexcept.hpp

Derivation:
IBase
IVBase
IErrorInfo
IGUIErrorInfo

Preferred Features — IGUIErrorInfo

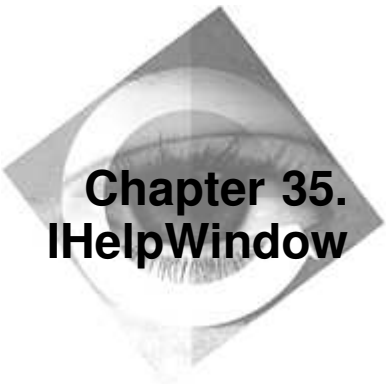
text	Returns the error text.
this	A Pointer to the instance.

Features — IGUIErrorInfo

Action	Attribute	Type
operator const char *	asDebugInfo	`IString`
throwError ■	asString	`IString`
throwGUIError ■	**available**	`Boolean`
	errorId	`unsigned long`
	text	`const char*`
	this	`IGUIErrorInfo*`

© Copyright IBM Corp. 1995, 1996

IGUIErrorInfo



Chapter 35. IHelpWindow

Description: IBM window for online help
Part type: Nonvisual
Part file: vbbase
Header file: ihelp.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IHelpWindow

Use an IHelpWindow* part to display help information for your application. The help window represents your set of help panels. You can use the help window for general and contextual help.

The user can open a help window from a help menu, a help push button, or the F1 function key.

Portability — IHelpWindow

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor

© Copyright IBM Corp. 1995, 1996

- resetInactiveColor
- resetShadowColor

Related Parts — IHelpWindow

Help parts:

- "IHelpWindow" on page 189
- "IVBFlyText" on page 377
- "IVBInfoArea" on page 385

Building Guidelines — IHelpWindow

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining the Title
- Specifying the Help Type
- Identifying Help Files
- Identifying a Help Panel

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Associating Help with a Frame Window

Adding the Part

To add an IHelpWindow* part to a composite part, select an IHelpWindow* part from the parts palette. Then, drop the part on the free-form surface.

Defining the Title

To define the title for your help window, enter the title in the **Title** field on the IHelpWindow* part's **General** settings page.

Specifying the Help Type

If you are using Information Presentation Facility (IPF) help, set the **ipfCompatible** field to **On** on the IHelpWindow* part's **Styles** settings page.

If you are using native help for the system, set the **ipfCompatible** field to **Off** or, if you have not change the default style setting, use the **Default** setting. For Windows, the native help is Rich Text Format (RTF). For OS/2, the native help is IPF.

Identifying Help Files

To identify the help (.hlp) files for your application, enter the file names, separated by spaces, in the **Help libraries** field on the IHelpWindow* part's **General** settings page. If you do not specify a path with the file name, the environment help path is used.

Identifying a Help Panel

To link each frame window and control with its help panel, enter the help panel ID in the **Help panel ID** field on its Control settings page. The panel ID is the resource number specified for the the help panel heading.

Associating Help with a Frame Window

For a multiple-window application, you need to keep the help window informed of which window is active. If you have only one frame window, the help window is automatically associated with that frame window.

To associate a secondary frame window with the help window, connect an event opening the secondary window to the *setAssociatedWindow* action of the IHelpWindow* part, passing the frame window as a parameter.

For example, if you open a secondary window when a push button is pressed, connect the IPushButton* part's *buttonClickEvent* feature to the IHelpWindow* part's *setAssociatedWindow* action. Then, connect the secondary IFrameWindow* part's *this* attribute to the *buttonClickEvent*-to-*setAssociatedWindow* connection to provide the frame window parameter.

Alternatively, if you create the secondary window dynamically with an IVBFactory* part, connect the IVBFactory* part's *newEvent* feature to the IHelpWindow* part's *setAssociatedWindow* action. The *newEvent* feature provides the frame window object as a parameter.

Preferred Features — IHelpWindow

addLibraries Adds a library or list of libraries to those already used by the Information Presentation Facility (IPF).

setActiveWindow
Sets the application window that the Information Presentation Facility (IPF) will treat a subsequent request for contextual or general help as coming from.

setAssociatedWindow
Associates an application frame window with the help window.

setFocus Sets the input focus to the window.

setTitle Sets the title bar text for the help window.

IHelpWindow

show	Show help panel.
showGeneralHelp	Show general help.
showHelpPanel	Show help panel.
showIndexHelp	Show index help.
this	A Pointer to the instance.

Features — IHelpWindow

Action	Attribute	Event
addLibraries ■	activeColor ■ ►	*activeColor*
applyBidiSettings ■	asDebugInfo	anyEvent
capturePointer ■	asString	*backgroundColor*
convertToGUIStyle ■	autoDeleteObject ■	*borderColor*
disable	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableUpdate	bidiSupported	*disabledForegroundColor*
dispatchRemainingHandlers ■	borderColor ■ ►	*enabled*
enable ■	characterSize	*focus*
enableUpdate ■	**communicationWindow**	*font*
handleException ■	**contentsWindow**	*foregroundColor*
helpWindow ■	**coverPageWindow**	gotFocusEvent ■
hide	defaultPushButton	*hiliteBackgroundColor*
hidePanelIds	disabledBackgroundColor ■ ►	*hiliteForegroundColor*
hideSourceEmphasis	disabledForegroundColor ■ ►	*inactiveColor*
isLayoutDistorted ■	enabled ■ ►	inputDisabledEvent ■
matchForMnemonic ■	enabledForNotification ■	inputEnabledEvent ■
notifyObservers ■	focus ►	lostFocusEvent ■
positionBehindSibling ■	font ■ ►	*position*
positionBehindSiblings	foregroundColor ■ ►	*shadowColor*
positionOnSiblings	frameWindow	*size*
postEvent ■	group	systemCommandEvent
refresh ■	handle	visibilityDisabledEvent ■
releasePointer	helpId ■	visibilityEnabledEvent ■
releasePresSpace ■	hiliteBackgroundColor ■ ►	***visible***
resetActiveColor	hiliteForegroundColor ■ ►	
resetBackgroundColor	id ■	
resetBorderColor	inactiveColor ■ ►	
resetDisabledBackgroundColor	**indexWindow**	
resetDisabledForegroundColor	itemProvider ■	
resetFont	layoutAdjustment	
resetForegroundColor	menu	
resetHiliteBackgroundColor	minimumSize ■	
resetHiliteForegroundColor	nativeRect	
resetInactiveColor	owner ■	
resetMinimumSize	parent ■	

IHelpWindow

Action	Attribute	Event
resetShadowColor	parentSize	
sendEvent ■	pointerCaptured	
setActiveWindow ■	position ■ ►	
setAssociatedWindow ■	presSpace	
setFocus	rect ■	
setHelpTable ■	**searchListWindow**	
setLayoutDistorted ■	shadowColor ■ ►	
setTitle ■	showing	
setUsingHelp ■	size ■ ►	
show ■	tabStop	
showContentsHelp ■	**this**	
showGeneralHelp ■	valid	
showHelpPanel ■	**viewedPagesWindow**	
showIndexHelp ■	**visible** ►	
showKeysHelp ■	visibleRectangle	
showPanelIds ■		
showSourceEmphasis ■		
showUsingHelp ■		

IHelpWindow

Chapter 36. IIconControl

Description: IBM icon control
Part type: Visual
Part file: vbbase
Header file: iiconctl.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IStaticText
IBitmapControl
IIconControl

Use an IIconControl* part to display an icon. Be aware that an icon is not resized when you place it on either a set canvas or an expandable row or column of a multicell canvas.

Portability — IIconControl

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor

© Copyright IBM Corp. 1995, 1996

- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Building Guidelines — IIconControl

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Assigning an Icon

Adding the Part

To add an IIconControl* part to a composite part, select an IIconControl* part from the parts palette. Then, drop the part on a canvas.

Assigning an Icon

To assign an icon to the control, set the **DLL name** and **Resource ID** fields on the IIconControl* part's **General** settings page.

Preferred Features — IIconControl

bitmap	Returns the handle to the bitmap.
icon	Returns the handle of the currently set icon.
limit	Returns the number of characters set by setLimit.
menu	Popup menu link.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

Features — IIconControl

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	alignment ■	anyEvent
convertToGUIStyle ■	asDebugInfo	*backgroundColor*
disable	asString	*borderColor*
disableFillBackground	autoDeleteObject ■	commandEvent
disableGroup	autoDestroyWindow ■	*disabledBackgroundColor*
disableHalftone	backgroundColor ■ ►	*disabledForegroundColor*
disableNotification	bidiSupported	*enabled*

IIconControl

Action	Attribute	Event
disableStrikeout	bitmap ■	*fillBackground*
disableTabStop	borderColor ■ ►	*fillColor*
disableUnderscore	characterSize	*focus*
disableUpdate	clipboardHasTextFormat	*font*
dispatchRemainingHandlers ■	defaultPushButton	*foregroundColor*
enable ■	disabledBackgroundColor ■ ►	gotFocusEvent ■
enableUpdate ■	disabledForegroundColor ■ ►	*halftone*
handleException ■	displaySize	*hiliteBackgroundColor*
hide	enabled ■ ►	*hiliteForegroundColor*
hideSourceEmphasis	enabledForNotification ■	*inactiveColor*
isLayoutDistorted ■	fillBackground ■ ►	inputDisabledEvent ■
matchForMnemonic ■	fillColor ■ ►	inputEnabledEvent ■
notifyObservers ■	focus ►	*isDigits*
positionBehindSibling ■	font ■ ►	*limit*
positionBehindSiblings	foregroundColor ■ ►	lostFocusEvent ■
positionOnSiblings	frameWindow	*position*
postEvent ■	group ■	*shadowColor*
refresh ■	halftone ■ ►	*size*
releasePointer	handle	*strikeout*
releasePresSpace ■	helpId ■	systemCommandEvent
resetActiveColor	hiliteBackgroundColor ■ ►	*text*
resetBackgroundColor	hiliteForegroundColor ■ ►	*textLength*
resetBorderColor	**icon** ■	*underscore*
resetDisabledBackgroundColor	id ■	*valueAsDouble*
resetDisabledForegroundColor	inactiveColor ■ ►	*valueAsInt*
resetFillColor	isDigits ►	*valueAsUnsigned*
resetFont	itemProvider ■	visibilityDisabledEvent ■
resetForegroundColor	layoutAdjustment	visibilityEnabledEvent ■
resetHiliteBackgroundColor	limit ■ ►	*visible*
resetHiliteForegroundColor	menu	
resetInactiveColor	minimumSize ■	
resetMinimumSize	nativeRect	
resetShadowColor	owner ■	
sendEvent ■	parent ■	
setFocus	parentSize	
setLayoutDistorted ■	pointerCaptured	
show ■	position ■ ►	
showSourceEmphasis ■	presSpace	
	rect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	strikeout ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	

IIconControl

Action	Attribute	Event
	underscore ■ ►	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

Chapter 37. IListBox

Description: IBM string list-box control
Part type: Visual
Part file: vbbase
Header file: ilistbox.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IBaseListBox
IListBox

Use an IListBox* part to provide a modifiable list of string choices for the user. The list box provides a set of string choices. An ICollectionViewListBox* part, by contrast, provides a view of an object collection. Your application can add or remove choices from the list.

Alternatively, for string specification from an entry field or selection use an IComboBox* part. You can provide only single selection with a combination box part, but you can provide multiple selection with a list box.

For selection of objects in a collection rather than strings, use an ICollectionViewComboBox* part or an ICollectionViewListBox* part.

By default, the user can select only one choice from the list. When the user selects a choice, any previously selected choice is no longer selected. You can change the behavior of the list box to allow multiple selection.

Portability — IListBox

© Copyright IBM Corp. 1995, 1996

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- disableExtendedSelect
- disableMultipleSelect
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- extendedSelect
- multipleSelect

Related Parts — IListBox

Selection parts:

- "ICollectionViewComboBox" on page 105
- "ICollectionViewListBox" on page 113
- "IComboBox" on page 123
- "IListBox" on page 199
- "INumericSpinButton" on page 243
- "ITextSpinButton" on page 329

Building Guidelines — IListBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting Initial Contents
- Defining Selection
- Enabling Tab Key Access

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Choice
- Adding a Choice
- Removing a Choice

Adding the Part

To add an IListBox* part to a composite part, select an IListBox* part from the parts palette. Then, drop the part on a canvas.

Setting Initial Contents

To define initial selection choices for the list box, enter the choices in the **Contents** field on the IListBox* part's **General** settings page.

Defining Selection

To allow the user to select only one choice, ensure that the **extendedSelection** field and the **multipleSelection** field on the IListBox* part's **Styles** settings page are both off. Set both fields to **Off** or, if you have not modified selection in the default style, to **Default**.

To allow extended selection, set the **extendedSelection** field to **On**. Set both **singleSelection** and **multipleSelection** to **Off**.

To allow multiple selection, set the **multipleSelection** field to **On**. Set both **singleSelection** and **extendedSelection** to **Off**.

Enabling Tab Key Access

To allow the user to reach the list box by using a tab key, set the **tabStop** field to **On** on the IListBox* part's **Styles** settings page.

Getting the Selected Choice

To copy the text of the selected item to an entry field when a push button is clicked, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to the IListBox* part's *itemText* action.
2. Connect the IListBox* part's *selection* aattribute to the *buttonClickEvent*-to-*itemText* connection to provide the index parameter for the action.
3. Connect the*buttonClickEvent*-to-*itemText* connection's **actionResult** to the IEntryField* part's *text* attribute.

Adding a Choice

To add a new choice from an entry field to the end of the list when a push button is pressed, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to the *addAsLast* action of the IListBox* part.
2. Connect the IEntryField* part's *text* attribute to the *text* parameter of the *buttonClickEvent*-to-*addAsLast* connection. This provides the entry field text as the parameter for the *addAsLast* action.

Removing a Choice

To remove the selected choice from the list box when a push button is pressed, do the following:

1. Connect the IPushButton* part's *buttonClickEvent* feature to the *remove* action of the IListBox* part.
2. Connect the IListBox* part's *selection* attribute to the *index* parameter of the *buttonClickEvent*-to-*remove* connection. This provides the selection index as the parameter for the *remove* action.

Preferred Features — IListBox

addAscending	Inserts the line of text in ascending sort order and returns the index of the item inserted.
addAsLast	Inserts one of the following, depending on which form of the virtual function you use.
enterEvent	Notification identifier provided to observers when the user `double` clicks on an item or presses enter on an item with the cursor in a list box.
menu	Popup menu link.

remove	Removes the specified item from the list box and returns the count of items that remain.
removeAll	Removes all items from the list box.
selection	Returns the 0-based index of the selected item.
selectionText	Returns the itemText using selection as the index.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

Features — IListBox

Action	Attribute	Event
add ■	activeColor ■ ►	*activeColor*
addAscending ■	asDebugInfo	anyEvent
addAsFirst ■	asString	*backgroundColor*
addAsLast ■	autoDeleteObject ■	*borderColor*
addDescending ■	autoDestroyWindow ■	commandEvent
applyBidiSettings ■	backgroundColor ■ ►	*disabledBackgroundColor*
capturePointer ■	bidiSupported	*disabledForegroundColor*
convertToGUIStyle ■	borderColor ■ ►	*enabled*
deselect ■	characterSize	enterEvent
deselectAll	count	*focus*
disable	defaultPushButton	*font*
disableDrawItem	disabledBackgroundColor ■ ►	*foregroundColor*
disableExtendedSelect	disabledForegroundColor ■ ►	gotFocusEvent ■
disableGroup	drawItem ■	*hiliteBackgroundColor*
disableMultipleSelect	empty	*hiliteForegroundColor*
disableNoAdjustPosition	enabled ■ ►	*inactiveColor*
disableNotification	enabledForNotification ■	inputDisabledEvent ■
disableTabStop	extendedSelect ■	inputEnabledEvent ■
disableUpdate	focus ►	lostFocusEvent ■
dispatchRemainingHandlers ■	font ■ ►	*numberOfSelections*
enable ■	foregroundColor ■ ►	*position*
enableUpdate ■	frameWindow	*selection*
handleException ■	group ■	*selectionText*
hide	handle	*shadowColor*
hideSourceEmphasis	helpId ■	*size*
isLayoutDistorted ■	hiliteBackgroundColor ■ ►	systemCommandEvent
isSelected ■	hiliteForegroundColor ■ ►	visibilityDisabledEvent ■
itemHandle ■	horizontalScroll	visibilityEnabledEvent ■
itemText ■	id ■	*visible*
locateText ■	inactiveColor ■ ►	
matchForMnemonic ■	itemProvider ■	
notifyObservers ■	layoutAdjustment	
positionBehindSibling ■	menu	
positionBehindSiblings	minimumCharacters ■	
positionOnSiblings	minimumRows ■	

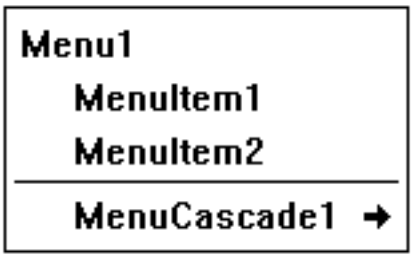
IListBox

Action	Attribute	Event
postEvent ■	minimumSize ■	
refresh ■	multipleSelect ■	
releasePointer	nativeRect	
releasePresSpace ■	noAdjustPosition ■	
remove ■	numberOfSelections ►	
removeAll	owner ■	
resetActiveColor	parent ■	
resetBackgroundColor	parentSize	
resetBorderColor	pointerCaptured	
resetDisabledBackgroundColor	position ■ ►	
resetDisabledForegroundColor	presSpace	
resetFont	rect ■	
resetForegroundColor	selection ■ ►	
resetHiliteBackgroundColor	selectionText ►	
resetHiliteForegroundColor	shadowColor ■ ►	
resetInactiveColor	showing	
resetMinimumSize	size ■ ►	
resetShadowColor	tabStop ■	
selectAll	**this**	
sendEvent ■	top ■	
setFocus	valid	
setItemHandle ■	visible ■ ►	
setItemHeight ■	visibleRectangle	
setItemText ■		
setLayoutDistorted ■		
show ■		
showSourceEmphasis ■		

Chapter 38. IMenu

Description: IBM menu control
Part type: Nonvisual
Part file: vbbase
Header file: imenu.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IMenu



Use an IMenu* part to provide a list of choices related to an object. Depending on your use of the menu, it is implemented by one of the following parts:

- IMenuBar*
- IPopUpMenu*
- ISubmenu*

An IMenuBar* part is a menu of choices for objects in a window. The choices appear in a horizontal bar beneath the title bar. Each choice opens a pull-down menu.

An IPopUpMenu* part is a menu of context-sensitive choices for one or more objects. The choices appear in a pop-up window near the objects.

An ISubmenu* part is a list of action, routing, and settings choices in a menu opened from another menu. The menu is either a pull-down menu opened from a menu bar or a cascade menu opened from another menu.

The user can select a choice, or menu item, to apply the choice to the object.

© Copyright IBM Corp. 1995, 1996

Portability — IMenu

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IMenu

Menu parts:

- "IMenu" on page 205
- "IMenuCascade" on page 211
- "IMenuItem" on page 215
- "IMenuSeparator" on page 219

Frame parts:

- "IFrameWindow" on page 167
- "IMenu" on page 205
- "IScrollBar" on page 287
- "ITitle" on page 337
- "IToolBar" on page 343
- "IVBInfoArea" on page 385

Building Guidelines — IMenu

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding a Menu
- Reordering Menu Items

Adding the Part

To add an IMenu* part to a composite part, select an IMenu* part from the parts palette. Then, drop the part on the free-form surface or on another IMenu* part.

Adding a Menu

To add the menu to another part, do one of the following:

- For a menu bar on a frame window, connect the IMenu* part's *this* attribute to the IFrameWindow* part's *menu* attribute. The menu bar is implemented by an IMenuBar part in your application. Leave room for the generated menu bar at the top of the client area.
- For a pop-up menu on a control, connect the IMenu* part's *this* attribute to the control part's *menu* attribute. The menu is implemented by an IPopUpMenu part in your application.
- For a pull-down menu from a menu bar or for a cascade menu from another menu, drop an IMenu* part on the target menu instead of on the free-form surface. The following happens automatically:
  - The new IMenu* part is placed on the free-form surface.
  - An IMenuCascade* part is added to the target menu.
  - The new IMenu* part's *this* attribute is connected to the IMenuCascade* part's *menu* attribute.

  Alternatively, drop the new IMenu* part on the free-form surface. Then, connect the new IMenu* part's *this* attribute to an existing IMenuCascade* part's *menu* attribute.

  The new menu is implemented by an ISubmenu part in your application.

Reordering Menu Items

To move the menu items within your menu, drag them within the IMenu* part.

Preferred Features — IMenu

this A Pointer to the instance.

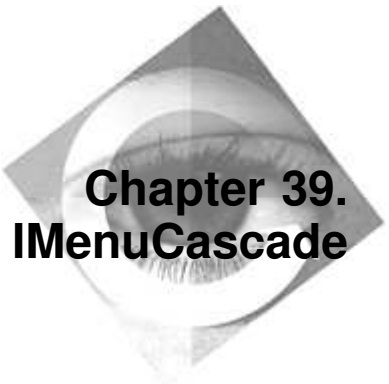
Features — IMenu

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableNotification	autoDestroyWindow ■	commandEvent
disableUpdate	backgroundColor ■ ►	*disabledBackgroundColor*
dispatchRemainingHandlers ■	bidiSupported	*disabledForegroundColor*
enable ■	borderColor ■ ►	*enabled*
enableUpdate ■	characterSize	*focus*
handleException ■	defaultPushButton	*font*
hide	disabledBackgroundColor ■ ►	*foregroundColor*
hideSourceEmphasis	disabledForegroundColor ■ ►	gotFocusEvent ■
isLayoutDistorted ■	enabled ■ ►	*hiliteBackgroundColor*
matchForMnemonic ■	enabledForNotification ■	*hiliteForegroundColor*
notifyObservers ■	focus ►	*inactiveColor*
positionBehindSibling ■	font ■ ►	inputDisabledEvent ■
positionBehindSiblings	foregroundColor ■ ►	inputEnabledEvent ■
positionOnSiblings	frameWindow	lostFocusEvent ■
postEvent ■	group	*position*
refresh ■	handle	*shadowColor*
releasePointer	hiliteBackgroundColor ■ ►	*size*
releasePresSpace ■	hiliteForegroundColor ■ ►	systemCommandEvent
resetActiveColor	id ■	visibilityDisabledEvent ■
resetBackgroundColor	inactiveColor ■ ►	visibilityEnabledEvent ■
resetBorderColor	itemProvider ■	*visible*
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	rect ■	
setLayoutDistorted ■	shadowColor ■ ►	
show ■	showing	
showSourceEmphasis ■	size ■ ►	
	tabStop	

IMenu

Action	Attribute	Event
	this	
	valid	
	visible ■ ►	
	visibleRectangle	

IMenu



Chapter 39. IMenuCascade

Description:	IBM Cascade Menu Item	**Derivation**:	
Part type:	Visual	*IBase*	
Part file:	vbbase	IMenuItem	
Header file:	imnItem.hpp	**IMenuCascade**	

Use an IMenuCascade* part to represent a set of closely related menu choices as a high-level menu choice. The closely related menu choices can then be presented in a submenu. This keeps menus shorter and defers detailed choices from the higher-level menu. You can present the choice to the user as text or as a bitmap.

The user can select a menu cascade to open a submenu.

Portability — IMenuCascade

Features active with restriction on Windows:

- framed
- noDismiss

Related Parts — IMenuCascade

Menu parts:

- "IMenu" on page 205
- "IMenuCascade"
- "IMenuItem" on page 215
- "IMenuSeparator" on page 219

Building Guidelines — IMenuCascade

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Assigning a Bitmap
- Adding a Menu

© Copyright IBM Corp. 1995, 1996

- Defining Menu Information Text
- Reordering Menu Items
- Setting the Initial State

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Disabling the Part
- Enabling the Part

Adding the Part

To add an IMenuCascade* part to a composite part, select an IMenuCascade* part from the parts palette. Then, drop the part on an IMenu* part.

Alternatively, you can easily add an IMenuCascade* part and attach a submenu by dropping a new IMenu* part on the target IMenu* part.

Defining Text for the Part

To define text for the cascade menu item, select the **Text** choice in the **Menu item type** field on the IMenuCascade* part's **General** settings page. Then, enter the text you want in the **Text** field.

If you want to establish a mnemonic for the cascade menu item, type a tilde (~) before the mnemonic character: For Windows, Visual Builder maps the tilde to an ampersand (&). You can type an ampersand for Windows applications, but using the tilde provides portability.

Assigning a Bitmap

To assign a bitmap to the cascade menu item, select the **Graphic** choice in the **Menu item type** field on the IMenuCascade* part's **General** settings page. Then, identify the bitmap in the **DLL name** and **Resource ID** fields.

Adding a Menu

To associate the cascade menu item with a submenu, connect the submenu part's *this* attribute to the IMenuCascade* part's *menu* attribute. The submenu opens when the user selects the cascade menu item.

Defining Menu Information Text

If the cascade menu item is a choice on a menu bar, you can define brief information for the menu item. This text appears in the frame window information area when the user selects the menu choice. To define information text for the cascade menu item, enter the text you want in the **Info area text** field on the IMenuCascade* part's **General** settings page.

Reordering Menu Items

To move the cascade menu item within your menu, drag the IMenuCascade* part to a new location in the IMenu* part.

Setting the Initial State

To make the cascade menu item initially available, ensure that the **Disabled** field on the IMenuCascade* part's **General** settings page is not set. To make the cascade menu item initially unavailable, set the **Disabled** field.

Disabling the Part

To disable the cascade menu item when an event occurs, such as selection of another item, do the following:

1. Connect the event to the IMenuCascade* part's *disabled* attribute.
2. Open settings for the connection, press the **Set parameters** push button, and check the **Disabled** setting.

Enabling the Part

To enable the cascade menu item when an event occurs, such as selection of another item, connect the event to the IMenuCascade* part's *disabled* attribute with the **Disabled** setting not checked.

Preferred Features — IMenuCascade

bitmap	Returns the bit-map handle of the item.
checked	If the attribute checked is set, `true` is returned.
menu	Pulldown menu.
selectable	If the menu item is selectable, returns `true`.
text	Returns an `IString` representing the text to be displayed.
this	A Pointer to the instance.

Features — IMenuCascade

Action	Attribute	Event
convertToGUIStyle ■	asDebugInfo	commandEvent
disable	asString	
enable ■	bitmap ■	
setSeparator	checked ■	
	commandType ■	
	disabled ■	
	drawItem ■	
	framed ■	
	highlighted ■	
	id	

IMenuCascade

Action	Attribute	Event
	index ■	
	isBitmap	
	isText	
	layoutType ■	
	menu	
	noDismiss ■	
	selectable ■	
	separator	
	submenu	
	submenuHandle ■	
	text ■	
	this	

Chapter 40. IMenuItem

Description: IBM menu item
Part type: Visual
Part file: vbbase
Header file: imnitem.hpp

Derivation:
IBase
IMenuItem

Use an IMenuItem* part to present a menu choice related to an object. You can present the choice to the user as text or as a bitmap.

The user can select a choice, or menu item, to apply the choice to an object.

Portability — IMenuItem

Features active with restriction on Windows:

- framed
- noDismiss

Related Parts — IMenuItem

Menu parts:

- "IMenu" on page 205
- "IMenuCascade" on page 211
- "IMenuItem"
- "IMenuSeparator" on page 219

Building Guidelines — IMenuItem

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Assigning a Bitmap
- Defining an Accelerator
- Defining Menu Information Text
- Preselecting the Choice

© Copyright IBM Corp. 1995, 1996

- Reordering Menu Items
- Setting the Initial State

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Disabling the Part
- Enabling the Part
- Starting the Requested Action

Adding the Part

To add an IMenuItem* part to a composite part, select an IMenuItem* part from the parts palette. Then, drop the part on an IMenu* part.

Defining Text for the Part

To define text for the menu item, select the **Text** choice in the **Menu item type** field on the IMenuItem* part's **General** settings page. Then, enter the text you want in the **Text** field.

If you want to establish a mnemonic for the menu item, type a tilde (~) before the mnemonic character: For Windows, Visual Builder maps the tilde to an ampersand (&). You can type an ampersand for Windows applications, but using the tilde provides portability.

Assigning a Bitmap

To assign a bitmap to the menu item, select the **Graphic** choice in the **Menu item type** field on the IMenuItem* part's **General** settings page. Then, identify the bitmap in the **DLL name** and **Resource ID** fields.

Defining an Accelerator

To define an accelerator for the menu item, specify the keys you want to use in the **accelerator** field on the IMenuItem* part's **General** settings page. If you want the accelerator identified on the menu item, set the **Show text on menu** field.

Defining Menu Information Text

If the menu item is a choice on a menu bar, you can define brief information for the menu item. This text appears in the frame window information area when the user selects the menu choice. To define information text for the menu item, enter the text you want in the **Info area text** field on the IMenuItem* part's **General** settings page.

Preselecting the Choice

To preselect the menu item, set the **Checked** field on the IMenuItem* part's **General** settings page.

Reordering Menu Items

To move the menu item within your menu, drag the IMenuItem* part to a new location in the IMenu* part.

Setting the Initial State

To make the menu item initially available, ensure that the **Disabled** field on the IMenuItem* part's **General** settings page is not set. To make the menu item initially unavailable, set the **Disabled** field.

Disabling the Part

To disable the menu item when an event occurs, such as selection of another item, do the following:

1. Connect the event to the IMenuItem* part's *disabled* attribute.
2. Open settings for the connection, press the **Set parameters** push button, and check the **Disabled** setting.

Enabling the Part

To enable the menu item when an event occurs, such as selection of another item, connect the event to the IMenuItem* part's *disabled* attribute with the **Disabled** setting not checked.

Starting the Requested Action

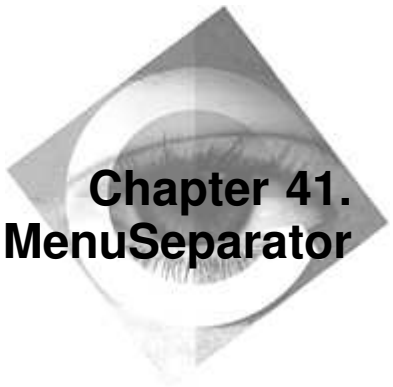
To start an action when the user selects the menu item, connect the IMenuItem* part's *commandEvent* feature to the target action.

Preferred Features — IMenuItem

bitmap	Returns the bit-map handle of the item.
checked	If the attribute checked is set, `true` is returned.
commandEvent	Notification identifier provided to observers when a WM_COMMAND event occurs.
disable	Disable the menu item.
enable	Enable the menu item.
text	Returns an `IString` representing the text to be displayed.
this	A Pointer to the instance.

Features — IMenuItem

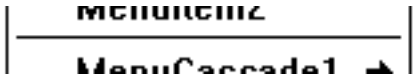
Action	Attribute	Event
convertToGUIStyle ■	asDebugInfo	**commandEvent**
disable	asString	
enable ■	**bitmap** ■	
setSeparator	**checked** ■	
	commandType ■	
	disabled ■	
	drawItem ■	
	framed ■	
	highlighted ■	
	id	
	index ■	
	isBitmap	
	isText	
	layoutType ■	
	noDismiss ■	
	selectable ■	
	separator	
	submenu	
	submenuHandle ■	
	text ■	
	this	



Chapter 41. IMenuSeparator

Description: IBM Separator Menu Item
Part type: Nonvisual
Part file: vbbase
Header file: imnItem.hpp

Derivation:
IBase
IMenuItem
IMenuSeparator



Use an IMenuSeparator* part to mark the beginning or end of a group of related menu choices.

The user can recognize groups of related menu choices by noting the separators between the groups.

Portability — IMenuSeparator

Features active with restriction on Windows:

- framed
- noDismiss

Related Parts — IMenuSeparator

Menu parts:

- "IMenu" on page 205
- "IMenuCascade" on page 211
- "IMenuItem" on page 215
- "IMenuSeparator"

Building Guidelines — IMenuSeparator

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Reordering Menu Items

© Copyright IBM Corp. 1995, 1996

IMenuSeparator

Adding the Part

To add an IMenuSeparator* part to a composite part, select an IMenuSeparator* part from the parts palette. Then, drop the part on an IMenu* part.

Reordering Menu Items

To move the menu separator within your menu, drag the IMenuSeparator* part to a new location in the IMenu* part.

Preferred Features — IMenuSeparator

this A Pointer to the instance.

Features — IMenuSeparator

Action	Attribute	Event
convertToGUIStyle ■	asDebugInfo	commandEvent
disable	asString	
enable ■	bitmap ■	
setSeparator	checked ■	
	commandType ■	
	disabled ■	
	drawItem ■	
	framed ■	
	highlighted ■	
	id	
	index ■	
	isBitmap	
	isText	
	layoutType ■	
	noDismiss ■	
	selectable ■	
	separator	
	submenu	
	submenuHandle ■	
	this	

Chapter 42. IMessageBox

Description: IBM message dialog window
Part type: Class
Part file: vbbase
Header file: imsgbox.hpp

Derivation:
IBase
IVBase
IMessageBox

Use an IMessageBox* part to display information about activity or problems. A message box is a pop-up window with text information and button choices. For example, your application can do the following:

- Inform the user about what a task is doing
- Display an error message to the user

The user can note the information or error presented in a message. If given a choice, the user can select an action in response to the message.

Related Parts — IMessageBox

Dialog parts:

- "IMessageBox"
- "IVBFileDialog" on page 373
- "IVBFontDialog" on page 381

Building Guidelines — IMessageBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Assigning Ownership

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Showing Exceptions

© Copyright IBM Corp. 1995, 1996

- Showing Messages

Adding the Part

To add an IMessageBox* part to a composite part, select an IMessageBox* part from the parts palette. Then, drop the part on the free-form surface.

Defining Text for the Part

To define a title for your message box, enter text in the **Title** field on the IMessageBox* part's **General** settings page.

Assigning Ownership

If you have only one frame window in your view, the message box is automatically owned by that frame window.

To relate the message box to a second frame window, reference the IFrameWindow* part's **Subpart name**, which appears on its **General** settings page. Enter the IFrameWindow* part's subpart name, prefixed with "i," in the **Owner** field on the IMessageBox* part's **General** settings page.

Showing Exceptions

To report exceptions in the message box, connect *exceptionOccurred* on each connection that could yield an exception to the IMessageBox* part's *showException* action.

Showing Messages

To show messages in the message box, do the following:

1. Connect an event for each message to the IMessageBox* part's *show* action.
2. Open settings for the connection and press the **Set parameters** push button. Then, enter the message text and a message severity. The message text must be entered as a C++ string enclosed in quotation marks. The severity must be one of the following enumeration values:

   - information
   - warning
   - action
   - critical
   - catastrophic

   The IMessageBox class description in the *IBM Open Class Library Reference* provides information about these Severity enumeration values.

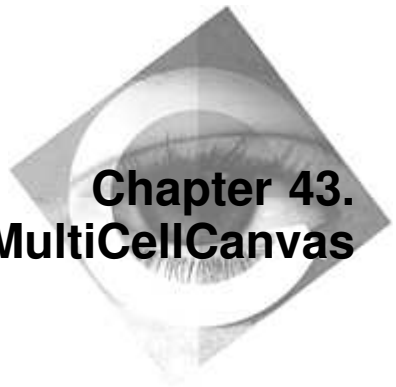
Preferred Features — IMessageBox

setTitle	Sets the message box's title.
show	Shows a message box.
showCustomized	Shows a customized message box.
showErrorInfo	Shows a message box.
showException	Shows a message box.
this	A Pointer to the instance.

Features — IMessageBox

Action	Attribute	Type
setTitle ■	asDebugInfo	`IString`
show ■	asString	`IString`
showCustomized ■	**this**	`IMessageBox*`
showErrorInfo ■		
showException ■		

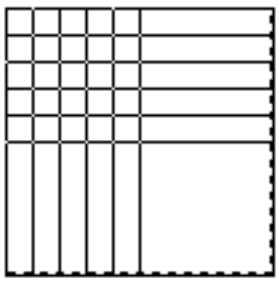
IMessageBox



Chapter 43. IMultiCellCanvas

Description: IBM resizable multicell canvas
Part type: Visual
Part file: vbbase
Header file: imcelcv.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
IMultiCellCanvas



Use an IMultiCellCanvas* part for flexible placement of controls in rows and columns.

You can use an IMultiCellCanvas* part in another composer part, such as the following:

- IFrameWindow* client area
- IViewPort*
- IMultiCellCanvas*
- ISplitCanvas*
- IVBNotebookPage*

Then, you can add controls to the canvas within the other composer part.

You can also use a multicell canvas as the base part for a composite part. Then, you can add the canvas with its controls as a subpart in another composite part.

The cells in a multicell canvas are adjusted for varying control text length. This provides built-in flexibility for language translation. If you build the multicell canvas with some expandable rows and columns, controls in the cells remain uniformly spaced when the user resizes the canvas.

© Copyright IBM Corp. 1995, 1996

Alternatively, use one of the following canvas parts:

- ICanvas* for a simple canvas
- IDrawingCanvas* for graphic objects
- ISetCanvas* for canvas cells in uniform rows or columns
- ISplitCanvas* for split windows

The user interacts with controls placed on the canvas, not with the canvas itself.

Portability — IMultiCellCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IMultiCellCanvas

Canvas parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

Client parts:

- “ICanvas” on page 83

- "IDrawingCanvas" on page 145
- "IMultiCellCanvas" on page 225
- "IMultiLineEdit" on page 231
- "INotebook" on page 237
- "ISetCanvas" on page 293
- "ISplitCanvas" on page 307
- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — IMultiCellCanvas

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding Controls
- Spanning Controls Across Cells
- Adding a Row
- Removing a Row
- Adding a Column
- Removing a Column
- Defining Expansion Space
- Defining Margins

Adding the Part

To add an IMultiCellCanvas* part to a composite part, select an IMultiCellCanvas* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- An IViewPort* part
- An IVBNotebookPage* part
- A canvas part

If you drop the IMultiCellCanvas* part on a view port, set the **expandableViewWindow** setting on the IViewPort* part's **Styles** settings page. This enables the IViewPort* part to automatically add or remove scroll bars based on the minimum size of cells in the IMultiCellCanvas* part.

Adding Controls

To add controls to the multicell canvas, select the control parts from the parts palette and drop them on the IMultiCellCanvas* part. Place each control in a cell. The cell is sized to accommodate the largest minimum size of controls in the row and column.

To add sets of controls, drop an ISetCanvas* part on the IMultiCellCanvas* part and drop controls on the set canvas. If you want to add a set of controls in a group box, use an ISetCanvas* part and define it as a group box. This gives the appearance you want without sacrificing automatic sizing of controls.

If you want margins at the edges of your canvas, do not place controls in the first row or the first column. Reserve them as left and top margins.

If you drop a control in the expansion space initially provided on the right and bottom of the canvas, a new row or column is added for the control.

Spanning Controls Across Cells

To make a control span multiple rows or columns, do the following:

1. Select the control.
2. Press an Alt key and drag a corner handle of the control to the far side of the row or column that the control is to end in.

Adding a Row

To add a row to the canvas, do the following:

1. Open the IMultiCellCanvas* part's contextual menu and select **Rows**.
2. Select **Add row after** or **Add row before** in the submenu.

Removing a Row

To remove a row from the canvas, do the following:

1. Open the IMultiCellCanvas* part's contextual menu and select **Rows**.
2. Select **Delete row** in the submenu.

Adding a Column

To add a column to the canvas, do the following:

1. Open the IMultiCellCanvas* part's contextual menu and select **Columns**.
2. Select **Add column after** or **Add column before** in the submenu.

Removing a Column

To remove a column from the canvas, do the following:

1. Open the IMultiCellCanvas* part's contextual menu and select **Columns**.
2. Select **Delete column** in the submenu.

Defining Expansion Space

To make controls remain evenly spaced when the user resizes the canvas, change the **Expand** value for rows and columns between controls to y (yes). Change these values in the **Rows** table and the **Columns** table on the IMultiCellCanvas* part's **General** settings page.

The initial expansion space on the right or bottom of the canvas is removed when you make a column or row expandable.

You can also add and remove rows and columns in these tables.

Defining Margins

To create right and bottom margins for the multicell canvas, add a row and column at the right and bottom. Do not place any controls in these margins.

Preferred Features — IMultiCellCanvas

addToCell	Specifies the starting cell into which a window is placed.
dragLines	Queries whether a multiple-cell canvas has the style dragLines set.
gridLines	Queries whether a multiple-cell canvas has the style gridLines set.
menu	Popup menu link.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.
windowInCell	Returns the child window occupying the specified cell.

Features — IMultiCellCanvas

Action	Attribute	Event
addToCell ■	activeColor ■ ►	*activeColor*
applyBidiSettings ■	asDebugInfo	anyEvent
capturePointer ■	asString	*backgroundColor*
columnWidth ■	autoDeleteObject ■	*borderColor*
convertToGUIStyle ■	autoDestroyWindow ■	commandEvent
disable	backgroundColor ■ ►	*disabledBackgroundColor*
disableDragLines	bidiSupported	*disabledForegroundColor*
disableGridLines	borderColor ■ ►	*enabled*
disableGroup	characterSize	*focus*
disableNotification	**defaultCell** ■	*font*
disableTabStop	defaultPushButton	*foregroundColor*
disableUpdate	disabledBackgroundColor ■ ►	gotFocusEvent ■
dispatchRemainingHandlers ■	disabledForegroundColor ■ ►	*hiliteBackgroundColor*
enable ■	**dragLines** ■	*hiliteForegroundColor*
enableUpdate ■	enabled ■ ►	*inactiveColor*
handleException ■	enabledForNotification ■	inputDisabledEvent ■

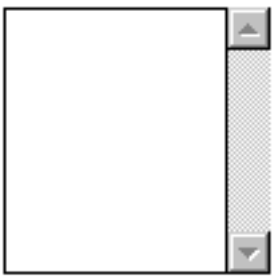
IMultiCellCanvas

Action	Attribute	Event
hide	focus ►	inputEnabledEvent ■
hideSourceEmphasis	font ■ ►	lostFocusEvent ■
isColumnExpandable ■	foregroundColor ■ ►	*position*
isLayoutDistorted ■	frameWindow	*shadowColor*
isRowExpandable ■	**gridLines** ■	*size*
matchForMnemonic ■	group ■	systemCommandEvent
notifyObservers ■	handle	visibilityDisabledEvent ■
positionBehindSibling ■	helpId ■	visibilityEnabledEvent ■
positionBehindSiblings	hiliteBackgroundColor ■ ►	*visible*
positionOnSiblings	hiliteForegroundColor ■ ►	
postEvent ■	id ■	
refresh ■	inactiveColor ■ ►	
releasePointer	itemProvider ■	
releasePresSpace ■	layoutAdjustment	
removeFromCell ■	menu	
resetActiveColor	minimumSize ■	
resetBackgroundColor	nativeRect	
resetBorderColor	origDefaultButtonHandle	
resetDisabledBackgroundColor	owner ■	
resetDisabledForegroundColor	parent ■	
resetFont	parentSize	
resetForegroundColor	pointerCaptured	
resetHiliteBackgroundColor	position ■ ►	
resetHiliteForegroundColor	presSpace	
resetInactiveColor	rect ■	
resetMinimumSize	shadowColor ■ ►	
resetShadowColor	showing	
rowHeight ■	size ■ ►	
sendEvent ■	tabStop ■	
setColumnWidth ■	**this**	
setFocus	valid	
setLayoutDistorted ■	visible ■ ►	
setRowHeight ■	visibleRectangle	
show ■		
showSourceEmphasis ■		
windowInCell ■		

Chapter 44. IMultiLineEdit

Description: IBM multiple-line edit control
Part type: Visual
Part file: vbbase
Header file: imle.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IMultiLineEdit



Use an IMultiLineEdit* (MLE) part to let the user enter or view text data in a multiline entry field. Alternatively, use an IEntryField* part for single-line data.

The user can type or place text into the multiline entry field.

Portability — IMultiLineEdit

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- disableWordWrap
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor

© Copyright IBM Corp. 1995, 1996

- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- wordWrap

Related Parts — IMultiLineEdit

Data entry parts:

- "IEntryField" on page 153
- "IMultiLineEdit" on page 231

Client parts:

- "ICanvas" on page 83
- "IDrawingCanvas" on page 145
- "IMultiCellCanvas" on page 225
- "IMultiLineEdit" on page 231
- "INotebook" on page 237
- "ISetCanvas" on page 293
- "ISplitCanvas" on page 307
- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — IMultiLineEdit

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Loading Initial Text
- Preventing User Changes
- Enabling Tab Key Access

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Loading Text
- Saving Text

Adding the Part

To add an IMultiLineEdit* part to a composite part, select an IMultiLineEdit* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- A canvas part
- An IVBNotebookPage* part

Defining Text for the Part

To define constant initial text for the MLE, enter the text you want in the **Text** field on the IMultiLineEdit* part's **General** settings page.

Loading Initial Text

To load variable initial text into the MLE, connect your data source to the IMultiLineEdit* part's *text* attribute. Your application should make this connection before it shows the frame window.

Preventing User Changes

To prevent the user from entering or modifying text, select the **readOnly** field on the IMultiLineEdit* part's **Styles** settings page.

Enabling Tab Key Access

To allow the user to reach the MLE by using a tab key, set the **tabStop** field to **On** on the IMultiLineEdit* part's **Styles** settings page.

Loading Text

To load the MLE when the user presses the **OK** push button in a file dialog, do the following:

1. Connect the IVBFileDialog* part's *pressedOK* event to the IMultiLineEdit* part's *importFromFile* action.
2. Connect the IVBFileDialog* part's *fileName* attribute to the *fileName* parameter of the *pressedOK*-to-*importFromFile* connection.

Saving Text

To save the contents of the MLE to a file when the user presses the **OK** push button in a file dialog, do the following:

1. Connect the IVBFileDialog* part's *pressedOK* event to the IMultiLineEdit* part's *exportToFile* action.
2. Connect the IVBFileDialog* part's *fileName* attribute to the *fileName* parameter of the *pressedOK*-to-*exportToFile* connection.

Preferred Features — IMultiLineEdit

addAsLast	Inserts the specified text into the MLE at the end of the current text, but does not change the cursor position.
copy	Copies the currently selected text from the MLE to the clipboard.
cut	Copies the currently selected text from the MLE to the clipboard and then deletes the selected text from the MLE.
exportToFile	Saves the contents of the MLE to the specified file and returns the number of bytes written to the file.
importFromFile	Inserts the contents of the specified file into the MLE at the current cursor position and returns the number of bytes imported.
limit	Returns the currently set number of bytes the MLE can hold.
menu	Popup menu link.
paste	Inserts the contents of the clipboard into the MLE at the cursor position.
removeAll	Deletes the entire contents of the MLE.
selectedText	Returns the selected text string.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

Features — IMultiLineEdit

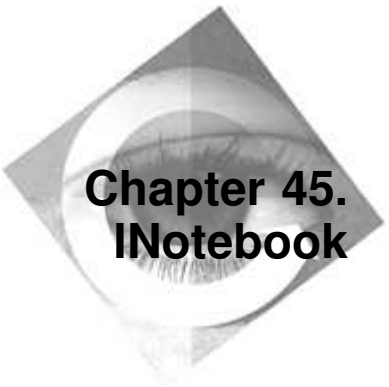
Action	Attribute	Event
add ■	activeColor ■ ►	*activeColor*
addAsLast ■	asDebugInfo	anyEvent
addAtOffset ■	asString	*backgroundColor*
addLine ■	autoDeleteObject ■	*borderColor*
addLineAsLast ■	autoDestroyWindow ■	commandEvent
applyBidiSettings ■	backgroundColor ■ ►	*disabledBackgroundColor*
capturePointer ■	bidiSupported	*disabledForegroundColor*
clear ■	borderColor ■ ►	*enabled*
convertToGUIStyle ■	**changed**	*focus*
copy ■	characterSize	*font*
cut ■	clipboardHasTextFormat	*foregroundColor*
disable	**cursorLinePosition** ■	gotFocusEvent ■
disableDataUpdate	**cursorPosition** ■	*hiliteBackgroundColor*
disableGroup	defaultPushButton	*hiliteForegroundColor*

IMultiLineEdit

Action	Attribute	Event
disableNotification	disabledBackgroundColor ■ ►	*inactiveColor*
disableTabStop	disabledForegroundColor ■ ►	inputDisabledEvent ■
disableUpdate	displaySize	inputEnabledEvent ■
disableWordWrap	**editRegionHeight** ■	*isDigits*
discard	**editRegionWidth** ■	***limit***
dispatchRemainingHandlers ■	enabled ■ ►	lostFocusEvent ■
enable ■	enabledForNotification ■	*position*
enableUpdate ■	focus ►	*shadowColor*
exportSelectedTextToFile ■	font ■ ►	*size*
exportToFile ■	foregroundColor ■ ►	systemCommandEvent
handleException ■	frameWindow	*text*
hide	group ■	*textLength*
hideSourceEmphasis	handle	*valueAsDouble*
importFromFile ■	**hasSelectedText**	*valueAsInt*
isLayoutDistorted ■	helpId ■	*valueAsUnsigned*
matchForMnemonic ■	hiliteBackgroundColor ■ ►	visibilityDisabledEvent ■
notifyObservers ■	hiliteForegroundColor ■ ►	visibilityEnabledEvent ■
paste	id ■	*visible*
positionBehindSibling ■	inactiveColor ■ ►	***writeable***
positionBehindSiblings	isDigits ►	
positionOnSiblings	itemProvider ■	
postEvent ■	layoutAdjustment	
refresh ■	**limit** ■ ►	
releasePointer	menu	
releasePresSpace ■	minimumSize ■	
removeAll	nativeRect	
removeLine ■	**numberOfLines**	
resetActiveColor	owner ■	
resetBackgroundColor	parent ■	
resetBorderColor	parentSize	
resetChangedFlag	pointerCaptured	
resetDisabledBackgroundColor	position ■ ►	
resetDisabledForegroundColor	presSpace	
resetFont	rect ■	
resetForegroundColor	**selectedRange**	
resetHiliteBackgroundColor	**selectedText**	
resetHiliteForegroundColor	**selectedTextLength**	
resetInactiveColor	shadowColor ■ ►	
resetMinimumSize	showing	
resetShadowColor	size ■ ►	
selectRange ■	tabStop ■	
sendEvent ■	text ■ ►	
setChangedFlag ■	textLength ►	
setEditRegion	**this**	
setFocus	**top** ■	
setLayoutDistorted ■	**undoable**	
setTab ■	valid	

IMultiLineEdit

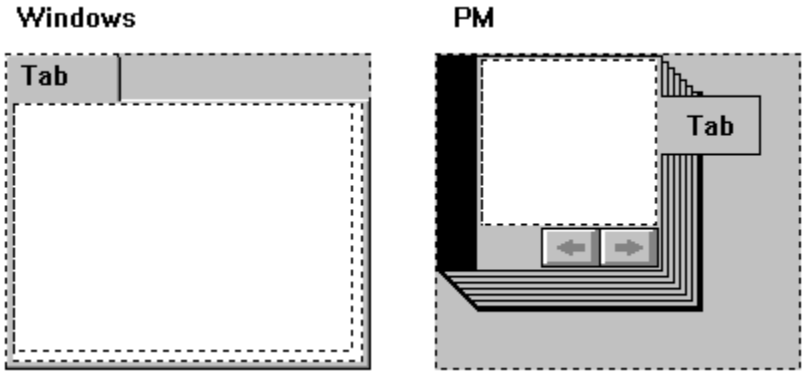
Action	Attribute	Event
show ■	valueAsDouble ■ ►	
showSourceEmphasis ■	valueAsInt ■ ►	
undo	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleLines	
	visibleRectangle	
	wordWrap ■	
	writeable ■ ►	



Chapter 45. INotebook

Description: IBM notebook control
Part type: Visual
Part file: vbbase
Header file: inotebk.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
INotebook



Use an INotebook* part for settings choices or for information that can be organized into sections. A notebook presents information on pages and marks the first page of each section with a tab.

You can use either a notebook style that is native to the system or a style that is compatible with the OS/2 Presentation Manager (PM) notebook. The PM-compatible notebook provides choices of orientation, binding, and page tab shape. It also supports status text.

The user can do the following with a notebook:

- Turn the pages by clicking on page buttons
- Move directly to a section by selecting a tab
- View or change information by using other controls on the notebook pages

Portability — INotebook

© Copyright IBM Corp. 1995, 1996

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — INotebook

Notebook parts:

- “INotebook” on page 237
- “IVBNotebookPage” on page 391

Client parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “IMultiLineEdit” on page 231
- “INotebook” on page 237
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307
- “IVBContainerControl” on page 359
- “IViewPort” on page 399

Building Guidelines — INotebook

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting PM Compatibility
- Selecting the Orientation
- Selecting Notebook Binding
- Selecting the Tab Shape
- Aligning Status Text
- Adding a Page

Adding the Part

To add an INotebook* part to a composite part, select an INotebook* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- A canvas part

Selecting PM Compatibility

To provide a PM-compatible notebook, set the **pmCompatible** field on the INotebook* part's **Styles** settings page.

Selecting the Orientation

To define the orientation of the notebook, select the back page and major tab orientation you want in the **Layout** field on the INotebook* part's **General** settings page.

This setting is not applicable for the native Windows notebook. The notebook does not display page depth. The notebook is always oriented horizontally with tabs at the top.

Selecting Notebook Binding

To define the binding for the notebook, select the **Solid** or **Spiral** choice in the **Binding** field on the INotebook* part's **General** settings page.

This setting is not applicable for the native Windows notebook.

Selecting the Tab Shape

To define tab appearance for the notebook, select **Justification** choices on the INotebook* part's **General** settings page:

- Select the shape in the **Tab shape** field

- Select the tab text alignment in the **Tab** field

By default, the underlying class determines the tab size. To specify a tab size for the notebook, set the **majorTab** and **minorTab** fields on the INotebook* part's **General** settings page. Specify both **Width** and **Height** fields to set a tab size. If both fields are not specified, neither field is set.

This setting is not applicable for the native Windows notebook. The tabs are always rounded.

Aligning Status Text

To align the status text at the bottom of each notebook page, select the alignment you want in the **Status area Justification** field on the INotebook* part's **General** settings page.

This setting is not applicable for the native Windows notebook. The notebook does not have a status area.

Adding a Page

To add a page to the notebook, select an add page choice on the contextual menu of either the INotebook* part or an IVBNotebookPage* part.

- To add a page to an empty notebook, select **Add initial page** on the INotebook* part's contextual menu.
- To add a page before or after the top page of a notebook, select **Add page** on the INotebook* part's contextual menu. Then, select **Before top page** or **After top page** on the submenu.
- To add a page before or after the currently selected page, select **Add page** on the IVBNotebookPage* part's contextual menu. Then, select **Before** or **After** on the submenu.

An IVBNotebookPage* part is placed in the notebook.

Preferred Features — INotebook

menu	Popup menu link.
orientation	Returns the current orientation of the notebook, which is the location of the major tabs in relation to the back pages' corner.
pageSize	Returns the size of the notebook's pages.
pagesToEnd	Returns the number of pages in the notebook from a specified notebook page to the end of the notebook.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

totalPages	Returns the total number of pages in the notebook.
turnToPage	Moves a specified page to the top of the notebook.

Features — INotebook

Action	Attribute	Event
addFirstPage ■	activeColor ■ ►	*activeColor*
addLastPage ■	asDebugInfo	anyEvent
addPageAfter ■	asString	*backgroundColor*
addPageBefore ■	autoDeleteObject ■	*borderColor*
applyBidiSettings ■	autoDestroyWindow ■	commandEvent
capturePointer ■	backgroundColor ■ ►	*disabledBackgroundColor*
convertToGUIStyle ■	bidiSupported	*disabledForegroundColor*
disable	**binding** ■	*enabled*
disableGroup	borderColor ■ ►	*focus*
disableNotification	characterSize	*font*
disableTabStop	defaultPushButton	*foregroundColor*
disableUpdate	disabledBackgroundColor ■ ►	gotFocusEvent ■
dispatchRemainingHandlers ■	disabledForegroundColor ■ ►	*hiliteBackgroundColor*
enable ■	**empty**	*hiliteForegroundColor*
enableUpdate ■	enabled ■ ►	*inactiveColor*
handleException ■	enabledForNotification ■	inputDisabledEvent ■
hide	**firstPage**	inputEnabledEvent ■
hideSourceEmphasis	focus ►	lostFocusEvent ■
isLayoutDistorted ■	font ■ ►	***majorTabBackgroundColor***
matchForMnemonic ■	foregroundColor ■ ►	***majorTabForegroundColor***
nextPage ■	frameWindow	***minorTabBackgroundColor***
notebookSize ■	group ■	***minorTabForegroundColor***
notifyObservers ■	handle	***orientation***
pageSettings ■	helpId ■	***pageBackgroundColor***
pagesToEnd ■	hiliteBackgroundColor ■ ►	*position*
pagesToMajorTab ■	hiliteForegroundColor ■ ►	*shadowColor*
pagesToMinorTab ■	id ■	*size*
positionBehindSibling ■	inactiveColor ■ ►	systemCommandEvent
positionBehindSiblings	itemProvider ■	visibilityDisabledEvent ■
positionOnSiblings	**lastPage**	visibilityEnabledEvent ■
postEvent ■	layoutAdjustment	*visible*
previousPage ■	**majorTabBackgroundColor** ■ ►	
refresh ■	**majorTabForegroundColor** ■ ►	
refreshTabs	menu	
releasePointer	minimumSize ■	
releasePresSpace ■	**minorTabBackgroundColor** ■ ►	
removeAllPages	**minorTabForegroundColor** ■ ►	
removePage ■	nativeRect	
removeTabSection ■	**orientation** ■ ►	
resetActiveColor	owner ■	
resetBackgroundColor	**pageBackgroundColor** ■ ►	

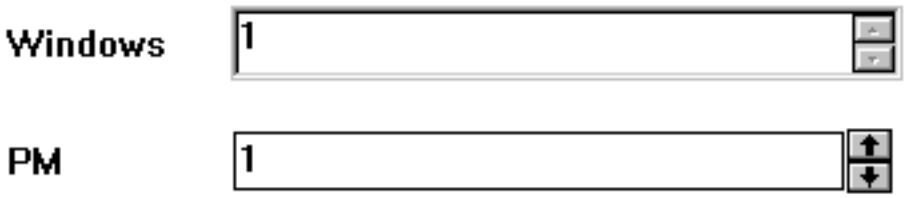
INotebook

Action	Attribute	Event
resetBorderColor	**pageSize**	
resetDisabledBackgroundColor	parent ■	
resetDisabledForegroundColor	parentSize	
resetFont	pointerCaptured	
resetForegroundColor	position ■ ►	
resetHiliteBackgroundColor	presSpace	
resetHiliteForegroundColor	rect ■	
resetInactiveColor	shadowColor ■ ►	
resetMajorTabBackgroundColor	showing	
resetMajorTabForegroundColor	size ■ ►	
resetMinimumSize	**statusTextAlignment** ■	
resetMinorTabBackgroundColor	**tabShape** ■	
resetMinorTabForegroundColor	tabStop ■	
resetPageBackgroundColor	**tabTextAlignment** ■	
resetShadowColor	**this**	
sendEvent ■	**topPage**	
setFocus	**totalPages**	
setLayoutDistorted ■	valid	
setMajorTabSize ■	visible ■ ►	
setMinorTabSize ■	visibleRectangle	
setPageButtonSize ■		
setStatusText ■		
setTabBitmap ■		
setTabText ■		
setUserData ■		
setWindow ■		
show ■		
showSourceEmphasis ■		
turnToPage ■		
window ■		

Chapter 46. INumericSpinButton

Description: IBM spin-button control with numeric contents
Part type: Visual
Part file: vbbase
Header file: ispinnum.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IBaseSpinButton
INumericSpinButton



Use an INumericSpinButton* part to provide, in sequence, a ring of related but mutually exclusive numeric choices. One choice is visible at a time. For example, use a numeric spin button for a selection list of the months of the year by number.

You can use either a spin button style that is native to the system or a style that is compatible with the OS/2 Presentation Manager (PM) spin button. The two styles are functionally equivalent.

Alternatively, use an ITextSpinButton* for a ring of text choices.

The user can spin the ring upward or downward to select another choice.

Portability — INumericSpinButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor

© Copyright IBM Corp. 1995, 1996

- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment

Related Parts — INumericSpinButton

Selection parts:

- “ICollectionViewComboBox” on page 105
- “ICollectionViewListBox” on page 113
- “IComboBox” on page 123
- “IListBox” on page 199
- “INumericSpinButton” on page 243
- “ITextSpinButton” on page 329

Button parts:

- “IAnimatedButton” on page 73
- “ICustomButton” on page 137
- “IGraphicPushButton” on page 175
- “INumericSpinButton” on page 243
- “IPushButton” on page 267
- “IRadioButton” on page 273
- “ITextSpinButton” on page 329
- “IToolBarButton” on page 349

Building Guidelines — INumericSpinButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Value Range
- Setting the Initial Value
- Aligning Values
- Preventing User Changes
- Enabling Fast Spinning
- Selecting PM Compatibility

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Value

Adding the Part

To add an INumericSpinButton* part to a composite part, select an INumericSpinButton* part from the parts palette. Then, drop the part on a canvas.

Setting the Value Range

To define the range of numbers for the spin button, set the **Numeric range** field on the INumericSpinButton* part's **General** settings page.

Setting the Initial Value

To define a default selection for the spin button, set the **Value** field on the INumericSpinButton* part's **General** settings page.

Aligning Values

To align numeric values to the right in the spin button entry field, select the **Right** choice in the **Alignment** field on the INumericSpinButton* part's **General** settings page.

Preventing User Changes

To prevent the user from typing a choice, set the **readOnly** field on the INumericSpinButton* part's **Styles** settings page.

Enabling Fast Spinning

To make the spin button skip values to cycle faster when the user continues to spin the button, set the **fastSpin** field on the INumericSpinButton* part's **Styles** settings page.

Selecting PM Compatibility

To provide a PM-compatible spin button, set the **pmCompatible** field on the INumericSpinButton* part's **Styles** settings page.

Getting the Selected Value

To copy the selection from the spin button to a numeric connection parameter or attribute, connect the INumericSpinButton* part's *value* attribute to the numeric target.

Preferred Features — INumericSpinButton

disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
enabled	If the window is sent mouse and keyboard input, `true` is returned.
limit	Returns the number of characters permitted in the spin field.
menu	Popup menu link.
range	Returns the number range of the spin button.
setFocus	Sets the input focus to the window.
spinFieldValid	If the contents of the spin field are within the number range, `true` is returned.
spinTo	Spins the spin button until the specified value displays.
this	A Pointer to the instance.
value	Returns the current value displayed in the spin field.

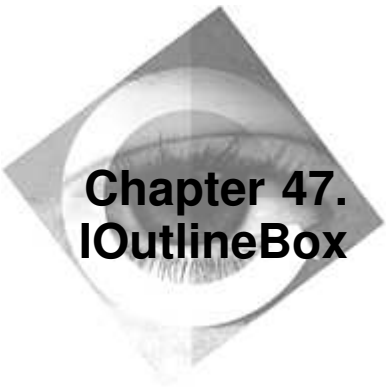
Features — INumericSpinButton

Action	Attribute	Event
addBorder ■	activeColor ■ ►	*activeColor*
applyBidiSettings ■	alignment ■	anyEvent
capturePointer ■	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
disable	autoDeleteObject ■	commandEvent
disableDataUpdate	autoDestroyWindow ■	*disabledBackgroundColor*
disableFastSpin	backgroundColor ■ ►	*disabledForegroundColor*
disableGroup	bidiSupported	*enabled*
disableNotification	border	*focus*
disableTabStop	borderColor ■ ►	*font*
disableUpdate	characterSize	*foregroundColor*
dispatchRemainingHandlers ■	defaultPushButton	gotFocusEvent ■
enable ■	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
enableUpdate ■	disabledForegroundColor ■ ►	*hiliteForegroundColor*
handleException ■	enabled ■ ►	*inactiveColor*
hide	enabledForNotification ■	inputDisabledEvent ■
hideSourceEmphasis	fastSpinEnabled ■	inputEnabledEvent ■

INumericSpinButton

Action	Attribute	Event
isLayoutDistorted ■	focus ►	lostFocusEvent ■
matchForMnemonic ■	font ■ ►	*position*
notifyObservers ■	foregroundColor ■ ►	*shadowColor*
positionBehindSibling ■	frameWindow	*size*
positionBehindSiblings	group ■	systemCommandEvent
positionOnSiblings	handle	***value***
postEvent ■	helpId ■	visibilityDisabledEvent ■
refresh ■	hiliteBackgroundColor ■ ►	visibilityEnabledEvent ■
releasePointer	hiliteForegroundColor ■ ►	*visible*
releasePresSpace ■	id ■	
removeBorder	inactiveColor ■ ►	
resetActiveColor	itemProvider ■	
resetBackgroundColor	layoutAdjustment	
resetBorderColor	limit ■	
resetDisabledBackgroundColor	master	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	**range** ■	
setLayoutDistorted ■	rect ■	
setMaster ■	servant	
show ■	shadowColor ■ ►	
showSourceEmphasis ■	showing	
spinDown ■	size ■ ►	
spinTo ■	**spinFieldValid**	
spinUp ■	tabStop ■	
	this	
	valid	
	value ■ ►	
	visible ■ ►	
	visibleRectangle	
	writeable ■	

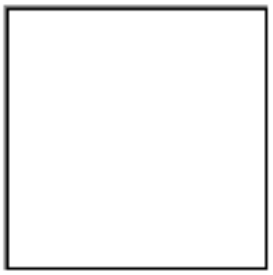
INumericSpinButton



Chapter 47. IOutlineBox

Description: IBM outline control without text label
Part type: Visual
Part file: vbbase
Header file: ioutlbox.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IOutlineBox



Use an IOutlineBox* part to outline, but not label, a group of controls. Alternatively, use an IGroupBox* part if you want to label the group. In an IMultiCellCanvas* part, you should use an ISetCanvas* part to represent an outline box.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Portability — IOutlineBox

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor

© Copyright IBM Corp. 1995, 1996

- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IOutlineBox

Label parts:

- "IGroupBox" on page 181
- "IOutlineBox" on page 249
- "IStaticText" on page 315

Building Guidelines — IOutlineBox

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Sizing the Part
- Adding Controls
- Reordering Controls
- Defining a Control Group

Adding the Part

To add an IOutlineBox* part to a composite part, select an IOutlineBox* part from the parts palette. Then, drop the part on a canvas.

Sizing the Part

To resize the IOutlineBox* part to the size you need, select the part. Then, drag a corner handle to size the part.

Adding Controls

To add the controls you want in the outline box, select control parts from the parts palette and drop them on the IOutlineBox* part.

Reordering Controls

If you add the outline box after adding the controls you are enclosing, you must change the depth order of controls on the canvas to position the outline box ahead of the controls within it.

Also, if you added the controls in the outline box out of order, reorder the controls to reflect the logical order you want.

To reorder controls on a canvas, open the canvas part's contextual menu and select **View parts list**. Then, in the parts list window, drag parts to the new positions you want.

Defining a Control Group

To make the controls in the outline box a control group that the user can reach with a tab key, do the following:

- Set the **group** and **tabStop** fields to **On** on the **Styles** settings page of the first control in the outline box.
- Leave these fields set to **Off** for the rest of the controls in the outline box.
- Ensure that the first control following the outline box on your canvas has the **group** field set to **On**. This makes the last control in the outline box the end of the control group.

Preferred Features — IOutlineBox

menu	Popup menu link.
outlineType	Returns the current type of outline for this outline box object.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

Features — IOutlineBox

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	defaultPushButton	*font*

IOutlineBox

Action	Attribute	Event
enableUpdate ■	disabledBackgroundColor ■ ►	*foregroundColor*
handleException ■	disabledForegroundColor ■ ►	gotFocusEvent ■
hide	enabled ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	focus ►	*inactiveColor*
matchForMnemonic ■	font ■ ►	inputDisabledEvent ■
notifyObservers ■	foregroundColor ■ ►	inputEnabledEvent ■
positionBehindSibling ■	frameWindow	lostFocusEvent ■
positionBehindSiblings	group ■	*position*
positionOnSiblings	handle	*shadowColor*
postEvent ■	helpId ■	*size*
refresh ■	hiliteBackgroundColor ■ ►	systemCommandEvent
releasePointer	hiliteForegroundColor ■ ►	visibilityDisabledEvent ■
releasePresSpace ■	id ■	visibilityEnabledEvent ■
resetActiveColor	inactiveColor ■ ►	*visible*
resetBackgroundColor	itemProvider ■	
resetBorderColor	layoutAdjustment	
resetDisabledBackgroundColor	menu	
resetDisabledForegroundColor	minimumSize ■	
resetFont	nativeRect	
resetForegroundColor	**outlineType** ■	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	rect ■	
setLayoutDistorted ■	shadowColor ■ ►	
show ■	showing	
showSourceEmphasis ■	size ■ ►	
	tabStop ■	
	this	
	valid	
	visible ■ ►	
	visibleRectangle	

Chapter 48. IPair

Description: IBM ordered pair of coordinates
Part type: Class
Part file: vbbase
Header file: ipoint.hpp

Derivation:
IBase
IPair

Preferred Features — IPair

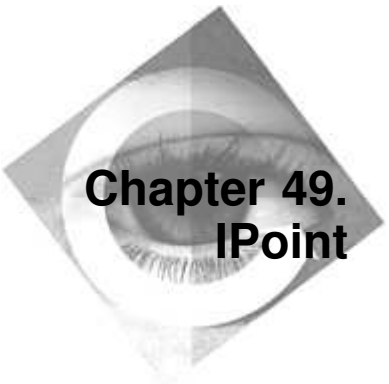
this A Pointer to the instance.

Features — IPair

Action	Attribute	Type
distanceFrom ■	asDebugInfo	`IString`
dotProduct ■	asString	`IString`
maximum ■	**coord1** ■	`IPair::Coord`
minimum ■	**coord2** ■	`IPair::Coord`
operator != ■	**this**	`IPair*`
operator %= ■		
operator *= ■		
operator += ■		
operator -		
operator -= ■		
operator /= ■		
operator < ■		
operator <= ■		
operator == ■		
operator > ■		
operator >= ■		
scaleBy ■		
scaledBy ■		
transpose		

© Copyright IBM Corp. 1995, 1996

IPair



Chapter 49. IPoint

Description: IBM 2-dimensional mathematical point
Part type: Class
Part file: vbbase
Header file: ipoint.hpp

Derivation:
IBase
IPair
IPoint

Preferred Features — IPoint

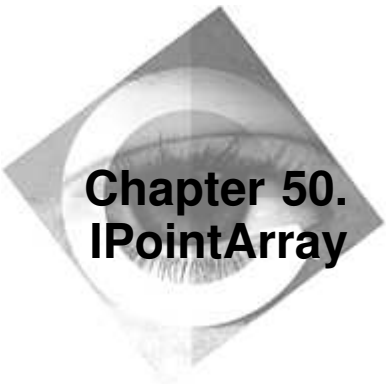
asPOINTL	Renders the point as a POINTL structure.
this	A Pointer to the instance.
x	Returns the point's X-coordinate.
y	Returns the point's Y-coordinate.

Features — IPoint

Action	Attribute	Type
distanceFrom ■	asDebugInfo	`IString`
dotProduct ■	**asPOINTL**	`struct _POINTL`
maximum ■	asString	`IString`
minimum ■	coord1 ■	`IPair::Coord`
operator != ■	coord2 ■	`IPair::Coord`
operator %= ■	**this**	`IPoint*`
operator *= ■	**x** ■	`Coord`
operator += ■	**y** ■	`Coord`
operator -		
operator -= ■		
operator /= ■		
operator < ■		
operator <= ■		
operator == ■		
operator > ■		
operator >= ■		
scaleBy ■		
scaledBy ■		
transpose		

© Copyright IBM Corp. 1995, 1996

IPoint



Chapter 50. IPointArray

Description: IBM point array
Part type: Class
Part file: vbbase
Header file: iptarray.hpp

Derivation:
IBase
IPointArray

Preferred Features — IPointArray

remove	Removes a point at the specified index.
this	A Pointer to the instance.

Features — IPointArray

Action	Attribute	Type
add ■	asDebugInfo	`IString`
insert ■	asString	`IString`
operator != ■	**reversed**	`IPointArray`
operator = ■	**size**	`unsigned long`
operator == ■	**this**	`IPointArray*`
operator [] ■		
remove ■		
resize ■		
reverse		

© Copyright IBM Corp. 1995, 1996

IPointArray

Chapter 51. IProfile

Description: IBM application-profile file
Part type: Class
Part file: vbbase
Header file: iprofile.hpp

Derivation:
IBase
IVBase
IProfile

Portability — IProfile

Features not defined for Windows:

- handle

Preferred Features — IProfile

addOrReplaceElementWithKey
Writes the specified numeric, text, or binary data.

containsKeyName
If the profile contains data for the specified application and key pair, returns `true`.

defaultApplicationName
Returns the default application name.

elementWithKey
Reads the application data and returns it as an `IString`.

integerWithKey Reads the application data and returns it as a `long` integer.

name Returns the profile's file name.

numberOfKeys Returns the number of keys for the application name.

systemProfile Returns the system profile.

this A Pointer to the instance.

Features — IProfile

Action	Attribute	Type
addOrReplaceElementWithKey ■	asDebugInfo	`IString`
containsApplication ■	asString	`IString`
containsKeyName ■	**defaultApplicationName** ■	`IString`
deleteElementWithApplication ■	**name**	`IString`

© Copyright IBM Corp. 1995, 1996

IProfile

Action	Attribute	Type
deleteElementWithKey ■	**numberOfApplications**	`unsigned long`
elementWithKey ■	**numberOfKeys**	`unsigned long`
integerWithKey ■	**systemProfile**	`IProfile::IProfile`
operator = ■	**this**	`IProfile*`
	userProfile	`IProfile::IProfile`

Chapter 52. IProgressIndicator

Description: IBM read-only slider control
Part type: Visual
Part file: vbbase
Header file: islider.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IProgressIndicator

Use an IProgressIndicator* part to show the progress of a task of significant duration.

A horizontal or vertical shaft represents the task. An *arm* moves along the shaft to show the progress of the task. The arm starts at one end of the shaft, called the *home position*, and reaches the other end when the task finishes. The shaft can be colored between the home position and the arm to emphasize the progress of the task. This colored portion of the shaft is called a *ribbon strip*.

You can use either a progress indicator style that is native to the system or a style that is compatible with the OS/2 Presentation Manager (PM) progress indicator. The two styles are functionally equivalent.

The user can note the progress of a task by observing the progress indicator.

Portability — IProgressIndicator

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor

© Copyright IBM Corp. 1995, 1996

- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IProgressIndicator

Slider parts:

- "ICircularSlider" on page 95
- "IProgressIndicator" on page 261
- "IScrollBar" on page 287
- "ISlider" on page 301

Building Guidelines — IProgressIndicator

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting PM Compatibility
- Defining a Scale
- Adding a Ribbon Strip
- Selecting the Orientation
- Setting the Home Position
- Setting the Initial Position
- Positioning to Ticks

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Position

Adding the Part

To add an IProgressIndicator* part to a composite part, select an IProgressIndicator* part from the parts palette. Then, drop the part on a canvas.

Selecting PM Compatibility

To provide a PM-compatible progress indicator, set the **pmCompatible** field on the IProgressIndicator* part's **Styles** settings page.

Defining a Scale

To define the value scale for the progress indicator, do the following:

- Enter the number of ticks in the **Number of ticks** field on the IProgressIndicator* part's **General** settings page.
- Add scale tick marks and text to the progress indicator. Do this using custom logic. Open the contextual menu of the free-form surface and connect the composite part *ready* event to IProgressIndicator* **Custom logic**. Enter C++ code to set the size and text for each tick mark.

  For example, if you want to show percentage of completion with a primary scale marked at ten-percent intervals and a secondary scale marked at five-percent intervals, you could enter the following code:

```
int i;

target->setPrimaryScale(IProgressIndicator::scale1);
target->setTicks(IProgressIndicator::scale1, 101);
for (i = 0; i < 101; i += 10)
{
  target->setTickLength(i, 10);
  target->setTickText(i, IString(i));
}

target->setPrimaryScale(IProgressIndicator::scale2);
target->setTicks(IProgressIndicator::scale2, 101);
for (i = 0; i < 101; i += 5)
{
  target->setTickLength(i, 5);
  target->setTickText(i, IString(i));
}
```

  For a horizontal progress indicator, `scale1` places the scale above the shaft; `scale2` places the scale below the shaft.

  For a vertical progress indicator, `scale1` places the scale to the right of the shaft; `scale2` places the scale to the left of the shaft.

Adding a Ribbon Strip

To add a ribbon strip to the progress indicator, set the **ribbonStrip** field to **On** on the IProgressIndicator* part's **Styles** settings page.

Selecting the Orientation

By default, the progress indicator is horizontal. To orient the progress indicator vertically, set the **vertical** field to **On** and set the **horizontal** field to **Off** on the IProgressIndicator* part's **Styles** settings page.

Setting the Home Position

To define the home position for the progress indicator, select a choice in the **Home position** field on the IProgressIndicator* part's **General** settings page.

If you want the home position at the left of a horizontal progress indicator or at the bottom of a vertical indicator, select **Left or bottom**.

If you want the home position at the right of a horizontal progress indicator or at the top of a vertical indicator, select **Top or right**.

Setting the Initial Position

To initially position the arm somewhere other than at the home position, set the **Arm offset** field on the IProgressIndicator* part's **General** settings page. Specify either a tick offset for **Ticks** or a pixel offset for **Pixels**.

Positioning to Ticks

To force the arm to the nearest tick if you use a pixel offset, set the **snapToTickMark** field to **On** on the IProgressIndicator* part's **Styles** settings page.

Changing the Position

To advance the arm to indicate progress, connect the source of your progress information to the IProgressIndicator* part's *armTickOffset* attribute, providing the new tick location of the arm.

Preferred Features — IProgressIndicator

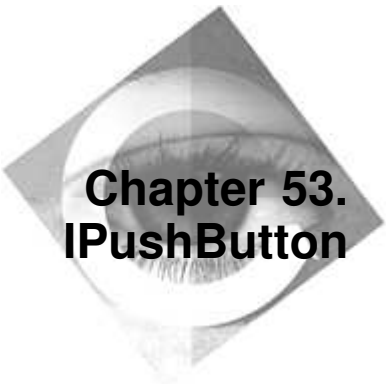
armPixelOffset	Returns the offset, in pixels, of the arm from the home position.
armTickOffset	Returns the position of the arm as a tick number.
menu	Popup menu link.
setFocus	Sets the input focus to the window.
setTickLength	Sets the length of one or all ticks on the progress indicator scale.
setTickText	Sets the text associated with the tick at the specified index.
this	A Pointer to the instance.

Features — IProgressIndicator

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**alignment**	anyEvent
convertToGUIStyle ■	**armPixelOffset** ■ ►	***armPixelOffset***
disable	**armRange**	***armTickOffset***
disableDrawItem	**armTickOffset** ■ ►	*backgroundColor*
disableGroup	asDebugInfo	*borderColor*
disableNotification	asString	commandEvent
disableRibbonStrip	autoDeleteObject ■	*disabledBackgroundColor*
disableSnapToTick	autoDestroyWindow ■	*disabledForegroundColor*
disableTabStop	backgroundColor ■ ►	*enabled*
disableUpdate	bidiSupported	*focus*
dispatchRemainingHandlers ■	borderColor ■ ►	*font*
enable ■	characterSize	*foregroundColor*
enableUpdate ■	defaultPushButton	gotFocusEvent ■
handleException ■	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
hide	disabledForegroundColor ■ ►	*hiliteForegroundColor*
hideSourceEmphasis	**drawItemEnabled** ■	*inactiveColor*
isLayoutDistorted ■	enabled ■ ►	inputDisabledEvent ■
matchForMnemonic ■	enabledForNotification ■	inputEnabledEvent ■
notifyObservers ■	focus ►	lostFocusEvent ■
numberOfTicks ■	font ■ ►	*position*
positionBehindSibling ■	foregroundColor ■ ►	***primaryScale***
positionBehindSiblings	frameWindow	**scaleEvent**
positionOnSiblings	group ■	*shadowColor*
postEvent ■	handle	*size*
refresh ■	helpId ■	systemCommandEvent
releasePointer	hiliteBackgroundColor ■ ►	visibilityDisabledEvent ■
releasePresSpace ■	hiliteForegroundColor ■ ►	visibilityEnabledEvent ■
resetActiveColor	**homePosition** ■	*visible*
resetBackgroundColor	id ■	
resetBorderColor	inactiveColor ■ ►	
resetDisabledBackgroundColor	itemProvider ■	
resetDisabledForegroundColor	layoutAdjustment	
resetFont	menu	
resetForegroundColor	minimumSize ■	
resetHiliteBackgroundColor	nativeRect	
resetHiliteForegroundColor	owner ■	
resetInactiveColor	parent ■	
resetMinimumSize	parentSize	
resetShadowColor	pointerCaptured	
sendEvent ■	position ■ ►	
setFocus	presSpace	
setLayoutDistorted ■	**primaryScale** ■ ►	
setShaftBreadth ■	rect ■	

IProgressIndicator

Action	Attribute	Event
setTickLength ■	**ribbonStripEnabled** ■	
setTicks ■	shadowColor ■ ►	
setTickText ■	**shaftPosition** ■	
show ■	**shaftSize**	
showSourceEmphasis ■	showing	
tickLength ■	size ■ ►	
tickPosition ■	**snapToTickEnabled** ■	
tickSpacing ■	tabStop ■	
tickText ■	**this**	
	valid	
	vertical	
	visible ■ ►	
	visibleRectangle	



Chapter 53. IPushButton

Description: IBM push-button control
Part type: Visual
Part file: vbbase
Header file: ipushbut.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
IPushButton

Use an IPushButton* part to let the user confirm changes or request a function. The action or routing choice is identified by text on the push button. **OK** and **Help** push buttons are commonly used examples.

Alternatively, to identify the choice with a graphic image, use an IGraphicPushButton* part.

The user can click the push button to perform the indicated function.

Portability — IPushButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- border
- removeBorder
- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IPushButton

Button parts:

- "IAnimatedButton" on page 73
- "ICustomButton" on page 137
- "IGraphicPushButton" on page 175
- "INumericSpinButton" on page 243
- "IPushButton" on page 267
- "IRadioButton" on page 273
- "ITextSpinButton" on page 329
- "IToolBarButton" on page 349

Building Guidelines — IPushButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Defining the Default Button
- Defining a Help Button
- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting the Requested Action
- Disabling the Part
- Enabling the Part

Adding the Part

To add an IPushButton* part to a composite part, select an IPushButton* part from the parts palette. Then, drop the part on a canvas.

Defining Text for the Part

To assign text to the push button, set the **Text** field on the IPushButton* part's **General** settings page.

Defining the Default Button

The default button is behaves as if it were clicked when the user presses the Enter key and no button has the input focus. To define the push button as the default button, set the **defaultButton** field on the IPushButton* part's **Styles** settings page.

Defining a Help Button

To define the push button as a help button, select the **Help** and **noPointerFocus** fields on the IPushButton* part's **Styles** settings page.

Enabling Tab Key Access

To allow the user to reach the button by using a tab key, set the **tabStop** field to **On** on the IPushButton* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the IPushButton* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Starting the Requested Action

To start the action represented by the button, connect the IPushButton* part's *buttonClickEvent* feature to the action on the target part.

Disabling the Part

To disable the push button when an event occurs, connect the event to the IPushButton* part's *disable* action.

Enabling the Part

To enable the push button when an event occurs, do the following:

1. Connect the event to the IPushButton* part's *enable* action.

2. Open settings for the connection, select **Set parameters**, and check the **enabled** setting.

Preferred Features — IPushButton

buttonClickEvent
Notification identifier provided to observers when the button control is clicked by the user.

default
If the style defaultButton is set, returns `true`.

disable
Prevents keyboard and mouse input from being sent to the window.

disableDefault
Removes the style defaultButton from the push button.

enable
Enables the window to accept keyboard and mouse input.

enabled
If the window is sent mouse and keyboard input, `true` is returned.

menu
Popup menu link.

setFocus
Sets the input focus to the window.

text
Returns the control window's text.

this
A Pointer to the instance.

Features — IPushButton

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	allowsMouseClickFocus ■	anyEvent
click	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
disable	autoDeleteObject ■	buttonClickEvent
disableDefault	autoDestroyWindow ■	commandEvent
disableGroup	backgroundColor ■ ►	*disabledBackgroundColor*
disableHelp	bidiSupported	*disabledForegroundColor*
disableMouseClickFocus	**border** ■	*enabled*
disableNotification	borderColor ■ ►	*focus*
disableSystemCommand	characterSize	*font*
disableTabStop	clipboardHasTextFormat	*foregroundColor*
disableUpdate	**default** ■	gotFocusEvent ■
dispatchRemainingHandlers ■	defaultPushButton	*hiliteBackgroundColor*
enable ■	disabledBackgroundColor ■ ►	*hiliteForegroundColor*
enableUpdate ■	disabledForegroundColor ■ ►	*inactiveColor*
handleException ■	displaySize	inputDisabledEvent ■
hide	enabled ■ ►	inputEnabledEvent ■
hideSourceEmphasis	enabledForNotification ■	*isDigits*
highlight ■	focus ►	lostFocusEvent ■
isLayoutDistorted ■	font ■ ►	*position*
matchForMnemonic ■	foregroundColor ■ ►	*shadowColor*
notifyObservers ■	frameWindow	*size*
positionBehindSibling ■	group ■	***systemCommand***

IPushButton

Action	Attribute	Event
positionBehindSiblings	handle	systemCommandEvent
positionOnSiblings	**help** ■	*text*
postEvent ■	helpId ■	*textLength*
refresh ■	highlighted	*valueAsDouble*
releasePointer	hiliteBackgroundColor ■ ►	*valueAsInt*
releasePresSpace ■	hiliteForegroundColor ■ ►	*valueAsUnsigned*
removeBorder	id ■	visibilityDisabledEvent ■
resetActiveColor	inactiveColor ■ ►	visibilityEnabledEvent ■
resetBackgroundColor	isDigits ►	*visible*
resetBorderColor	itemProvider ■	
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	rect ■	
setLayoutDistorted ■	shadowColor ■ ►	
show ■	showing	
showSourceEmphasis ■	size ■ ►	
unhighlight	**systemCommand** ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

IPushButton

Chapter 54. IRadioButton

Description: IBM radio-button control
Part type: Visual
Part file: vbbase
Header file: iradiobt.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
ISettingButton
IRadioButton

Use an IRadioButton* part to provide a choice for a set of mutually exclusive choices, where one choice is always selected. Selection of the choice is indicated by a mark in the radio button.

Use radio buttons in a group for a small, fixed set of choices. For example, use a group of radio buttons for a choice of sizes, such as small, medium, and large. You can also use a group of radio buttons for a set of answers to a multiple-choice question.

Alternatively, use a group of ICheckBox parts for a set of choices that are not mutually exclusive. For mutually exclusive choices, you can also use a single-select list box.

The user can select a single choice from a group of radio buttons. When the user selects a choice, the previously selected choice is no longer selected.

Portability — IRadioButton

Features not defined for Windows:

- messageQueue

© Copyright IBM Corp. 1995, 1996

Features not active on Windows:

- cursorSelect
- disableCursorSelect
- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IRadioButton

Setting choice parts:

- “I3StateCheckBox” on page 61
- “ICheckBox” on page 89
- “IRadioButton” on page 273

Button parts:

- “IAnimatedButton” on page 73
- “ICustomButton” on page 137
- “IGraphicPushButton” on page 175
- “INumericSpinButton” on page 243
- “IPushButton” on page 267
- “IRadioButton” on page 273
- “ITextSpinButton” on page 329
- “IToolBarButton” on page 349

Building Guidelines — IRadioButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Text for the Part
- Setting the Initial State
- Enabling Tab Key Access
- Defining a Control Group

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Disabling Another Part
- Enabling Another Part
- Disabling the Part
- Enabling the Part

Adding the Part

To add an IRadioButton* part to a composite part, select an IRadioButton* part from the parts palette. Then, drop the part on a canvas.

Defining Text for the Part

To set the text for the radio button, enter the text you want in the **Text** field on the IRadioButton* part's **General** settings page.

Setting the Initial State

To set the initial state of the radio button, select or deselect the **Selected** field on the IRadioButton* part's **General** settings page.

To initially select your radio button, make sure the **Selected** field check box is checked. If it is not checked, select it.

To initially not select your radio button, explicitly deselect the **Selected** field check box. If it is checked, select it to uncheck it. If it is not checked, select it; then select on it again to uncheck it.

Ensure that only one radio button in a control group is initially selected.

Enabling Tab Key Access

To allow the user to reach the radio button by using a tab key, set the **tabStop** field to **On** on the IRadioButton* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the IRadioButton* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

Placing radio buttons in a group also ensures that when the user selects one radio button, the other buttons are deselected.

Disabling Another Part

To disable another control when the user selects the radio button, connect the IRadioButton* part's *selected* attribute to the *value* attribute of an IVBBooleanPart* part. Then, connect the IVBBooleanPart* part's *notValue* attribute to the *enabled* attribute of the other control. When the user deselects the radio button, the other control is enabled.

IVBBooleanPart is available in the vbsample.vbb part file.

Enabling Another Part

To enable another control when the user selects the radio button, connect the IRadioButton* part's *selected* attribute to the *enabled* attribute of the other control. When the user deselects the radio button, the other control is disabled.

Disabling the Part

To disable the radio button when an event occurs, connect the event to the IRadioButton* part's *disable* action.

Enabling the Part

To enable the radio button when an event occurs, do the following:

1. Connect the event to the IRadioButton* part's *enable* action.
2. The button is enabled by default. If you have changed this default, open settings for the connection, select **Set parameters**, and check the **Enabled** setting.

Preferred Features — IRadioButton

buttonClickEvent
: Notification identifier provided to observers when the button control is clicked by the user.

disable
: Prevents keyboard and mouse input from being sent to the window.

enable
: Enables the window to accept keyboard and mouse input.

enabled
: If the window is sent mouse and keyboard input, `true` is returned.

menu
: Popup menu link.

selected
: If the button is selected, `true` is returned.

selectedIndex
: Returns the 0-based index of the selected radio button in a group.

setFocus
: Sets the input focus to the window.

text
: Returns the control window's text.

this
: A Pointer to the instance.

Features — IRadioButton

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	allowsMouseClickFocus ■	anyEvent
click	asDebugInfo	*backgroundColor*
convertToGUIStyle ■	asString	*borderColor*
deselect	autoDeleteObject ■	buttonClickEvent
disable	autoDestroyWindow ■	commandEvent
disableAutoSelect	**autoSelect** ■	*disabledBackgroundColor*
disableCursorSelect	backgroundColor ■ ►	*disabledForegroundColor*
disableGroup	bidiSupported	*enabled*
disableMouseClickFocus	borderColor ■ ►	*focus*
disableNotification	characterSize	*font*
disableTabStop	clipboardHasTextFormat	*foregroundColor*
disableUpdate	**cursorSelect** ■	gotFocusEvent ■
dispatchRemainingHandlers ■	defaultPushButton	*hiliteBackgroundColor*
enable ■	disabledBackgroundColor ■ ►	*hiliteForegroundColor*
enableUpdate ■	disabledForegroundColor ■ ►	*inactiveColor*
handleException ■	displaySize	inputDisabledEvent ■
hide	enabled ■ ►	inputEnabledEvent ■
hideSourceEmphasis	enabledForNotification ■	*isDigits*
highlight ■	focus ►	lostFocusEvent ■
isLayoutDistorted ■	font ■ ►	*position*
matchForMnemonic ■	foregroundColor ■ ►	*selected*
notifyObservers ■	frameWindow	*shadowColor*
positionBehindSibling ■	group ■	*size*
positionBehindSiblings	handle	systemCommandEvent
positionOnSiblings	helpId ■	*text*
postEvent ■	highlighted	*textLength*
refresh ■	hiliteBackgroundColor ■ ►	*valueAsDouble*

IRadioButton

Action	Attribute	Event
releasePointer	hiliteForegroundColor ■ ►	*valueAsInt*
releasePresSpace ■	id ■	*valueAsUnsigned*
resetActiveColor	inactiveColor ■ ►	visibilityDisabledEvent ■
resetBackgroundColor	isDigits ►	visibilityEnabledEvent ■
resetBorderColor	itemProvider ■	*visible*
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	rect ■	
setLayoutDistorted ■	selected ■ ►	
show ■	**selectedIndex**	
showSourceEmphasis ■	shadowColor ■ ►	
unhighlight	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

Chapter 55. IRange

Description: IBM range of coordinate values
Part type: Class
Part file: vbbase
Header file: ipoint.hpp

Derivation:
IBase
IPair
IRange

Preferred Features — IRange

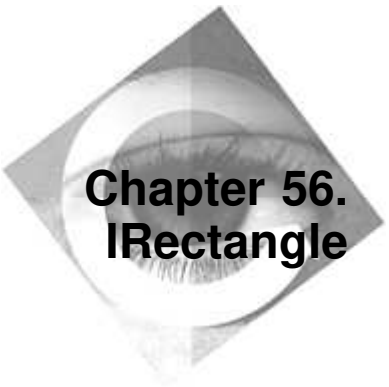
lowerBound	Returns the lower bound of the range.
this	A Pointer to the instance.
upperBound	Returns the upper bound of the range.

Features — IRange

Action	Attribute	Type
distanceFrom ■	asDebugInfo	`IString`
dotProduct ■	asString	`IString`
includes ■	coord1 ■	`IPair::Coord`
maximum ■	coord2 ■	`IPair::Coord`
minimum ■	**lowerBound** ■	`Coord`
operator != ■	**this**	`IRange*`
operator %= ■	**upperBound** ■	`Coord`
operator *= ■		
operator += ■		
operator -		
operator -= ■		
operator /= ■		
operator < ■		
operator <= ■		
operator == ■		
operator > ■		
operator >= ■		
scaleBy ■		
scaledBy ■		
transpose		

© Copyright IBM Corp. 1995, 1996

IRange



Chapter 56. IRectangle

Description: IBM rectangle, defined by 2 points
Part type: Class
Part file: vbbase
Header file: irect.hpp

Derivation:
IBase
IRectangle

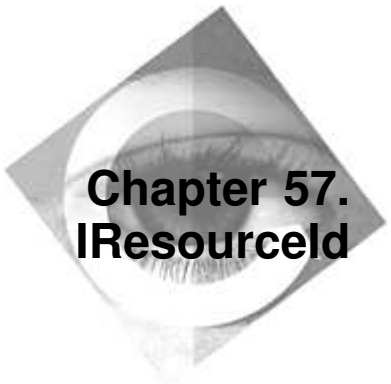
Preferred Features — IRectangle

asRECTL	Converts the rectangle into a system dependent structure.
bottomCenter	Returns the X- and Y-coordinates of the bottom-center point of the rectangle.
bottomLeft	Returns the X- and Y-coordinates of the bottom-left corner of the rectangle.
bottomRight	Returns the X- and Y-coordinates of the bottom-right corner of the rectangle.
center	Returns the X- and Y-coordinates of the center point of the rectangle.
leftCenter	Returns the X- and Y-coordinates of the left-center point of the rectangle.
rightCenter	Returns the X- and Y-coordinates of the right-center point of the rectangle.
size	Returns the `ISize`(width, height).
this	A Pointer to the instance.
topCenter	Returns the X- and Y-coordinates of the top-center point of the rectangle.
topLeft	Returns the X- and Y-coordinates of the top-left corner of the rectangle.
topRight	Returns the X- and Y-coordinates of the top-right corner of the rectangle.

© Copyright IBM Corp. 1995, 1996

Features — IRectangle

Action	Attribute	Type
centerAt ■	**area**	IRectangle::Coord
centeredAt ■	asDebugInfo	IString
contains ■	**asRECTL**	struct _RECTL
expandBy ■	asString	IString
expandedBy ■	**bottom**	IRectangle::Coord
intersects ■	**bottomCenter**	IPoint
moveBy ■	**bottomLeft** ■	IPoint
movedBy ■	**bottomRight**	IPoint
movedTo ■	**center**	IPoint
operator != ■	**centerXCenterY**	IPoint
operator & ■	**centerXMaxY**	IPoint
operator &= ■	**centerXMinY**	IPoint
operator == ■	**height**	IRectangle::Coord
operator | ■	**left**	IRectangle::Coord
operator |= ■	**leftCenter**	IPoint
scaleBy ■	**maxX**	IRectangle::Coord
scaledBy ■	**maxXCenterY**	IPoint
shrinkBy ■	**maxXMaxY**	IPoint
shrunkBy ■	**maxXMinY**	IPoint
sizeBy ■	**maxY**	IRectangle::Coord
sizedBy ■	**minX**	IRectangle::Coord
sizedTo ■	**minXCenterY**	IPoint
	minXMaxY	IPoint
	minXMinY	IPoint
	minY	IRectangle::Coord
	right	IRectangle::Coord
	rightCenter	IPoint
	size ■	ISize
	this	IRectangle*
	top	IRectangle::Coord
	topCenter	IPoint
	topLeft	IPoint
	topRight	IPoint
	width	IRectangle::Coord



Chapter 57. IResourceId

Description: IBM resource identifier
Part type: Class
Part file: vbbase
Header file: ireslib.hpp

Derivation:
IBase
IResourceId

Preferred Features — IResourceId

id Returns the identifier used for the resource.

resourceLibrary
Returns a const reference to the resource library.

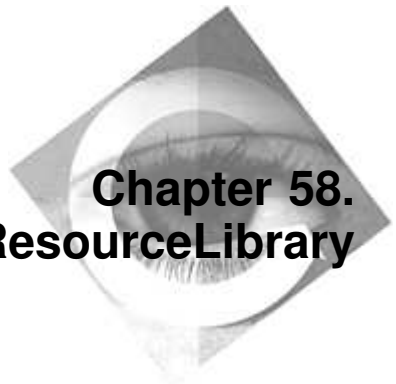
this A Pointer to the instance.

Features — IResourceId

Action	Attribute	Type
operator unsigned long	asDebugInfo	`IString`
operator= ∎	asString	`IString`
	id	`unsigned long`
	resourceLibrary	`IResourceLibrary`
	this	`IResourceId*`

© Copyright IBM Corp. 1995, 1996

IResourceId



Chapter 58. IResourceLibrary

Description: IBM resource file
Part type: Class
Part file: vbbase
Header file: ireslib.hpp

Derivation:
IBase
IVBase
IResourceLibrary

Preferred Features — IResourceLibrary

fileName	Returns the fully qualified file name of the resource library.
open	Returns the open state of the resource file.
this	A Pointer to the instance.

Features — IResourceLibrary

Action	Attribute	Type
loadAccelTable ■	asDebugInfo	`IString`
loadBitmap ■	asString	`IString`
loadDialog ■	**fileName**	`IString`
loadHelpTable ■	**handle**	`IModuleHandle`
loadIcon ■	**open**	`Boolean`
loadMenu ■	**this**	`IResourceLibrary*`
loadMessage ■		
loadPointer ■		
loadString ■		
operator = ■		
tryToLoadBitmap ■		
tryToLoadIcon ■		
tryToLoadMessage ■		
tryToLoadString ■		

© Copyright IBM Corp. 1995, 1996

IResourceLibrary

Chapter 59. IScrollBar

Description: IBM scroll-bar control
Part type: Visual
Part file: vbbase
Header file: iscroll.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IScrollBar

Use an IScrollBar* part for a scrollable view or slider that requires custom logic. Generally, we strongly recommend that you use an IViewPort* part for a view with scroll bars. Use an ISlider* part for typical slider control.

A scroll bar consists of a shaft, a scroll box in the shaft, and scroll buttons at the ends of the shaft. A scroll bar is either horizontal or vertical. With a scrollable view, use the scroll bar orientation to indicate the scrolling dimension.

The user can move the scroll box along the shaft in any of the following ways:

- Drag it with the mouse.
- Click or hold down a mouse button in the shaft away from the scroll box to pull it in that direction. This moves the scroll box and view contents by increments of the view width or height.
- Click or hold down a scroll button to pull the scroll box in the direction of the scroll button. This moves the scroll box and view contents by increments of the line height or character width.

Portability — IScrollBar

© Copyright IBM Corp. 1995, 1996

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IScrollBar

Frame parts:

- “IFrameWindow” on page 167
- “IMenu” on page 205
- “IScrollBar” on page 287
- “ITitle” on page 337
- “IToolBar” on page 343
- “IVBInfoArea” on page 385

Slider parts:

- “ICircularSlider” on page 95
- “IProgressIndicator” on page 261
- “IScrollBar” on page 287
- “ISlider” on page 301

Building Guidelines — IScrollBar

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Attaching a Scroll Bar to a Frame Window
- Selecting the Orientation
- Adding a Handler for the Part

Adding the Part

To add an IScrollBar* part to a composite part, select an IScrollBar* part from the parts palette. Then, drop the part on the free-form surface or on a canvas. To use the scroll bar for the client area of a frame window, drop the part on the free-form surface. To use the scroll bar as a slider, drop the part on a canvas.

Attaching a Scroll Bar to a Frame Window

To attach a scroll bar for the client area of a frame window, do the following:

1. Connect the composite part's *ready* event (from the free-form surface) to the IFrameWindow* part's *addExtension* action.
2. Connect the IScrollBar* part's *this* attribute to the *newExtension* parameter of the *ready*-to-*addExtension* connection.
3. Opening settings for the *this*-to-*newExtension* connection. Then, press the **Set parameters** push button and select a choice from the **location** field. This defines the location of the scroll bar relative to the client area.

You also need to define the scroll bar as a child of the frame window. To accomplish this, do the following:

- Connect the IFrameWindow* part's *this* attribute to the IScrollBar* part's *owner* attribute. This assigns the frame window as the owner of the scroll bar.
- Connect the IFrameWindow* part's *this* attribute to the IScrollBar* part's *parent* attribute. This assigns the frame window as the parent of the scroll bar.

Selecting the Orientation

The orientation of a client area scroll bar is determined by its location relative to the client.

To set the orientation of a slider scroll bar, set the **vertical** field and the **horizontal** field on the IScrollBar* part's **Styles** settings page.

For a vertical slider, set the **vertical** field to **On** and the **horizontal** field to **Off**.

For a horizontal slider, set the **vertical** field to **Off** and the **horizontal** field to **On**.

Adding a Handler for the Part

You need to derive a handler from the IScrollHandler class to scroll the client area or process slider values. To add the handler for the scroll bar, do the following:

- Add the handler name and constructor parameters in the **Handler list** field on the IScrollBar* part's **Handlers** settings page.
- If the handler is a part that is loaded into Visual Builder, you do not need to specify the handler's header file name. Otherwise, type the header file name for the handler in the Class Editor **Required include files** field.

Preferred Features — IScrollBar

disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
enabled	If the window is sent mouse and keyboard input, `true` is returned.
menu	Popup menu link.
scrollBoxPosition	Returns the current position of the scroll box.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

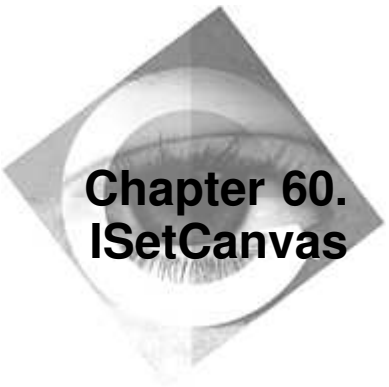
Features — IScrollBar

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	defaultPushButton	*font*
enableUpdate ■	disabledBackgroundColor ■ ►	*foregroundColor*
handleException ■	disabledForegroundColor ■ ►	gotFocusEvent ■
hide	enabled ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	focus ►	*inactiveColor*
matchForMnemonic ■	font ■ ►	inputDisabledEvent ■
notifyObservers ■	foregroundColor ■ ►	inputEnabledEvent ■
positionBehindSibling ■	frameWindow	lostFocusEvent ■
positionBehindSiblings	group ■	*position*

IScrollBar

Action	Attribute	Event
positionOnSiblings	handle	***scrollBoxPosition***
postEvent ■	helpId ■	*shadowColor*
refresh ■	hiliteBackgroundColor ■ ►	*size*
releasePointer	hiliteForegroundColor ■ ►	systemCommandEvent
releasePresSpace ■	**horizontal**	visibilityDisabledEvent ■
resetActiveColor	id ■	visibilityEnabledEvent ■
resetBackgroundColor	inactiveColor ■ ►	*visible*
resetBorderColor	itemProvider ■	
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	**minScrollIncrement** ■	
resetHiliteBackgroundColor	nativeRect	
resetHiliteForegroundColor	owner ■	
resetInactiveColor	**pageScrollIncrement** ■	
resetMinimumSize	parent ■	
resetShadowColor	parentSize	
sendEvent ■	pointerCaptured	
setFocus	position ■ ►	
setLayoutDistorted ■	presSpace	
setScrollBar ■	rect ■	
show ■	**scrollableRange** ■	
showSourceEmphasis ■	**scrollBoxPosition** ■ ►	
	scrollBoxRange	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	systemScrollBarWidth	
	systemScrollBoxLength	
	systemScrollButtonLength	
	tabStop ■	
	this	
	valid	
	vertical	
	visible ■ ►	
	visibleCount ■	
	visibleRectangle	

IScrollBar



Chapter 60. ISetCanvas

Description: IBM canvas control with child windows set in decks
Part type: Visual
Part file: vbbase
Header file: isetcv.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
ISetCanvas

Use an ISetCanvas* part to place controls in rows or columns, called *decks*. For example, you can easily arrange a field of push buttons on a set canvas.

You can use an ISetCanvas* part in another composer part, such as the following:

- IFrameWindow* client area
- IViewPort*
- IMultiCellCanvas*
- ISplitCanvas*
- IVBNotebookPage*

Then, you can add controls to the canvas within the other composer part.

You can also use a set canvas as the base part for a composite part. Then, you can add the canvas with its controls as a subpart in other composite parts.

Controls on a set canvas are adjusted for varying control text length. This provides built-in flexibility for language translation.

Alternatively, use one of the following canvas parts:

- ICanvas* for a simple canvas
- IDrawingCanvas* for graphic objects
- IMultiCellCanvas* to provide freely arranged canvas cells
- ISplitCanvas* for split windows

© Copyright IBM Corp. 1995, 1996

The user interacts with controls placed in the cells, not with the canvas itself.

Portability — ISetCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ISetCanvas

Canvas parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

Client parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “IMultiLineEdit” on page 231
- “INotebook” on page 237
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — ISetCanvas

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting the Orientation
- Adding Controls
- Aligning Controls
- Sizing Controls
- Reordering Controls
- Defining as a Group Box

Adding the Part

To add an ISetCanvas* part to a composite part, select an ISetCanvas* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- An IViewPort* part
- An IVBNotebookPage* part
- A canvas part
- The free-form surface

Selecting the Orientation

To orient controls in either rows or columns, select a choice in the **Deck orientation** field on the ISetCanvas* part's **General** settings page. By default, the controls are organized in a single row. To change the control orientation to columns, select the **vertical** choice.

To change the number of *decks* (rows or columns), set the **Deck count** field on the ISetCanvas* part's **General** settings page.

Adding Controls

To add controls to the canvas, select the control parts from the parts palette and drop them on the ISetCanvas* part. Each column or row is made large enough to accommodate the largest minimum size of the controls it contains.

Aligning Controls

To align controls across decks to give the appearance of both rows and columns, select the **Even** setting in the **Pack type** field on the ISetCanvas* part's **General** settings page.

To align the controls within a deck, select the alignment you want in the **Alignment** field on the ISetCanvas* part's **General** settings page.

Sizing Controls

To make all controls in a deck the same size, select the **Expanded** setting in the **Pack type** field on the ISetCanvas* part's **General** settings page. The controls are sized to the largest minimum size of controls in the deck.

Reordering Controls

To reorder controls on the canvas, open the ISetCanvas* part's contextual menu and select **View parts list**. Then, in the parts list window, drag parts within the list to reorder them.

Defining as a Group Box

Use a set canvas to achieve the appearance of a group box around controls on a multicell canvas. The set canvas can provide the appearance you want without sacrificing automatic sizing of controls. To define a set canvas as a group box on a multicell canvas, do the following:

- Drop a ISetCanvas* on a cell of the IMultiCellCanvas* part.
- Enter the group box text you want in the **Text** field on the ISetCanvas* part's **General** settings page.
- For vertical alignment of controls, select the **vertical** choice in the **Alignment** field on the ISetCanvas* part's **General** settings page.
- Drop the controls you want on the ISetCanvas* part.

Preferred Features — ISetCanvas

alignment	Returns the enumerator for deck alignment.
deckCount	Returns the maximum number of decks used by the canvas.
deckOrientation	Returns an enumerator for deck direction.
menu	Popup menu link.
packType	Returns an enumerator for child window spacing in decks.
pad	Returns the pad width and height, which is the space between child windows in a deck and between multiple decks.
setFocus	Sets the input focus to the window.
text	Returns the text of the group box.

this A Pointer to the instance.

Features — ISetCanvas

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**alignment** ■	anyEvent
convertToGUIStyle ■	asDebugInfo	*backgroundColor*
disable	asString	*borderColor*
disableGroup	autoDeleteObject ■	commandEvent
disableNotification	autoDestroyWindow ■	***deckOrientation***
disableTabStop	backgroundColor ■ ►	*disabledBackgroundColor*
disableUpdate	bidiSupported	*disabledForegroundColor*
dispatchRemainingHandlers ■	borderColor ■ ►	*enabled*
enable ■	characterSize	*focus*
enableUpdate ■	**deckCount** ■	*font*
handleException ■	**deckOrientation** ■ ►	*foregroundColor*
hide	defaultPushButton	gotFocusEvent ■
hideSourceEmphasis	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
isLayoutDistorted ■	disabledForegroundColor ■ ►	*hiliteForegroundColor*
matchForMnemonic ■	enabled ■ ►	*inactiveColor*
notifyObservers ■	enabledForNotification ■	inputDisabledEvent ■
positionBehindSibling ■	focus ►	inputEnabledEvent ■
positionBehindSiblings	font ■ ►	lostFocusEvent ■
positionOnSiblings	foregroundColor ■ ►	*position*
postEvent ■	frameWindow	*shadowColor*
refresh ■	group ■	*size*
releasePointer	**groupPad** ■	systemCommandEvent
releasePresSpace ■	handle	***text***
resetActiveColor	helpId ■	visibilityDisabledEvent ■
resetBackgroundColor	hiliteBackgroundColor ■ ►	visibilityEnabledEvent ■
resetBorderColor	hiliteForegroundColor ■ ►	*visible*
resetDisabledBackgroundColor	id ■	
resetDisabledForegroundColor	inactiveColor ■ ►	
resetFont	itemProvider ■	
resetForegroundColor	layoutAdjustment	
resetHiliteBackgroundColor	**margin** ■	
resetHiliteForegroundColor	menu	
resetInactiveColor	minimumSize ■	
resetMinimumSize	nativeRect	
resetShadowColor	origDefaultButtonHandle	
sendEvent ■	owner ■	
setFocus	**packType** ■	
setLayoutDistorted ■	**pad** ■	
show ■	parent ■	
showSourceEmphasis ■	parentSize	
	pointerCaptured	

ISetCanvas

Action	Attribute	Event
	position ■ ►	
	presSpace	
	rect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	this	
	valid	
	visible ■ ►	
	visibleRectangle	

Chapter 61. ISize

Description: IBM support for size in height and width
Part type: Class
Part file: vbbase
Header file: ipoint.hpp

Derivation:
IBase
IPair
ISize

Preferred Features — ISize

asSIZEL	Returns the `ISize` as a SIZEL structure.
height	Returns the height represented by the `ISize` object.
this	A Pointer to the instance.
width	Returns the width represented by the `ISize` object.

Features — ISize

Action	Attribute	Type
distanceFrom ■	asDebugInfo	`IString`
dotProduct ■	**asSIZEL**	`SIZEL`
maximum ■	asString	`IString`
minimum ■	coord1 ■	`IPair::Coord`
operator != ■	coord2 ■	`IPair::Coord`
operator %= ■	**height** ■	`Coord`
operator *= ■	**this**	`ISize*`
operator += ■	**width** ■	`Coord`
operator -		
operator -= ■		
operator /= ■		
operator < ■		
operator <= ■		
operator == ■		
operator > ■		
operator >= ■		
scaleBy ■		
scaledBy ■		
transpose		

© Copyright IBM Corp. 1995, 1996

ISize

Chapter 62. ISlider

Description: IBM linear slider control
Part type: Visual
Part file: vbbase
Header file: islider.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IProgressIndicator
ISlider

Windows

PM

Use an ISlider* part to show a range of selection values, such as the following:

- Shades of color
- Radio station frequencies
- Temperature on a thermostat

A horizontal or vertical shaft represents the range of values. A slider *arm* marks the selected value.

You can use either a slider style that is native to the system or a style that is compatible with the OS/2 Presentation Manager (PM) slider. The two styles are functionally equivalent.

The user moves the arm along the shaft to change the selected value. The user can move the arm in any of the following ways:

- Drag it with the mouse.
- Click or press a *slider button* to move the scroll box in the direction indicated on the button. This moves the arm by tick increments defined for the slider.

Portability — ISlider

© Copyright IBM Corp. 1995, 1996

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ISlider

Slider parts:

- "ICircularSlider" on page 95
- "IProgressIndicator" on page 261
- "IScrollBar" on page 287
- "ISlider" on page 301

Building Guidelines — ISlider

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting PM Compatibility
- Defining a Scale
- Selecting the Orientation
- Selecting the Button Location
- Setting the Home Position
- Setting the Initial Position
- Positioning to Ticks

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Position

Adding the Part

To add an ISlider* part to a composite part, select an ISlider* part from the parts palette. Then, drop the part on a canvas.

Selecting PM Compatibility

To provide a PM-compatible slider, set the **pmCompatible** field on the ISlider* part's **Styles** settings page.

Defining a Scale

To define the value scale for the slider, do the following:

- Enter the number of ticks in the **Number of ticks** field on the ISlider* part's **General** settings page.
- Add scale tick marks and text to the slider. Do this using custom logic. Open the contextual menu of the free-form surface and connect the composite part *ready* event to ISlider* **Custom logic**. Enter C++ code to set the size and text for each tick mark.

  For example, if you want the user to set a rating from 0 to 100 with a primary scale marked at intervals of 10 and a secondary scale marked at intervals of 5, you could enter the following code:

```
int i;

target->setPrimaryScale(IProgressIndicator::scale1);
target->setTicks(IProgressIndicator::scale1, 101);
for (i = 0; i < 101; i += 10)
{
  target->setTickLength(i, 10);
  target->setTickText(i, IString(i));
}

target->setPrimaryScale(IProgressIndicator::scale2);
target->setTicks(IProgressIndicator::scale2, 101);
for (i = 0; i < 101; i += 5)
{
  target->setTickLength(i, 5);
  target->setTickText(i, IString(i));
}
```

  For a horizontal slider, `scale1` places the scale above the shaft; `scale2` places the scale below the shaft.

For a vertical slider, `scale1` places the scale to the right of the shaft; `scale2` places the scale to the left of the shaft.

Selecting the Orientation

By default, the orientation is horizontal. To orient the slider vertically, set the **vertical** field to **On** and set the **horizontal** field to **Off** on the ISlider* part's **Styles** settings page.

Selecting the Button Location

To select the location of slider buttons relative to the slider shaft, select a button location choice on the ISlider* part's **Styles** settings page.

If you want slider buttons, set one of the location choices to **On** and set the others to **Off**.

If you do not want slider buttons, set all of the location choices to **Off**.

Setting the Home Position

To define the home position for the slider, select a choice in the **Home position** field on the ISlider* part's **General** settings page.

If you want the home position at the left of a horizontal slider or at the bottom of a vertical slider, select **Left or bottom**.

If you want the home position at the right of a horizontal slider or at the top of a vertical slider, select **Top or right**.

Setting the Initial Position

To initially position the arm somewhere other than at the home position, set the **Arm offset** field on the ISlider* part's **General** settings page. Specify either a tick offset for **Ticks** or a pixel offset for **Pixels**.

Positioning to Ticks

To force the arm to the nearest tick if you use a pixel offset, set the **snapToTickMark** field to **On** on the ISlider* part's **Styles** settings page.

Changing the Position

To provide the selected value to an attribute in another part or to an action parameter, connect the ISlider* part's *armTickOffset* attribute to the target attribute or parameter.

Preferred Features — ISlider

armPixelOffset	Returns the offset, in pixels, of the arm from the home position.
armTickOffset	Returns the position of the arm as a tick number.
disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
enabled	If the window is sent mouse and keyboard input, `true` is returned.
menu	Popup menu link.
setFocus	Sets the input focus to the window.
setTickLength	Sets the length of one or all ticks on the progress indicator scale.
setTickText	Sets the text associated with the tick at the specified index.
this	A Pointer to the instance.

Features — ISlider

Action	Attribute	Event
addDetent ■	activeColor ■ ►	*activeColor*
applyBidiSettings ■	alignment	anyEvent
capturePointer ■	armPixelOffset ■ ►	*armPixelOffset*
convertToGUIStyle ■	armRange	*armTickOffset*
detentPosition ■	**armSize** ■	**armTrackEvent**
disable	armTickOffset ■ ►	*backgroundColor*
disableDrawItem	asDebugInfo	*borderColor*
disableGroup	asString	commandEvent
disableNotification	autoDeleteObject ■	*disabledBackgroundColor*
disableRibbonStrip	autoDestroyWindow ■	*disabledForegroundColor*
disableSnapToTick	backgroundColor ■ ►	*enabled*
disableTabStop	bidiSupported	*focus*
disableUpdate	borderColor ■ ►	*font*
dispatchRemainingHandlers ■	**buttonsPosition**	*foregroundColor*
enable ■	characterSize	gotFocusEvent ■
enableUpdate ■	defaultPushButton	*hiliteBackgroundColor*
handleException ■	disabledBackgroundColor ■ ►	*hiliteForegroundColor*
hide	disabledForegroundColor ■ ►	*inactiveColor*
hideSourceEmphasis	drawItemEnabled ■	inputDisabledEvent ■
isLayoutDistorted ■	enabled ■ ►	inputEnabledEvent ■
matchForMnemonic ■	enabledForNotification ■	lostFocusEvent ■
notifyObservers ■	focus ►	*position*
numberOfTicks ■	font ■ ►	*primaryScale*
positionBehindSibling ■	foregroundColor ■ ►	scaleEvent
positionBehindSiblings	frameWindow	*shadowColor*
positionOnSiblings	group ■	*size*
postEvent ■	handle	systemCommandEvent
refresh ■	helpId ■	visibilityDisabledEvent ■
releasePointer	hiliteBackgroundColor ■ ►	visibilityEnabledEvent ■
releasePresSpace ■	hiliteForegroundColor ■ ►	*visible*

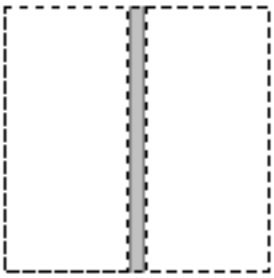
ISlider

Action	Attribute	Event
removeDetent ■	homePosition ■	
resetActiveColor	id ■	
resetBackgroundColor	inactiveColor ■ ►	
resetBorderColor	itemProvider ■	
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	owner ■	
resetHiliteForegroundColor	parent ■	
resetInactiveColor	parentSize	
resetMinimumSize	pointerCaptured	
resetShadowColor	position ■ ►	
sendEvent ■	presSpace	
setFocus	primaryScale ■ ►	
setLayoutDistorted ■	rect ■	
setShaftBreadth ■	ribbonStripEnabled ■	
setTickLength ■	shadowColor ■ ►	
setTicks ■	shaftPosition ■	
setTickText ■	shaftSize	
show ■	showing	
showSourceEmphasis ■	size ■ ►	
tickLength ■	snapToTickEnabled ■	
tickPosition ■	tabStop ■	
tickSpacing ■	**this**	
tickText ■	valid	
	vertical	
	visible ■ ►	
	visibleRectangle	

Chapter 63. ISplitCanvas

Description: IBM canvas divided into resizable panes
Part type: Visual
Part file: vbbase
Header file: isplitcv.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
ISplitCanvas



Use an ISplitCanvas* part to provide split windows.

You can use an ISplitCanvas* part in another composer part, such as the following:

- IFrameWindow* client area
- IMultiCellCanvas*
- IVBNotebookPage*

Then, you can add a control to each pane of the canvas within the other composer part.

You can also use a split canvas as the base part for a composite part. Then, you can add the canvas with its controls as a subpart in another composite part.

Alternatively, use one of the following canvas parts:

- ICanvas* for a simple canvas
- IDrawingCanvas* for graphic objects
- IMultiCellCanvas* to provide freely arranged canvas cells
- ISetCanvas* for canvas cells in uniform rows or columns

© Copyright IBM Corp. 1995, 1996

The user can move a split bar, which separates panes in the split canvas, by dragging it.

Portability — ISplitCanvas

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ISplitCanvas

Canvas parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “ISetCanvas” on page 293
- “ISplitCanvas” on page 307

Client parts:

- “ICanvas” on page 83
- “IDrawingCanvas” on page 145
- “IMultiCellCanvas” on page 225
- “IMultiLineEdit” on page 231
- “INotebook” on page 237
- “ISetCanvas” on page 293

- "ISplitCanvas" on page 307
- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — ISplitCanvas

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding Panes
- Selecting the Orientation
- Reordering Panes

Adding the Part

To add an ISplitCanvas* part to a composite part, select an ISplitCanvas* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- An IViewPort* part
- An IVBNotebookPage* part
- A canvas part
- The free-form surface

The canvas is initially filled with a primitive part.

Adding Panes

To create and fill the panes for the split window, select canvas or view port parts from the parts palette and drop them on the split canvas. For example, to create three panes with view ports in them, do the following:

1. Drop an IViewPort* part on the primitive part that initially fills the split canvas. A split bar and two panes appear, with the view port in the first pane.
2. Drop an IViewPort* part on the primitive part or on the split bar. The new view port replaces the primitive part in the second pane.
3. Drop an IViewPort* part on the split bar. A second split bar and third pane appear, with the new view port in the third pane.

Selecting the Orientation

To determine the orientation of window splits, select a choice in the **Orientation** field on the ISplitCanvas* part's **General** settings page.

By default, the split orientation is vertical. To change the orientation to horizontal, select the **Horizontal** setting in the **Orientation** field.

To open the settings view for an ISplitCanvas* part after panes have been created, open its contextual menu on a split bar.

Reordering Panes

To reorder panes on the canvas, open the ISetCanvas* part's contextual menu and select **View parts list**. Then, in the parts list window, drag the primary parts in panes to reorder the panes.

Preferred Features — ISplitCanvas

menu Popup menu link.

orientation Returns an enumerator Orientation for the orientation of the canvas's split bars.

setFocus Sets the input focus to the window.

setSplitWindowPercentage Sets the percentage of the canvas occupied by the specified window.

splitWindowPercentage Returns the percentage of the width or height of the canvas currently occupied by the specified window.

this A Pointer to the instance.

Features — ISplitCanvas

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	defaultPushButton	*font*
enableUpdate ■	disabledBackgroundColor ■ ►	*foregroundColor*
handleException ■	disabledForegroundColor ■ ►	gotFocusEvent ■
hide	enabled ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	focus ►	*inactiveColor*
matchForMnemonic ■	font ■ ►	inputDisabledEvent ■

ISplitCanvas

Action	Attribute	Event
notifyObservers ■	foregroundColor ■ ►	inputEnabledEvent ■
positionBehindSibling ■	frameWindow	lostFocusEvent ■
positionBehindSiblings	group ■	***orientation***
positionOnSiblings	handle	*position*
postEvent ■	helpId ■	*shadowColor*
refresh ■	hiliteBackgroundColor ■ ►	*size*
releasePointer	hiliteForegroundColor ■ ►	systemCommandEvent
releasePresSpace ■	id ■	visibilityDisabledEvent ■
resetActiveColor	inactiveColor ■ ►	visibilityEnabledEvent ■
resetBackgroundColor	itemProvider ■	*visible*
resetBorderColor	layoutAdjustment	
resetDisabledBackgroundColor	menu	
resetDisabledForegroundColor	minimumSize ■	
resetFont	nativeRect	
resetForegroundColor	**orientation** ■ ►	
resetHiliteBackgroundColor	origDefaultButtonHandle	
resetHiliteForegroundColor	owner ■	
resetInactiveColor	parent ■	
resetMinimumSize	parentSize	
resetShadowColor	pointerCaptured	
resetSplitBarEdgeColor	position ■ ►	
resetSplitBarMiddleColor	presSpace	
sendEvent ■	rect ■	
setFocus	shadowColor ■ ►	
setLayoutDistorted ■	showing	
setSplitBarThickness ■	size ■ ►	
setSplitWindowPercentage ■	**splitBarEdgeColor** ■	
show ■	**splitBarMiddleColor** ■	
showSourceEmphasis ■	tabStop ■	
splitBarThickness ■	**this**	
splitWindowPercentage ■	valid	
	visible ■ ►	
	visibleRectangle	

ISplitCanvas

Chapter 64. IStandardNotifier

Description: IBM base class for all nonvisual VB parts
Part type: Nonvisual
Part file: vbbase
Header file: istdntfy.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier

Use an IStandardNotifier* part as a base part for nonvisual parts you create.

Preferred Features — IStandardNotifier

enabledForNotification
Returns `true` if an object is sending notifications to its observers.

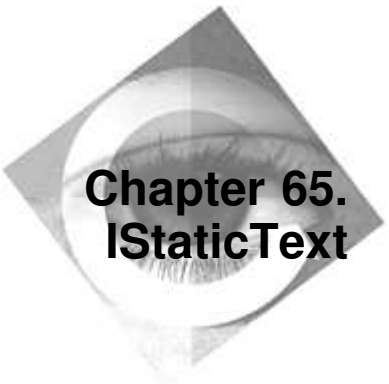
this A Pointer to the instance.

Features — IStandardNotifier

Action	Attribute	Event
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	
operator = ■	**enabledForNotification** ■	
	this	

© Copyright IBM Corp. 1995, 1996

IStandardNotifier



Chapter 65. IStaticText

Description: IBM static text control
Part type: Visual
Part file: vbbase
Header file: istattxt.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IStaticText

Use an IStaticText* part to show read-only text, such as a label for a control. The user cannot modify a static text field.

Portability — IStaticText

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

© Copyright IBM Corp. 1995, 1996

Related Parts — IStaticText

Label parts:

- "IGroupBox" on page 181
- "IOutlineBox" on page 249
- "IStaticText" on page 315

Building Guidelines — IStaticText

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining a Label
- Defining Instructions
- Showing a Block
- Showing the Date
- Selecting a Color
- Selecting a Font

Adding the Part

To add an IStaticText* part to a composite part, select an IStaticText* part from the parts palette. Then, drop the part on a canvas.

Defining a Label

To define a label for a control, enter the label you want in the **Text** field on the IStaticText* part's **General** settings page.

Defining Instructions

To provide instructions for a field, do the following:

- Enter the instructions in the **Text** field on the IStaticText* part's **General** settings page.
- Select a starting position for the text in the **Alignment** field on the IStaticText* part's **General** settings page.
- To extend instructions to multiple lines, select **Word wrap** on the IStaticText* part's **General** settings page.

Showing a Block

To display a block of color with no text, do the following:

- Clear the **Text** field on the IStaticText* part's **General** settings page.
- Select a fill color on the IStaticText* part's **Color** settings page as follows:

- Select **Fill** in the **Color area** group.
- Either select a choice from the **Colors** list or specify the **RGB values** you want.

Showing the Date

To show the date, do the following:

1. Add an IDate* part to the free-form surface.
2. Open the IDate* part's contextual menu and select the **Tear-off attribute** choice.
3. Select the *today* attribute. A today IVBVariable* part is added to the free-form surface and connected to the IDate* part.
4. Connect the today IVBVariable* part's *asString* attribute to the IStaticText* part's *text* attribute.

Selecting a Color

To select a color for the static text, do the following:

- Select **Foreground** in the **Color area** group on the IStaticText* part's **Color** settings page.
- Either select a choice from the **Colors** list or specify the **RGB values** you want.

Selecting a Font

To select a font for the static text, select the **Edit** choice on the IStaticText* part's **Font** settings page. Then, select the font you want.

If you placed the IStaticText* part on an ISetCanvas* part or on an IMultiCellCanvas* part, the IStaticText* part is dynamically expanded for font changes if more space is required.

Preferred Features — IStaticText

halftone	Queries whether the text has the style halftone.
limit	Returns the number of characters set by setLimit.
menu	Popup menu link.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.
valueAsInt	Return asInt from the `IString` text attribute

Features — IStaticText

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**alignment** ■	anyEvent
convertToGUIStyle ■	asDebugInfo	*backgroundColor*
disable	asString	*borderColor*
disableFillBackground	autoDeleteObject ■	commandEvent
disableGroup	autoDestroyWindow ■	*disabledBackgroundColor*
disableHalftone	backgroundColor ■ ►	*disabledForegroundColor*
disableNotification	bidiSupported	*enabled*
disableStrikeout	borderColor ■ ►	***fillBackground***
disableTabStop	characterSize	***fillColor***
disableUnderscore	clipboardHasTextFormat	*focus*
disableUpdate	defaultPushButton	*font*
dispatchRemainingHandlers ■	disabledBackgroundColor ■ ►	*foregroundColor*
enable ■	disabledForegroundColor ■ ►	gotFocusEvent ■
enableUpdate ■	displaySize	***halftone***
handleException ■	enabled ■ ►	*hiliteBackgroundColor*
hide	enabledForNotification ■	*hiliteForegroundColor*
hideSourceEmphasis	**fillBackground** ■ ►	*inactiveColor*
isLayoutDistorted ■	**fillColor** ■ ►	inputDisabledEvent ■
matchForMnemonic ■	focus ►	inputEnabledEvent ■
notifyObservers ■	font ■ ►	*isDigits*
positionBehindSibling ■	foregroundColor ■ ►	***limit***
positionBehindSiblings	frameWindow	lostFocusEvent ■
positionOnSiblings	group ■	*position*
postEvent ■	**halftone** ■ ►	*shadowColor*
refresh ■	handle	*size*
releasePointer	helpId ■	***strikeout***
releasePresSpace ■	hiliteBackgroundColor ■ ►	systemCommandEvent
resetActiveColor	hiliteForegroundColor ■ ►	*text*
resetBackgroundColor	id ■	*textLength*
resetBorderColor	inactiveColor ■ ►	***underscore***
resetDisabledBackgroundColor	isDigits ►	*valueAsDouble*
resetDisabledForegroundColor	itemProvider ■	*valueAsInt*
resetFillColor	layoutAdjustment	*valueAsUnsigned*
resetFont	**limit** ■ ►	visibilityDisabledEvent ■
resetForegroundColor	menu	visibilityEnabledEvent ■
resetHiliteBackgroundColor	minimumSize ■	*visible*
resetHiliteForegroundColor	nativeRect	
resetInactiveColor	owner ■	
resetMinimumSize	parent ■	
resetShadowColor	parentSize	
sendEvent ■	pointerCaptured	
setFocus	position ■ ►	
setLayoutDistorted ■	presSpace	

IStaticText

Action	Attribute	Event
show ■	rect ■	
showSourceEmphasis ■	shadowColor ■ ►	
	showing	
	size ■ ►	
	strikeout ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	underscore ■ ►	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

IStaticText

Chapter 66. IString

Description: IBM string class, indexed starting at 1
Part type: Class
Part file: vbbase
Header file: istring.hpp

Derivation:
IBase
IString

Preferred Features — IString

asDouble	Returns, as a `double`, the number that the string represents.
asInt	Returns the number that the string represents as a `long` integer.
asUnsigned	Returns, as an `unsigned long`, the integer that the string represents.
contents	Returns the string's contents.
copy	Replaces the receiver's contents with a specified number of replications of itself.
digits	If all the characters are in {'0'-'9'}, `true` is returned.
remove	Deletes the specified portion of the string (that is, the substring) from the receiver.
size	Returns the length of the string, not counting the terminating `NULL` character.
this	A Pointer to the instance.

Features — IString

Action	Attribute	Type
b2c ■	**alphabetic**	`Boolean`
b2d ■	**alphanumeric**	`Boolean`
b2x ■	asDebugInfo	`IString`
c2b	**asDouble**	`double`
c2d	**asInt**	`long`
c2x ■	asString	`IString`
center ■	**asUnsigned**	`unsigned long`
change ■	**binaryDigits**	`Boolean`
charType ■	**contents** ■	`IString`
copy ■	**control**	`Boolean`
d2b ■	**digits**	`Boolean`

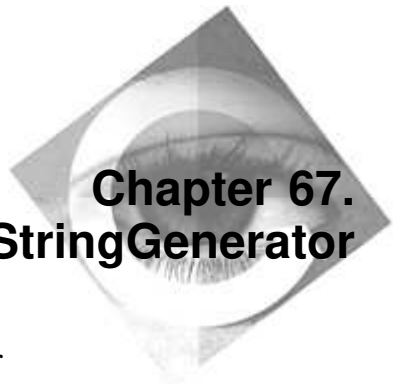
© Copyright IBM Corp. 1995, 1996

IString

Action	Attribute	Type
d2c	**graphics**	Boolean
d2x	**hexDigits**	Boolean
includes ■	**includesDBCS**	Boolean
indexOf ■	**includesMBCS**	Boolean
indexOfAnyBut ■	**includesSBCS**	Boolean
indexOfAnyOf ■	**isASCII**	Boolean
indexOfPhrase ■	**isDBCS**	Boolean
indexOfWord ■	**isLowerCase**	Boolean
insert ■	**isMBCS**	Boolean
isAbbreviationFor ■	**isSBCS**	Boolean
isLike ■	**isUpperCase**	Boolean
lastIndexOf ■	**length**	unsigned
lastIndexOfAnyBut ■	**numWords**	unsigned
lastIndexOfAnyOf ■	**printable**	Boolean
leftJustify ■	**punctuation**	Boolean
lengthOfWord ■	**size**	unsigned
lineFrom ■	**this**	IString*
lowerCase ■	**validDBCS**	Boolean
occurrencesOf ■	**validMBCS**	Boolean
operator & ■	**whiteSpace**	Boolean
operator &= ■		
operator + ■		
operator += ■		
operator = ■		
operator char *		
operator signed char *		
operator unsigned char *		
operator [] ■		
operator ^ ■		
operator ^= ■		
operator | ■		
operator |= ■		
operator ~		
overlayWith ■		
remove ■		
removeWords ■		
reverse ■		
rightJustify ■		
space ■		
strip ■		
stripBlanks ■		
stripLeading ■		
stripLeadingBlanks ■		
stripTrailing ■		
stripTrailingBlanks ■		
subString ■		
translate ■		

Action	Attribute	Type
upperCase ■		
word ■		
wordIndexOfPhrase ■		
words ■		
x2b		
x2c ■		
x2d ■		

IString



Chapter 67. IStringGenerator

Description: IBM string generator template class.
Part type: Class
Part file: vbbase
Header file: istrgen.hpp

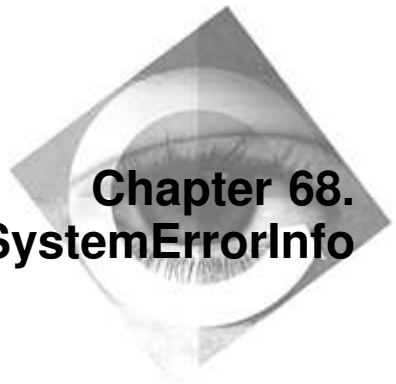
Derivation:
IStringGenerator

Features — IStringGenerator

Action	Attribute	Type
	this	`IStringGenerator*`

© Copyright IBM Corp. 1995, 1996

IStringGenerator



Chapter 68. ISystemErrorInfo

Description: IBM access to system error information
Part type: Class
Part file: vbbase
Header file: iexcept.hpp

Derivation:
IBase
IVBase
IErrorInfo
ISystemErrorInfo

Preferred Features — ISystemErrorInfo

text	Returns the error text.
this	A Pointer to the instance.

Features — ISystemErrorInfo

Action	Attribute	Type
operator const char *	asDebugInfo	`IString`
throwError ■	asString	`IString`
throwSystemError ■	**available**	`Boolean`
	errorId	`unsigned long`
	text	`const char*`
	this	`ISystemErrorInfo*`

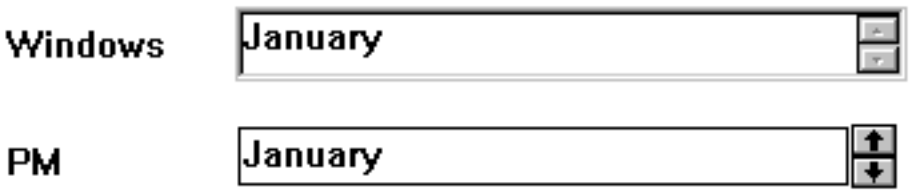
© Copyright IBM Corp. 1995, 1996

ISystemErrorInfo

Chapter 69. ITextSpinButton

Description: IBM text spin-button control
Part type: Visual
Part file: vbbase
Header file: ispintxt.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IBaseSpinButton
ITextSpinButton



Use an ITextSpinButton* part to provide, in sequence, a ring of related but mutually exclusive text choices. One choice is visible at a time. For example, use a text spin button for a selection list of the months of the year by name.

You can use either a spin button style that is native to the system or a style that is compatible with the OS/2 Presentation Manager (PM) spin button. The two styles are functionally equivalent.

Alternatively, use an INumericSpinButton* part for a ring of numeric choices.

The user can spin the ring upward or downward to select another choice.

Portability — ITextSpinButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Features active with restriction on Windows:

- alignment

Related Parts — ITextSpinButton

Selection parts:

- “ICollectionViewComboBox” on page 105
- “ICollectionViewListBox” on page 113
- “IComboBox” on page 123
- “IListBox” on page 199
- “INumericSpinButton” on page 243
- “ITextSpinButton” on page 329

Button parts:

- “IAnimatedButton” on page 73
- “ICustomButton” on page 137
- “IGraphicPushButton” on page 175
- “INumericSpinButton” on page 243
- “IPushButton” on page 267
- “IRadioButton” on page 273
- “ITextSpinButton” on page 329
- “IToolBarButton” on page 349

Building Guidelines — ITextSpinButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting Initial Contents
- Setting the Initial Choice
- Preventing User Changes
- Enabling Fast Spinning
- Selecting PM Compatibility

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Getting the Selected Choice

Adding the Part

To add an ITextSpinButton* part to a composite part, select an ITextSpinButton* part from the parts palette. Then, drop the part on a canvas.

Setting Initial Contents

To define the initial string choices for the spin button, enter the strings in the **Contents** field on the ITextSpinButton* part's **General** settings page.

Setting the Initial Choice

To define a default selection for the spin button, enter one of the strings from the **Contents** field in the **Text** field on the ITextSpinButton* part's **General** settings page.

Preventing User Changes

To prevent the user from typing a choice, set the **readOnly** field on the ITextSpinButton* part's **Styles** settings page.

Enabling Fast Spinning

To make the spin button skip values to cycle faster when the user continues to spin the button, set the **fastSpin** field on the ITextSpinButton* part's **Styles** settings page.

Selecting PM Compatibility

To provide a PM-compatible spin button, set the **pmCompatible** field on the ITextSpinButton* part's **Styles** settings page.

Getting the Selected Choice

To copy the selection from the spin button to a text field, connect the ITextSpinButton* part's *text* attribute to the target text field.

Preferred Features — ITextSpinButton

addAsLast	Adds the item as the last item.
disable	Prevents keyboard and mouse input from being sent to the window.
enable	Enables the window to accept keyboard and mouse input.
enabled	If the window is sent mouse and keyboard input, `true` is returned.
limit	Returns the number of characters permitted in the spin field.
menu	Popup menu link.
removeAll	Removes all items in the spin button.
setFocus	Sets the input focus to the window.
spinFieldValid	If the contents of the spin field matches one of the text values in the text array, `true` is returned.
spinTo	Spins the button to the specified cursor or index position.
text	Returns the displayed contents of the spin field.
this	A Pointer to the instance.
valueAsInt	Return asInt from the `IString` text attribute

Features — ITextSpinButton

Action	Attribute	Event
add ■	activeColor ■ ►	*activeColor*
addAsFirst ■	alignment ■	anyEvent
addAsLast ■	asDebugInfo	*backgroundColor*
addBorder ■	asString	*borderColor*
applyBidiSettings ■	autoDeleteObject ■	commandEvent
capturePointer ■	autoDestroyWindow ■	*disabledBackgroundColor*
convertToGUIStyle ■	backgroundColor ■ ►	*disabledForegroundColor*
disable	bidiSupported	*enabled*
disableDataUpdate	border	*focus*
disableFastSpin	borderColor ■ ►	*font*
disableGroup	characterSize	*foregroundColor*
disableNotification	defaultPushButton	gotFocusEvent ■
disableTabStop	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
disableUpdate	disabledForegroundColor ■ ►	*hiliteForegroundColor*
dispatchRemainingHandlers ■	enabled ■ ►	*inactiveColor*
elementAt ■	enabledForNotification ■	inputDisabledEvent ■
enable ■	fastSpinEnabled ■	inputEnabledEvent ■
enableUpdate ■	focus ►	***isDigits***
handleException ■	font ■ ►	lostFocusEvent ■
hide	foregroundColor ■ ►	*position*
hideSourceEmphasis	frameWindow	*shadowColor*

ITextSpinButton

Action	Attribute	Event
isLayoutDistorted ■	group ■	*size*
matchForMnemonic ■	handle	systemCommandEvent
notifyObservers ■	helpId ■	***text***
positionBehindSibling ■	hiliteBackgroundColor ■ ►	***textLength***
positionBehindSiblings	hiliteForegroundColor ■ ►	***valueAsDouble***
positionOnSiblings	id ■	***valueAsInt***
postEvent ■	inactiveColor ■ ►	***valueAsUnsigned***
refresh ■	**isDigits** ►	visibilityDisabledEvent ■
releasePointer	itemProvider ■	visibilityEnabledEvent ■
releasePresSpace ■	layoutAdjustment	*visible*
removeAll	limit ■	
removeBorder	master	
resetActiveColor	menu	
resetBackgroundColor	minimumSize ■	
resetBorderColor	nativeRect	
resetDisabledBackgroundColor	owner ■	
resetDisabledForegroundColor	parent ■	
resetFont	parentSize	
resetForegroundColor	pointerCaptured	
resetHiliteBackgroundColor	position ■ ►	
resetHiliteForegroundColor	presSpace	
resetInactiveColor	rect ■	
resetMinimumSize	servant	
resetShadowColor	shadowColor ■ ►	
sendEvent ■	showing	
setFocus	size ■ ►	
setLayoutDistorted ■	**spinFieldValid**	
setMaster ■	tabStop ■	
show ■	**text** ■ ►	
showSourceEmphasis ■	**textLength** ►	
spinDown ■	**this**	
spinTo ■	valid	
spinUp ■	**valueAsDouble** ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	
	writeable ■	

ITextSpinButton

Chapter 70. ITime

Description: IBM time class
Part type: Class
Part file: vbbase
Header file: itime.hpp

Derivation:
IBase
ITime

Use an ITime* part to query and format time values.

Portability — ITime

Features not defined for Windows:

- asCTIME

Preferred Features — ITime

asSeconds	Returns the number of seconds since midnight.
hours	Returns the number of hours past midnight.
minutes	Returns the number of minutes past the hour.
now	Returns the current time.
seconds	Returns the number of seconds past the minute.
this	A Pointer to the instance.

Features — ITime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	**asSeconds**	`long`
operator += ■	asString	`IString`
operator - ■	**hours**	`unsigned`
operator -= ■	**minutes**	`unsigned`
operator < ■	**now**	`ITime`
operator <= ■	**seconds**	`unsigned`
operator == ■	**this**	`ITime*`
operator > ■		
operator >= ■		

© Copyright IBM Corp. 1995, 1996

ITime

Chapter 71. ITitle

Description: IBM window-title textcontrol
Part type: Nonvisual
Part file: vbbase
Header file: ititle.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
ITitle

Use an ITitle* part to provide a frame window title that your application can change while the application is running.

The user can change the title if your application provides the capability through other controls. Otherwise, the user can only observe title changes made by your application.

Portability — ITitle

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeTextBackgroundColor
- activeTextForegroundColor
- inactiveTextBackgroundColor
- inactiveTextForegroundColor
- resetActiveTextBackgroundColor
- resetActiveTextForegroundColor
- resetInactiveTextBackgroundColor
- resetInactiveTextForegroundColor
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor

© Copyright IBM Corp. 1995, 1996

- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — ITitle

Frame parts:

- "IFrameWindow" on page 167
- "IMenu" on page 205
- "IScrollBar" on page 287
- "ITitle" on page 337
- "IToolBar" on page 343
- "IVBInfoArea" on page 385

Building Guidelines — ITitle

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Assigning Ownership
- Defining the Title
- Loading the Title

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Title

Adding the Part

To add an ITitle* part to a composite part, select an ITitle* part from the parts palette. Then, drop the part on the free-form surface.

Assigning Ownership

If you have only one frame window in your view, the title is automatically owned by that frame window.

To provide the title for a second frame window in your view, reference the IFrameWindow* part's **Subpart name**, which appears on its **General** settings page. Enter the frame window subpart name, prefixed with "i," in the **Owner** field on the ITitle* part's **General** settings page.

Defining the Title

To define an initial title, do the following:

- Enter the object name or desired title in the **Object text** field on the ITitle* part's **General** settings page:
- Optionally, enter the view name in the **View text** field on the ITitle* part's **General** settings page:
- Optionally, enter the view number in the **View number** field on the ITitle* part's **General** settings page:

Loading the Title

To load the title from one or more attributes of other parts, do the following:

- Connect a text attribute to the ITitle* part's *objectText* attribute.
- Optionally, connect a text attribute to the ITitle* part's *viewName* attribute.
- Optionally, connect a numeric attribute to the ITitle* part's *viewNumber* attribute.

Changing the Title

To change the title when an event occurs, do the following:

- Connect the event to the ITitle* part's *setTitleText* action.
- Open settings for the connection, select **Set parameters**, and enter the title in the *objectName* parameter. You can optionally specify *viewName* and *viewNum* parameters for the title.

Preferred Features — ITitle

objectText	Returns the object text.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.
viewNumber	Returns the view number.
viewText	Returns the view text.

ITitle

Features — ITitle

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**activeTextBackgroundColor ■ ►**	***activeTextBackgroundColor***
convertToGUIStyle ■	**activeTextForegroundColor ■ ►**	***activeTextForegroundColor***
disable	asDebugInfo	anyEvent
disableGroup	asString	*backgroundColor*
disableNotification	autoDeleteObject ■	*borderColor*
disableTabStop	autoDestroyWindow ■	commandEvent
disableUpdate	backgroundColor ■ ►	*disabledBackgroundColor*
dispatchRemainingHandlers ■	bidiSupported	*disabledForegroundColor*
enable ■	borderColor ■ ►	*enabled*
enableUpdate ■	characterSize	*focus*
handleException ■	clipboardHasTextFormat	*font*
hide	defaultPushButton	*foregroundColor*
hideSourceEmphasis	disabledBackgroundColor ■ ►	gotFocusEvent ■
isLayoutDistorted ■	disabledForegroundColor ■ ►	*hiliteBackgroundColor*
matchForMnemonic ■	displaySize	*hiliteForegroundColor*
notifyObservers ■	enabled ■ ►	*inactiveColor*
positionBehindSibling ■	enabledForNotification ■	***inactiveTextBackgroundColor***
positionBehindSiblings	focus ►	***inactiveTextForegroundColor***
positionOnSiblings	font ■ ►	inputDisabledEvent ■
postEvent ■	foregroundColor ■ ►	inputEnabledEvent ■
refresh ■	frameWindow	*isDigits*
releasePointer	group ■	lostFocusEvent ■
releasePresSpace ■	handle	***objectText***
resetActiveColor	helpId ■	*position*
resetActiveTextBackgroundColor	hiliteBackgroundColor ■ ►	*shadowColor*
resetActiveTextForegroundColor	hiliteForegroundColor ■ ►	*size*
resetBackgroundColor	id ■	systemCommandEvent
resetBorderColor	inactiveColor ■ ►	*text*
resetDisabledBackgroundColor	**inactiveTextBackgroundColor ■ ►**	*textLength*
resetDisabledForegroundColor	**inactiveTextForegroundColor ■ ►**	*valueAsDouble*
resetFont	isDigits ►	*valueAsInt*
resetForegroundColor	itemProvider ■	*valueAsUnsigned*
resetHiliteBackgroundColor	layoutAdjustment	***viewNumber***
resetHiliteForegroundColor	menu	***viewText***
resetInactiveColor	minimumSize ■	visibilityDisabledEvent ■
resetInactiveTextBackgroundColor	nativeRect	visibilityEnabledEvent ■
resetInactiveTextForegroundColor	**objectText ■ ►**	*visible*
resetMinimumSize	owner ■	
resetShadowColor	parent ■	
sendEvent ■	parentSize	
setFocus	pointerCaptured	

ITitle

Action	Attribute	Event
setLayoutDistorted ■	position ■ ►	
setTitleText ■	presSpace	
show ■	rect ■	
showSourceEmphasis ■	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	viewNumber ■ ►	
	viewText ■ ►	
	visible ■ ►	
	visibleRectangle	

ITitle

Chapter 72. IToolBar

Description: IBM tool-bar control
Part type: Visual
Part file: vbbase
Header file: itbar.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
ISetCanvas
IToolBar

Use an IToolBar* part to give the user a set of controls for commonly used functions. The tool bar is initially adjacent to the client area of the frame window it is related to.

The user can easily perform application functions using tool bar controls. The user can also drag the tool bar away from the frame window or to any side of the frame.

Portability — IToolBar

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor

© Copyright IBM Corp. 1995, 1996

- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IToolBar

Tool bar parts:

- "IToolBar" on page 343
- "IToolBarButton" on page 349

Frame parts:

- "IFrameWindow" on page 167
- "IMenu" on page 205
- "IScrollBar" on page 287
- "ITitle" on page 337
- "IToolBar" on page 343
- "IVBInfoArea" on page 385

Building Guidelines — IToolBar

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting the Frame Location
- Hiding the Tool Bar
- Defining the Title
- Adding Controls
- Setting the Initial View

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the View

Adding the Part

To add an IToolBar* part to a composite part, select an IToolBar* part from the parts palette. Then, drop the part on an IFrameWindow* part. The tool bar is placed in a

frame extension at the top of the client area. One IToolBarButton* part is automatically placed on the tool bar.

Selecting the Frame Location

To set the location of the tool bar, select a choice from the **Location** field on the IToolBar* part's **General** settings page.

If the tool bar is placed above or below the client area, its orientation is horizontal. If it is placed to the left or right of the client area, its orientation is vertical.

To group a second tool bar with one previously added to the frame window, select the **Group with preceding tool bar** field on the **General** settings page of the second IToolBar* part. The tool bar is placed after the first tool bar in the same row or column as the first tool.

Hiding the Tool Bar

To hide the tool bar, select the **Hidden** choice from the **Location** field on the IToolBar* part's **General** settings page.

Defining the Title

When the user drags the tool bar away from the frame window, it appears to float on the desktop. To define title bar text for the floating tool bar, enter text in the **Floating title** field on the IToolBar* part's **General** settings page.

Adding Controls

To add controls to the tool bar, select control parts from the parts palette and drop them on the IToolBar* part. The following controls are commonly added to tool bars:

- Tool bar buttons
- Entry fields
- Drop-down combination boxes
- Check boxes
- Radio buttons
- Push buttons
- Spin buttons

Setting the Initial View

To define the initial view of the tool bar, select the view you want on the IToolBar* part's **General** settings page. Set the one of the following to **On** and set the others to **Off**:

- bitmapVisible to show only bitmaps for tool bar buttons
- buttonTextVisible to show only text for tool bar buttons

- bitmapAndTextVisible to show both bitmaps and text for tool bar buttons

Changing the View

To change the tool bar view when a push button is clicked, connect the IPushButton* part's *buttonClickEvent* feature to the IToolBar* part's *buttonView* attribute. Specify the new view by doing either of the following:

- Define the value in the *buttonClickEvent*-to-*buttonView* connection parameter settings Provide the parameter with a connection to the
- *buttonClickEvent*-to-*buttonView* connection

Preferred Features — IToolBar

addAsFirst Adds the window as the first item in the tool bar.
addAsLast Adds the window as the last item in the tool bar.
addAsNext Adds the window as the next item in the tool bar immediately after the reference window.
allowsDragDrop
Returns `true` if the tool bar supports drag and drop.
menu Popup menu link.
remove Removes the window from the tool bar.
setFocus Sets the input focus to the window.
text Returns the text of the group box.
this A Pointer to the instance.

Features — IToolBar

Action	Attribute	Event
addAsFirst ■	activeColor ■ ►	*activeColor*
addAsLast ■	alignment ■	anyEvent
addAsNext ■	**allowsDragDrop** ■	*backgroundColor*
addAsPrevious ■	asDebugInfo	*borderColor*
applyBidiSettings ■	asString	commandEvent
capturePointer ■	autoDeleteObject ■	*deckOrientation*
collapse	autoDestroyWindow ■	*disabledBackgroundColor*
convertToGUIStyle ■	backgroundColor ■ ►	*disabledForegroundColor*
disable	bidiSupported	*enabled*
disableDragDrop	borderColor ■ ►	*focus*
disableGroup	**buttonView** ■	*font*
disableMisfitFiltering	characterSize	*foregroundColor*
disableNotification	deckCount ■	gotFocusEvent ■
disableTabStop	deckOrientation ■ ►	*hiliteBackgroundColor*
disableUpdate	**defaultGroupPad** ■	*hiliteForegroundColor*
dispatchRemainingHandlers ■	**defaultMargin** ■	*inactiveColor*

IToolBar

Action	Attribute	Event
enable ■	**defaultMisfitWidth** ■	inputDisabledEvent ■
enableUpdate ■	**defaultPad** ■	inputEnabledEvent ■
expand ■	defaultPushButton	lostFocusEvent ■
handleException ■	disabledBackgroundColor ■ ►	*position*
hide	disabledForegroundColor ■ ►	*shadowColor*
hideSourceEmphasis	enabled ■ ►	*size*
isLayoutDistorted ■	enabledForNotification ■	systemCommandEvent
matchForMnemonic ■	**expanded**	*text*
moveAfter ■	**floatingFrame**	visibilityDisabledEvent ■
moveBefore ■	**floatingPosition** ■	visibilityEnabledEvent ■
moveToFirst ■	**floatingTitle** ■	*visible*
moveToLast ■	focus ►	
notifyObservers ■	font ■ ►	
positionBehindSibling ■	foregroundColor ■ ►	
positionBehindSiblings	frameWindow	
positionOnSiblings	group ■	
postEvent ■	groupPad ■	
refresh ■	handle	
releasePointer	helpId ■	
releasePresSpace ■	hiliteBackgroundColor ■ ►	
remove ■	hiliteForegroundColor ■ ►	
resetActiveColor	id ■	
resetBackgroundColor	inactiveColor ■ ►	
resetBorderColor	itemProvider ■	
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	**location** ■	
resetFont	margin ■	
resetForegroundColor	menu	
resetHiliteBackgroundColor	minimumSize ■	
resetHiliteForegroundColor	**misfitFilteringEnabled** ■	
resetInactiveColor	nativeRect	
resetMinimumSize	origDefaultButtonHandle	
resetShadowColor	owner ■	
sendEvent ■	packType ■	
setFocus	pad ■	
setLayoutDistorted ■	parent ■	
show ■	parentSize	
showSourceEmphasis ■	pointerCaptured	
	position ■ ►	
	presSpace	
	rect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	this	

IToolBar

Action	Attribute	Event
	toolBarContainer	
	valid	
	visible ■ ►	
	visibleRectangle	

Chapter 73. IToolBarButton

Description: IBM tool-bar button control
Part type: Visual
Part file: vbbase
Header file: itbarbut.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IButton
ICustomButton
IToolBarButton

Use an IToolBarButton* part to give the user a function choice for a tool bar. The choice is identified by a graphic on the tool bar button.

The user can click the tool bar button to perform the indicated function.

Portability — IToolBarButton

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- allowsMouseClickFocus
- disableMouseClickFocus
- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor

© Copyright IBM Corp. 1995, 1996

- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IToolBarButton

Tool bar parts:

- "IToolBar" on page 343
- "IToolBarButton" on page 349

Button parts:

- "IAnimatedButton" on page 73
- "ICustomButton" on page 137
- "IGraphicPushButton" on page 175
- "INumericSpinButton" on page 243
- "IPushButton" on page 267
- "IRadioButton" on page 273
- "ITextSpinButton" on page 329
- "IToolBarButton" on page 349

Building Guidelines — IToolBarButton

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Using a Predefined Button
- Assigning a Bitmap
- Enabling Button Latching
- Enabling Tab Key Access
- Defining a Control Group
- Reordering Controls

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting the Requested Action
- Disabling the Part
- Enabling the Part

Adding the Part

To add an IToolBarButton* part to a composite part, select an IToolBarButton* part from the parts palette. Then, drop the part on an IToolBar* part.

Using a Predefined Button

To use a predefined button, set the **Select from type list** field and select a button identifier from the **Type list** field for **Predefined button types** on the IToolBarButton* part's **General** settings page. A predefined bitmap is placed on the button.

Assigning a Bitmap

To assign a bitmap to the tool bar button, enter a **DLL name** and a **Resource ID** for the **Bitmap** field on the IToolBarButton* part's **General** settings page.

Enabling Button Latching

To enable latching the button down when it is pressed, set the **latchable** field on the IToolBarButton* part's **Styles** settings page.

Enabling Tab Key Access

To allow the user to reach the button by using a tab key, set the **tabStop** field to **On** on the IToolBarButton* part's **Styles** settings page.

Defining a Control Group

To define a control group beginning with this part, set the **group** field to **On** on the IToolBarButton* part's **Styles** settings page. Then, make the first control after the group the beginning of another control group. You will probably also want to enable tab key access to the first control in the group.

A space appears on the tool bar between the last control of the previous group and the first control of a new group.

The user can move through the controls in a group using the keyboard cursor keys. When the user moves the cursor forward from the last control in the group, the cursor returns to the first control in the group.

This field is initially set for the tool bar button automatically placed on an IToolBar* part when it is added to a frame window.

Reordering Controls

To reorder tool bar buttons and other controls on a tool bar, open the IToolBar* part's contextual menu and select **View parts list**. Then, in the parts list window, drag parts within the list to reorder them.

Starting the Requested Action

To start the action represented by the button, connect the IToolBarButton* part's *buttonClickEvent* feature to the action on the target part.

Disabling the Part

To disable the tool bar button when an event occurs, connect the event to the IToolBarButton* part's *disable* action.

Enabling the Part

To enable the tool bar button when an event occurs, do the following:

1. Connect the event to the IToolBarButton* part's *enable* action.
2. The button is enabled by default. If you have changed this default, open settings for the connection, select **Set parameters**, and check the **Enabled** setting.

Preferred Features — IToolBarButton

bitmap Returns the handle of the default bitmap.

buttonClickEvent
Notification identifier provided to observers when the button control is clicked by the user.

disable Prevents keyboard and mouse input from being sent to the window.

enable Enables the window to accept keyboard and mouse input.

enabled If the window is sent mouse and keyboard input, `true` is returned.

latched Returns `true` if the button is in a latched state.

menu Popup menu link.

setFocus Sets the input focus to the window.

text Returns the control window's text.

this A Pointer to the instance.

Features — IToolBarButton

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	**allowsDragDelete** ■	anyEvent
clearDefaultTransparentColor	allowsMouseClickFocus ■	*backgroundColor*
click	asDebugInfo	*borderColor*
convertToGUIStyle ■	asString	buttonClickEvent
disable	autoDeleteObject ■	commandEvent
disableAutoLatch	autoDestroyWindow ■	*disabledBackgroundColor*
disableDragDelete	autoLatchEnabled ■	*disabledForegroundColor*
disableGroup	backgroundColor ■ ►	*enabled*
disableLatching	bidiSupported	*focus*
disableMouseClickFocus	**bitmap** ■	*font*

IToolBarButton

Action	Attribute	Event
disableNotification	**bitmapSize**	*foregroundColor*
disableTabStop	**bitmapVisible**	gotFocusEvent ■
disableUpdate	borderColor ■ ►	*hiliteBackgroundColor*
dispatchRemainingHandlers ■	characterSize	*hiliteForegroundColor*
enable ■	clipboardHasTextFormat	*inactiveColor*
enableUpdate ■	defaultPushButton	inputDisabledEvent ■
handleException ■	**defaultTransparentColor** ■	inputEnabledEvent ■
hide	**defaultTransparentColorSet**	*isDigits*
hideSourceEmphasis	disabledBackgroundColor ■ ►	*latched*
highlight ■	disabledForegroundColor ■ ►	lostFocusEvent ■
isLayoutDistorted ■	displaySize	*position*
matchForMnemonic ■	enabled ■ ►	*shadowColor*
notifyObservers ■	enabledForNotification ■	*size*
positionBehindSibling ■	focus ►	systemCommandEvent
positionBehindSiblings	font ■ ►	*text*
positionOnSiblings	foregroundColor ■ ►	*textLength*
postEvent ■	frameWindow	*valueAsDouble*
refresh ■	group ■	*valueAsInt*
releasePointer	handle	*valueAsUnsigned*
releasePresSpace ■	**hasTransparentColor**	visibilityDisabledEvent ■
resetActiveColor	helpId ■	visibilityEnabledEvent ■
resetBackgroundColor	highlighted	*visible*
resetBorderColor	hiliteBackgroundColor ■ ►	
resetDisabledBackgroundColor	hiliteForegroundColor ■ ►	
resetDisabledForegroundColor	id ■	
resetFont	inactiveColor ■ ►	
resetForegroundColor	isDigits ►	
resetHiliteBackgroundColor	itemProvider ■	
resetHiliteForegroundColor	latched ■ ►	
resetInactiveColor	latchedBackgroundColor ■	
resetLatchedBackgroundColor	latchedBackgroundColorHalftone	
resetLatchedForegroundColor	**latchedBitmap** ■	
resetMinimumSize	latchedForegroundColor ■	
resetShadowColor	latchingEnabled ■	
resetTransparentColor	layoutAdjustment	
sendEvent ■	menu	
setFocus	minimumSize ■	
setLayoutDistorted ■	nativeRect	
show ■	owner ■	
showSourceEmphasis ■	parent ■	
unhighlight	parentSize	
unlatch	pointerCaptured	
	position ■ ►	
	presSpace	
	rect ■	
	shadowColor ■ ►	
	showing	

IToolBarButton

Action	Attribute	Event
	size ■ ►	
	standardBitmapSize ■	
	standardFormat	
	standardTextLines ■	
	standardTextWidth ■	
	tabStop ■	
	text ■ ►	
	textLength ►	
	textVisible	
	this	
	transparentColor ■	
	userData ■	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	view ■	
	visible ■ ►	
	visibleRectangle	

Chapter 74. ITrace

Description: IBM support for module tracing
Part type: Class
Part file: vbbase
Header file: itrace.hpp

Derivation:
IBase
IVBase
ITrace

Preferred Features — ITrace

disableTrace Disables trace entries from being written.
enableTrace Enables trace entries to be written.
this A Pointer to the instance.
traceEnabled Determines whether tracing is currently enabled.
write Writes the specified text.
writeToStandardError
Sets the location for output to the standard error stream.

Features — ITrace

Action	Attribute	Type
disableTrace	asDebugInfo	`IString`
disableWriteLineNumber	asString	`IString`
disableWritePrefix	**this**	`ITrace*`
enableTrace	**traceDestination**	`ITrace::Destination`
enableWriteLineNumber	**traceEnabled**	`Boolean`
enableWritePrefix	**writeLineNumberEnabled**	`Boolean`
write ■	**writePrefixEnabled**	`Boolean`
writeToQueue		
writeToStandardError		
writeToStandardOutput		

© Copyright IBM Corp. 1995, 1996

ITrace

Chapter 75. IVBCheckMenuHandler

Description: IBM VB check menu handler class
Part type: Class
Part file: vbbase
Header file: ivbmenuh.hpp

Derivation:
IBase
IVBase
IHandler
IMenuHandler
IVBCheckMenuHandler

Preferred Features — IVBCheckMenuHandler

this A Pointer to the instance.

Features — IVBCheckMenuHandler

Action	Attribute	Type
disable	asDebugInfo	`IString`
menuSelected ■	asString	`IString`
start ■	enabled ■	`Boolean`
stop ■	**this**	`IVBCheckMenuHandler*`

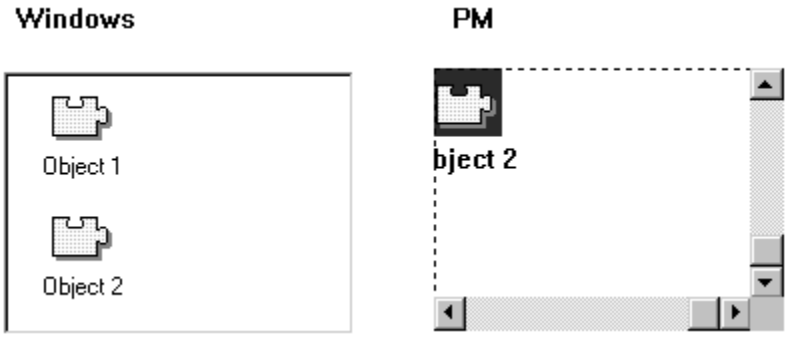
© Copyright IBM Corp. 1995, 1996

IVBCheckMenuHandler

Chapter 76. IVBContainerControl

Description: IBM VB container control
Part type: Visual
Part file: vbbase
Header file: ivbcnr.h

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
IContainerControl
IVBContainerControl



Use an IVBContainerControl* part to let the user work with objects in a container. Containers are useful for presenting collections of nonvisual model objects. File objects and program objects are common examples of objects managed in containers.

You can use either a container style that is native to the system or a style that is compatible with the OS/2 Presentation Manager (PM) container. These containers styles differ considerably in their details views.

The PM-compatible container can automatically size columns according to the data they show. It also allows different icons in any number of columns. The user can select an object with the mouse by clicking on the object row in any column.

The Windows native container does not automatically size columns for data, but the user can dynamically resize columns in your application by dragging column separators. The container shows only the object icon specified for the container. The icon is always defined in column one. The object text specified for the container is always defined in column two. No column separator appears between these two columns, so they appear as a single object identification column. The user can select

© Copyright IBM Corp. 1995, 1996

an object with the mouse by clicking on the object row only in the object identification column.

The user can open a container in one of the following views to work with the objects in the container:

Text and flowed text
List text representations of objects in a single column. The text view lists objects in a single column. The flowed text view lists them in multiple columns.

Name and flowed name
List text representations of objects, with a small icon of the object preceding the text. The name view lists objects in a single column. The flowed name view lists them in multiple columns.

Icon
Represents objects as icons, which can be freely positioned or arranged in rows within the container.

Tree (text, name, and icon)
Represent objects hierarchically. Objects capable of containing nested objects branch from the root of the container. The text, name, and icon tree views represent objects as their corresponding nontree views do.

The tree views are often used for folder hierarchies.

Details
Provides detailed information about each object in the container. The information is presented as a table, with a row for each object and a column for each represented object attribute.

Portability — IVBContainerControl

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor

- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IVBContainerControl

Container parts:

- "IContainerColumn" on page 131
- "IContainerObject" on page 135
- "IVBContainerControl" on page 359

Collection parts:

- "ICollectionViewComboBox" on page 105
- "ICollectionViewListBox" on page 113
- "IVBContainerControl" on page 359
- "IVSequence" on page 403

Client parts:

- "ICanvas" on page 83
- "IDrawingCanvas" on page 145
- "IMultiCellCanvas" on page 225
- "IMultiLineEdit" on page 231
- "INotebook" on page 237
- "ISetCanvas" on page 293
- "ISplitCanvas" on page 307
- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — IVBContainerControl

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining the Title
- Setting the Object Type
- Setting Object Representation
- Setting the Initial View

- Selecting PM Compatibility
- Adding a Column
- Reordering Columns
- Automatically Positioning Icons
- Defining Selection
- Adding a Menu
- Loading Objects

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting the Requested Action
- Getting the Selected Objects

Adding the Part

To add an IVBContainerControl* part to a composite part, select an IVBContainerControl* part from the parts palette. Then, drop the part on another composer part.

If you want to fill the client area of a frame window with a container, we recommend that you use an IMultiCellCanvas* part as the frame window client control. Define the IMultiCellCanvas* part with one row and one column, both expandable. Then, place an IVBContainerControl* part on the single expandable cell of the IMultiCellCanvas* part.

Defining the Title

To define a title for the container, set the **Title attributes** fields on the IVBContainerControl* part's **General** settings page. Enter the title you want in the **Title** field and check the **Show title** setting. If you want a separator below the title, check the **Show title separator** setting. Select the alignment you want in the **Alignment** field.

Setting the Object Type

To specify objects that the container can hold, set the **Container item attributes** fields on the IVBContainerControl* part's **General** settings page. Designate the object type in the **Item type** field.

Setting Object Representation

To define text representation of objects, select a choice in the **Text** field. If you select the asString choice, consider whether you want the default asString result for the object type or need to provide an asString override function to produce more meaningful text.

To define graphic representation of objects, specify object icons in the **Icon** field.

Setting the Initial View

To set the initial view for the container, select a view from the **View type** field on the IVBContainerControl* part's **General** settings page.

Selecting PM Compatibility

To provide a PM-compatible container, set the **pmCompatible** field on the IVBContainerControl* part's **Styles** settings page.

Adding a Column

To add a column for the details view of the container, select an IContainerColumn* part from the parts palette and drop it on the IVBContainerControl* part.

If you use the native Windows container, we strongly recommend that you define the first column to display the object icon and the second column to display the object text. This ensures that the user will see the object identification column when the details view opens.

The *object identification column* is the visual combination of the object icon and the object text. The user can select an object with the mouse only by clicking on the object identification column. If you define the first two columns to contain other data, Windows inserts the object icon and text before the columns you define and initially hides the object identification column. The user must then drag the edge of the first column you define to expose the object identification column.

To work with the IContainerColumn* part in the Composition Editor, set the initial view of the container to be the details view. When you have finished working with container columns, change the initial view to the view you want initially displayed in your application.

Reordering Columns

To reorder container columns, open the IVBContainerControl* part's contextual menu and select **View parts list**. Then, in the parts list window, drag IContainerColumn* parts within the list to reorder them.

Automatically Positioning Icons

To automatically position object icons in the icon view when objects are added to the container, set the **autoPosition** field on the IVBContainerControl* part's **Styles** settings page.

Defining Selection

To define the selection types supported by the container, set one or more of the following selection choices to **On** on the IVBContainerControl* part's **Styles** settings page:

- **extendedSelection**
- **multipleSelection**
- **singleSelection**

Adding a Menu

To add a pop-up menu for the container and objects it contains, select an IMenu* part from the parts palette and drop it on the free-form surface. Then, connect the IMenu* part's *this* attribute to the IVBContainerControl* part's *menu* attribute.

Loading Objects

To load a set of objects into the container, connect the *this* attribute of a collection part containing the choices to the IVBContainerControl* part's *items* attribute. You must use a collection part that has one template argument, such as an IVSequence* part.

Starting the Requested Action

To start an action when the user double-clicks a container object or selects an object and presses the Enter key, connect the IVBContainerControl* part's *enterEvent* feature to the target action.

Getting the Selected Objects

To get objects for an action, do one of the following:

- To get one object when the user double-clicks or presses the Enter key, connect the IVBContainerControl* part's *selectedItem* attribute to the target parameter or attribute. If more than one object is selected, the first selected object is provided to the target.
- To get all selected objects when the user double-clicks or presses the Enter key, connect the IVBContainerControl* part's *selectedItems* attribute to the target parameter or attribute.
- To get one object when the user clicks mouse button 2 on an object, connect the IVBContainerControl* part's *button2Item* attribute to the target parameter or attribute.
- To get one or more objects when the user clicks mouse button 2 on an object, connect the IVBContainerControl* part's *button2Items* attribute to the target parameter or attribute. If the user clicks mouse button 2 on an object that is not selected, only that object is provided to the target. In this case, any selected objects are not affected. If the user clicks mouse button 2 on a selected object, all selected objects are provided to the target.

If you make a connection from either the *selectedItem* attribute or the *button2Item* attribute, the target should expect an object of the type specified for the container.

For example, if the container holds ICustomer* objects, the target should accept an ICustomer* object.

If you make a connection from either the *selectedItems* attribute or the *button2Items* attribute, the target should expect a collection of objects of the type specified for the container. For example, if the container holds ICustomer* objects, the target should accept a collection of ICustomer* objects.

Preferred Features — IVBContainerControl

button2Item Returns the collection element corresponding to the first button2ed item in the container.

button2Items Returns a collection of the items that mouse button2 was clicked over.

enterEvent Notification identifier provided to observers when an item in the container is `double` clicked, or the user presses the enter key.

items Returns the collection currently being viewed.

menu Popup menu link.

selectedCollectionPosition
Returns the collection position corresponding to the first selected item in the container.

selectedCollectionPositions
Returns a collection of the indexes of all selected elements.

selectedItem Returns the item corresponding to the first selected item in the container.

selectedItems Returns a collection of the selected items.

this A Pointer to the instance.

Features — IVBContainerControl

Action	Attribute	Event
addColumn ■	activeColor ■ ►	*activeColor*
applyBidiSettings ■	areDetailsViewTitlesVisible ■	addEvent
arrangeIconView	asDebugInfo	anyEvent
capturePointer ■	asString	*backgroundColor*
closeEdit	autoDeleteObject ■	*borderColor*
collapse ■	autoDestroyWindow ■	***button2CnrItems***
collapseTree	backgroundColor ■ ►	***button2CollectionPosition***
columnAt ■	bidiSupported	***button2CollectionPositions***
columnUnderPoint ■	borderColor ■ ►	***button2Item***
containerFromHandle ■	**button2CnrItems ►**	***button2Items***
containsObject ■	**button2CollectionPosition ►**	***button2Point***
convertToGUIStyle ■	**button2CollectionPositions ►**	commandEvent
convertToWorkspace ■	**button2Item ►**	detailsViewTitlesEvent

IVBContainerControl

Action	Attribute	Event
copyObjectTo ■	**button2Items** ►	*disabledBackgroundColor*
descendentsOf ■	**button2Point** ►	*disabledForegroundColor*
deselect ■	cachingEnabled ■	*enabled*
detailsObjectRectangle ■	characterSize	enterEvent
disable	columnCount	*focus*
disableCaching	currentEditColumn	*font*
disableDataUpdate ■	currentEditMLE	*foregroundColor*
disableDrawBackground	currentEditObject	gotFocusEvent ■
disableDrawItem	**cursoredCollectionPosition**	*hiliteBackgroundColor*
disableDrop ■	**cursoredItem**	*hiliteForegroundColor*
disableGroup	cursoredObject	*inactiveColor*
disableNotification	defaultPushButton	inputDisabledEvent ■
disableTabStop	detailsTitleRectangle	inputEnabledEvent ■
disableTitleUpdate	detailsView	**itemChangedEvent**
disableUpdate	detailsViewPortOnWindow	***items***
dispatchRemainingHandlers ■	detailsViewPortOnWorkspace	lostFocusEvent ■
editColumnTitle ■	detailsViewSplit ■	***numberOfButton2Items***
editContainerTitle	disabledBackgroundColor ■ ►	***numberOfSelections***
editObject ■	disabledForegroundColor ■ ►	*position*
enable ■	drawBackgroundEnabled ■	removeEvent
enableDataUpdate ■	drawItemEnabled ■	***selectedCnrElement***
enableDrop ■	enabled ■ ►	***selectedCnrItem***
enableUpdate ■	enabledForNotification ■	***selectedCnrItems***
expand ■	extendedSelection	***selectedCollectionPosition***
expandTree	flowed	***selectedCollectionPositions***
filter ■	flowedNameView	***selectedElement***
handleException ■	flowedTextView	***selectedItem***
hide	focus ►	***selectedItems***
hideDetailsViewTitles	font ■ ►	selectEvent
hideObject ■	foregroundColor ■ ►	*shadowColor*
hideSourceEmphasis	frameWindow	*size*
hideSplitBar	group ■	systemCommandEvent
hideTitle	handle	*title*
hideTitleSeparator	helpId ■	*titleVisible*
hideTreeLine	hiliteBackgroundColor ■ ►	visibilityDisabledEvent ■
iconRectangle ■	hiliteForegroundColor ■ ►	visibilityEnabledEvent ■
immediateDescendentsOf ■	iconSize ■	*visible*
isCollapsed ■	iconView	
isColumnRight ■	id ■	
isCursored ■	inactiveColor ■ ►	
isDropOnAble ■	**initialize**	
isExpanded ■	itemProvider ■	
isInUse ■	**items** ■ ►	
isLayoutDistorted ■	layoutAdjustment	
isMoveValid ■	lineSpacing ■	
isSelected ■	menu	
isSource ■	minimumSize ■	

IVBContainerControl

Action	Attribute	Event
isTarget ■	mixedTargetEmphasis	
isWriteable ■	multipleSelection	
matchForMnemonic ■	nameView	
moveIconTo ■	nativeRect	
moveObjectTo ■	normalTargetEmphasis	
nlsCompare ■	**numberOfButton2Items** ►	
notifyObservers ■	numberOfColumnChanges	
objectAt ■	numberOfObjectChanges	
objectUnderPoint ■	**numberOfSelections** ►	
operator == ■	objectCount	
parentObject ■	objectList	
positionBehindSibling ■	orderedTargetEmphasis	
positionBehindSiblings	owner ■	
positionOnSiblings	parent ■	
postEvent ■	parentSize	
refresh ■	pointerCaptured	
refreshAllContainers	position ■ ►	
releasePointer	presSpace	
releasePresSpace ■	rect ■	
removeAllItems ■	refreshOn ■	
removeButton2Items ■	**selectedCnrElement** ►	
removeColumn ■	**selectedCnrItem** ►	
removeInUse ■	**selectedCnrItems** ►	
removeSelectedItems ■	**selectedCollectionPosition** ■ ►	
resetActiveColor	**selectedCollectionPositions** ►	
resetBackgroundColor	**selectedElement** ►	
resetBorderColor	**selectedItem** ►	
resetDisabledBackgroundColor	**selectedItems** ►	
resetDisabledForegroundColor	shadowColor ■ ►	
resetFont	showing	
resetForegroundColor	showingMiniIcons	
resetHiliteBackgroundColor	singleSelection	
resetHiliteForegroundColor	size ■ ►	
resetInactiveColor	splitBarOffset	
resetMinimumSize	tabStop ■	
resetShadowColor	textView	
scroll ■	**this**	
scrollDetailsHorizontally ■	title ■ ►	
scrollHorizontally ■	titleRectangle	
scrollToObject ■	titleSeparatorVisible ■	
scrollVertically ■	titleVisible ■ ►	
select ■	titleWriteable ■	
selectedCnrElements ■	treeIconView	
selectedElements ■	treeNameView	
sendEvent ■	treeTextView	
setCursor ■	treeView	
setDeleteColumnsOnClose ■	valid	

IVBContainerControl

Action	Attribute	Event
setDeleteObjectsOnClose ■	viewPortOnWindow	
setEditColumn ■	viewPortOnWorkspace	
setEditMLE ■	visible ■ ►	
setEditObject ■	visibleRectangle	
setExtendedSelection	willDeleteColumnsOnClose	
setFocus	willDeleteObjectsOnClose	
setInUse ■		
setLayoutDistorted ■		
setMixedTargetEmphasis		
setMultipleSelection		
setNormalTargetEmphasis		
setOrderedTargetEmphasis		
setRefreshOff		
setSelected ■		
setSingleSelection		
setTitleAlignment ■		
setTreeExpandIconSize ■		
setTreeItemIcons ■		
setTreeViewIndent ■		
show ■		
showDetailsView		
showFlowedNameView		
showFlowedTextView		
showIconView		
showMiniIcons ■		
showNameView		
showObject ■		
showSourceEmphasis ■		
showSplitBar ■		
showTextView		
showTreeIconView		
showTreeLine ■		
showTreeNameView		
showTreeTextView		
sortByIconText ■		
textRectangle ■		

Chapter 77. IVBDragDropHandler

Description: IBM VB Drag Drop Handler
Part type: Class
Part file: vbbase
Header file: ivbdragh.hpp

Derivation:
IBase
IVBase
IHandler
IVBDragDropHandler

Preferred Features — IVBDragDropHandler

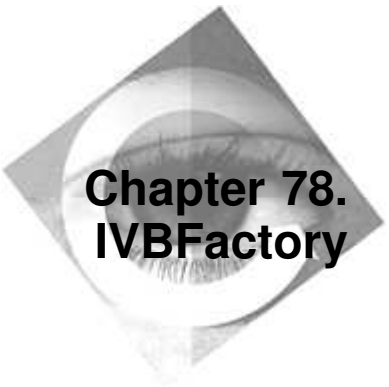
this A Pointer to the instance.

Features — IVBDragDropHandler

Action	Attribute	Type
disable	asDebugInfo	`IString`
start ■	asString	`IString`
stop ■	enabled ■	`Boolean`
	this	`IVBDragDropHandler*`

© Copyright IBM Corp. 1995, 1996

IVBDragDropHandler



Chapter 78. IVBFactory

Description: IBM VB Factory
Part type: Nonvisual
Part file: vbbase
Header file: None

Derivation:
IVBFactory

Use an IVBFactory* part to dynamically create new parts of a type you specify.

The user interacts with parts created by the object factory but does not have access to the factory itself.

Related Parts — IVBFactory

Model parts:

- "IVBFactory"
- "IVBVariable" on page 395
- "IVSequence" on page 403

Building Guidelines — IVBFactory

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Object Type

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Creating Objects

Adding the Part

To add an IVBFactory* part to a composite part, select an IVBFactory* part from the parts palette. Then, drop the part on the free-form surface.

© Copyright IBM Corp. 1995, 1996

IVBFactory

Setting the Object Type

To specify the object type that the factory creates, open the IVBFactory* part's contextual menu. Then, select the **Change type** choice and enter a pointer to the object type you want (ObjectType*).

Creating Objects

To create an object when an event occurs, connect the event to the IVBFactory* part's *new* action. Then, connect the IVBFactory* part's *newEvent* feature to an IVBVariable* part's *this* attribute. You must define the IVBVariable* part to hold an object of the type that the factory creates.

Preferred Features — IVBFactory

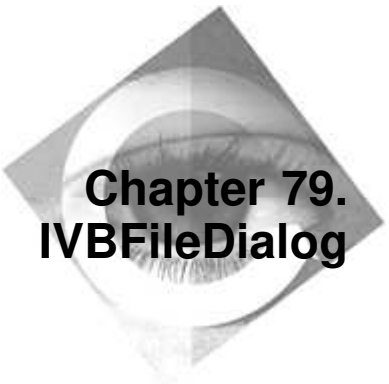
enabledForNotification
: Returns `true` if an object is sending notifications to its observers.

new
: Create a new object.

newEvent
: Creation of a new object.

Features — IVBFactory

Action	Attribute	Event
new	enabledForNotification	**newEvent**



Chapter 79. IVBFileDialog

Description: IBM VB file dialog window
Part type: Nonvisual
Part file: vbbase
Header file: ivbfiled.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBFileDialog

Use an IVBFileDialog* part to represent a file dialog during application building. A file dialog provides an input field and other controls for file identification.

With a file dialog, the user can select a file to work with or specify a file name for saving current work.

Portability — IVBFileDialog

Features not defined for Windows:

- saveAsEAType

Related Parts — IVBFileDialog

Dialog parts:

- "IMessageBox" on page 221
- "IVBFileDialog"
- "IVBFontDialog" on page 381

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IVBFileDialog

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining the Title
- Setting the Initial Choice
- Filtering the List
- Defining the OK Button
- Assigning Ownership

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Opening the Dialog
- Starting the Requested Action
- Starting a Member Function
- Starting Custom Logic

Adding the Part

To add an IVBFileDialog* part to a composite part, select an IVBFileDialog* part from the parts palette. Then, drop the part on the free-form surface.

Defining the Title

To define a title for the dialog, enter the title in the **Title** field on the IVBFileDialog* part's **General** settings page.

Setting the Initial Choice

To provide an initial file name for the dialog, enter the name in the **File name** field on the IVBFileDialog* part's **General** settings page.

Filtering the List

To list only files of a particular extended attribute type, specify the type in the **File (EA) type** field on the IVBFileDialog* part's **General** settings page. Leave this field empty to list all files.

Defining the OK Button

To specify text, such as **Open** or **Save as**, for the **OK** push button, enter the text in the **OK button text** field on the IVBFileDialog* part's **General** settings page.

Assigning Ownership

If you have only one frame window in your view, the dialog is automatically owned by that frame window.

To provide the file dialog for a second frame window in your view, reference the IFrameWindow* part's **Subpart name**, which appears on its **General** settings page. Enter the frame window subpart name, prefixed with "i," in the **Owner** field on the IVBFileDialog* part's **General** settings page.

Opening the Dialog

To show the dialog when the user selects a menu choice, connect the IMenuItem* part's *commandEvent* feature to the IVBFileDialog* part's *showModally* action.

Starting the Requested Action

To start a part's file action, connect to the action and provide the file name. For example, to load an IMultiLineEdit* part from a file, connect the IVBFileDialog* part's *fileName* attribute to the IMultiLineEdit* part's *importFromFile* action. The file dialog provides the file name when the user clicks the OK button.

Starting a Member Function

To call a member function for the file operation, do the following:

1. Connect the IVBFileDialog* part's *pressedOkEvent* feature to the free-form surface.
2. Select the **More** choice on the connection menu.
3. Enter the signature of a member function to perform the operation.
4. Connect the IVBFileDialog* part's *fileName* attribute to a file name parameter on the connection to the member function.

Starting Custom Logic

To run custom logic for the file operation, do the following:

1. Connect the IVBFileDialog* part's *pressedOkEvent* feature to the free-form surface.
2. Select **Custom logic** on the connection menu.
3. Enter C++ code to perform the operation. To access the file name in your custom logic, select **Source**. Enter the get member function of the *fileName* attribute after the source pointer inserted into your code. This is the resulting code segment:

```
source->fileName()
```

Preferred Features — IVBFileDialog

fileName	Returns the fully qualified file name selected by the user.
pressedOkEvent	Notification identifier provided to observers when the showing `IFileDialog` has been dismissed with ok button.
returnValue	Returns the return code, if an error occurred.
setSaveAsDialog	Creates a Save As dialog.
setTitle	Sets the dialog's title.
showModally	Show modally the file dialog.
this	A Pointer to the instance.

Features — IVBFileDialog

Action	Attribute	Event
addDrive ■	asDebugInfo	anyEvent
addFileType ■	asString	**createdEvent**
disableNotification	**buttonPressedId**	**dismissedEvent**
notifyObservers ■	enabledForNotification ■	***fileName***
operator = ■	**fileDialog**	**pressedOkEvent**
setDialogTemplate ■	**fileDialogSettings**	
setInitialDrive ■	**fileName** ■ ►	
setInitialFileType ■	**modeless**	
setOKButtonText ■	**pressedOK**	
setOpenDialog	**returnValue**	
setPosition ■	**selectedFileCount**	
setSaveAsDialog	**this**	
setTitle ■		
show ■		
showModally ■		

Chapter 80. IVBFlyText

Description: IBM VB fly-over text control
Part type: Nonvisual
Part file: vbbase
Header file: ivbfly.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IFlyText
IVBFlyText

Use an IVBFlyText* part to show fly-over text for controls in your frame window. *Fly-over text* appears when the user places the mouse pointer over the control. The text is typically shown in a small rectangle near the control.

The user can read help information in the fly-over text without opening another window for help.

Portability — IVBFlyText

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor

© Copyright IBM Corp. 1995, 1996

- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IVBFlyText

Help parts:

- "IHelpWindow" on page 189
- "IVBFlyText" on page 377
- "IVBInfoArea" on page 385

Building Guidelines — IVBFlyText

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Fly-Over Help Text
- Defining Fly-Over Information Text
- Assigning Ownership

Adding the Part

To add an IVBFlyText* part to a composite part, select an IVBFlyText* part from the parts palette. Then, drop the part on the free-form surface. Add one IVBFlyText* part for each frame window in your composite part.

Defining Fly-Over Help Text

To define short fly-over help text for frame window and control parts, enter the text in the **Fly-over short text** field on each part's **Control** settings page.

The text appears in a fly-over caption when the mouse pointer is over the part.

Defining Fly-Over Information Text

To define lengthier fly-over information for frame window and control parts, enter the text in the **Fly-over long text** field on each part's **Control** settings page.

To display the long fly-over text, you must identify the text control you want to display the text in. For example, to display the long fly-over text in an IVBInfoArea* part, connect the IVBInfoArea* part's *this* attribute to the IVBFlyText* part's *longTextControl* attribute.

Assigning Ownership

If you have only one frame window in your view, the fly-over text is automatically owned by that frame window.

To provide the fly-over text for a second frame window in your view, reference the IFrameWindow* part's **Subpart name**, which appears on its **General** settings page. Enter the frame window subpart name, prefixed with "i," in the **Owner** field on the **General** settings page of a second IVBFlyText* part.

Preferred Features — IVBFlyText

disableFlyOverHelp
Removes flyover help.

flyOverHelp If the `IFlyOverHelpHandler` is enabled, `true` is returned.

flyOverHelpHandler
Returns the `IFlyOverHelpHandler`.

longTextControl
Text control used to display the `long` version of the flyover text

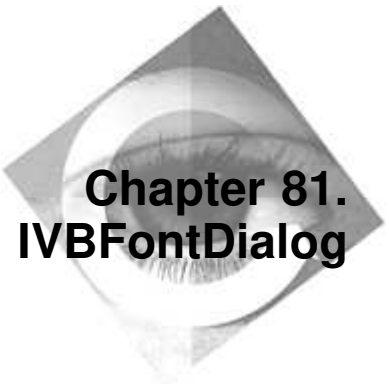
this A Pointer to the instance.

Features — IVBFlyText

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableFlyOverHelp	autoDestroyWindow ■	commandEvent
disableGroup	backgroundColor ■ ►	*disabledBackgroundColor*
disableNotification	bidiSupported	*disabledForegroundColor*
disableTabStop	borderColor ■ ►	*enabled*
disableUpdate	characterSize	*focus*
dispatchRemainingHandlers ■	clipboardHasTextFormat	*font*
enable ■	defaultPushButton	*foregroundColor*
enableUpdate ■	disabledBackgroundColor ■ ►	gotFocusEvent ■
handleException ■	disabledForegroundColor ■ ►	*hiliteBackgroundColor*
hide	displaySize	*hiliteForegroundColor*
hideSourceEmphasis	enabled ■ ►	*inactiveColor*
isLayoutDistorted ■	enabledForNotification ■	inputDisabledEvent ■
matchForMnemonic ■	**flyOverHelp** ■	inputEnabledEvent ■
notifyObservers ■	**flyOverHelpHandler**	*isDigits*
positionBehindSibling ■	focus ►	lostFocusEvent ■
positionBehindSiblings	font ■ ►	*position*
positionOnSiblings	foregroundColor ■ ►	*shadowColor*
postEvent ■	frameWindow	*size*

IVBFlyText

Action	Attribute	Event
refresh ■	group ■	systemCommandEvent
releasePointer	handle	*text*
releasePresSpace ■	helpId ■	*textLength*
resetActiveColor	hiliteBackgroundColor ■ ►	*valueAsDouble*
resetBackgroundColor	hiliteForegroundColor ■ ►	*valueAsInt*
resetBorderColor	id ■	*valueAsUnsigned*
resetDisabledBackgroundColor	inactiveColor ■ ►	visibilityDisabledEvent ■
resetDisabledForegroundColor	isDigits ►	visibilityEnabledEvent ■
resetFont	itemProvider ■	*visible*
resetForegroundColor	layoutAdjustment	
resetHiliteBackgroundColor	**longTextControl** ■	
resetHiliteForegroundColor	menu	
resetInactiveColor	minimumSize ■	
resetMinimumSize	nativeRect	
resetShadowColor	owner ■	
sendEvent ■	parent ■	
setFocus	parentSize	
setLayoutDistorted ■	pointerCaptured	
show ■	position ■ ►	
showSourceEmphasis ■	presSpace	
	rect ■	
	relativeWindowRect ■	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	



Chapter 81. IVBFontDialog

Description: IBM VB font dialog window
Part type: Nonvisual
Part file: vbbase
Header file: ivbfontd.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBFontDialog

Use an IVBFontDialog* part to represent a font dialog during application building. A font dialog provides selection fields and other controls for font specification.

With a font dialog, the user can select a font name, size, and styles to use with your application.

Portability — IVBFontDialog

Features not defined for Windows:

- emHeight
- externalLeading
- fontFamily
- fontWeight
- fontWidth
- nominalPointSize
- xHeight

Related Parts — IVBFontDialog

Dialog parts:

- "IMessageBox" on page 221
- "IVBFileDialog" on page 373
- "IVBFontDialog"

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IVBFontDialog

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining the Title
- Filtering the List
- Defining Preview Text
- Assigning Ownership

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Opening the Dialog
- Starting a Member Function
- Starting Custom Logic

Adding the Part

To add an IVBFontDialog* part to a composite part, select an IVBFontDialog* part from the parts palette. Then, drop the part on the free-form surface.

Defining the Title

To define a title for the dialog, enter the title in the **Title** field on the IVBFontDialog* part's **General** settings page.

Filtering the List

To list only fonts in one font family, such as Helvetica or Courier, specify the font family in the **Font family name** field on the IVBFontDialog* part's **General** settings page. Leave this field empty to list all available fonts.

Defining Preview Text

To provide preview text to show font appearance in the dialog, enter the text in the **Preview text** field on the IVBFontDialog* part's **General** settings page.

Assigning Ownership

If you have only one frame window in your view, the dialog is automatically owned by that frame window.

To provide the font dialog for a second frame window in your view, reference the IFrameWindow* part's **Subpart name**, which appears on its **General** settings page. Enter the frame window subpart name, prefixed with "i," in the **Owner** field on the IVBFontDialog* part's **General** settings page.

Opening the Dialog

To show the dialog when the user selects a menu choice, connect the IMenuItem* part's *commandEvent* feature to the IVBFontDialog* part's *setFontFromWindowModally* action. To identify the subject part that is the font dialog is opened for, connect the subject part's *this* attribute to the target parameter of the *commandEvent*-to-*setFontFromWindowModally* connection.

When the font dialog is opened, it shows the current font for the subject part. When the user clicks the OK button, the dialog applies the selected font to the subject part.

Starting a Member Function

To call a member function for the font operation, do the following:

1. Connect the IVBFontDialog* part's *pressedOkEvent* feature to the free-form surface
2. Select **More** on the connection menu.
3. Enter the signature of a member function to perform the operation.
4. Connect the IVBFontDialog* part's *font* attribute to a font parameter on the connection to the member function.

Starting Custom Logic

To run custom logic for the font operation, do the following:

1. Connect the IVBFontDialog* part's *pressedOkEvent* feature to the free-form surface
2. Select **Custom logic** on the connection menu.
3. Enter C++ code to perform the operation. To access the font selection in your custom logic, select **Source**. Enter the get member function of the *font* attribute after the source pointer inserted into your code. This is the resulting code segment:

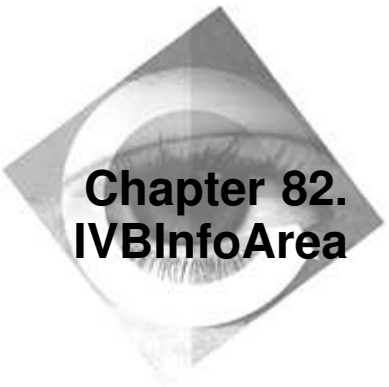
```
source->font()
```

Preferred Features — IVBFontDialog

font	Returns the font attribute.
pressedOkEvent	Notification identifier provided to observers when the showing `IFontDialog` has been dismissed with ok button.
returnValue	Returns the return code, if an error occurred.
setTitle	Sets the font dialog's title.
showModally	Results in an application-modal `IFontDialog` being displayed.
this	A Pointer to the instance.

Features — IVBFontDialog

Action	Attribute	Event
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	**createdEvent**
operator = ■	**buttonPressedId**	**dismissedEvent**
setDialogTemplate ■	enabledForNotification ■	***font***
setDisplayPS ■	**font** ■ ►	**pressedOkEvent**
setFamily ■	**fontDialog**	
setFontFromFontModally ■	**fontDialogSettings**	
setFontFromWindowModally ■	**modeless**	
setPosition ■	**pointSize**	
setPreviewText ■	**pressedOK**	
setPrinterPS ■	**returnValue**	
setSizeList ■	**this**	
setTitle ■		
show ■		
showFromFont ■		
showModally ■		



Chapter 82. IVBInfoArea

Description: IBM Visual Builder frame extension to show user prompts
Part type: Visual
Part file: vbbase
Header file: ivbinfoa.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ITextControl
IStaticText
IInfoArea
IVBInfoArea

Use an IVBInfoArea* part to briefly describe menu choices or to report the successful completion of a selected action. The *information area* is a frame extension at the bottom of a frame window.

The user can read help information in the information area without opening another window for help.

Portability — IVBInfoArea

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor
- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor

© Copyright IBM Corp. 1995, 1996

- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IVBInfoArea

Help parts:

- "IHelpWindow" on page 189
- "IVBFlyText" on page 377
- "IVBInfoArea" on page 385

Frame parts:

- "IFrameWindow" on page 167
- "IMenu" on page 205
- "IScrollBar" on page 287
- "ITitle" on page 337
- "IToolBar" on page 343
- "IVBInfoArea" on page 385

Building Guidelines — IVBInfoArea

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Defining Menu Information Text
- Defining Fly-Over Information Text

Adding the Part

To add an IVBInfoArea* part to a composite part, select an IVBInfoArea* part from the parts palette. Then, drop the part on an IFrameWindow* part's title bar. The information area is attached as a frame extension at the bottom of the frame window, beneath the client area.

Defining Menu Information Text

To define information area text for menus, do the following:

- The information area displays text describing menu items. To define text to display when no menu choice is selected, enter text in the **Inactive text** field on the IVBInfoArea* part's **General** settings page.
- To define text to display when the user selects a disabled menu item, enter text in the **Disabled text** field on the IVBInfoArea* part's **General** settings page.
- To define text to display when the user selects a menu item that has no information text defined, enter text in the **Missing text** field on the IVBInfoArea* part's **General** settings page.
- To define text to display a menu item when the user the item, enter text in the **Info area text** field on the IMenuItem* part's General settings page.

Defining Fly-Over Information Text

To display long fly-over help in the information area for controls in a frame window, do the following:

- Enter long fly-over help text in the **Fly-over long text** field on each part's Control settings page.
- Add an IVBFlyText* part for the frame window if you do not already have one.
- Connect the IVBInfoArea* part's *this* attribute to the IVBFlyText* part's *longTextControl* attribute.

Preferred Features — IVBInfoArea

inactiveText	Returns the text displayed when the menu is inactive.
limit	Returns the number of characters set by setLimit.
menu	Popup menu link.
missingText	Returns the text that displays when the ICLUI cannot obtain the required information string from the resource library.
setFocus	Sets the input focus to the window.
text	Returns the control window's text.
textLength	Returns the current length of the control window's text in bytes.
this	A Pointer to the instance.

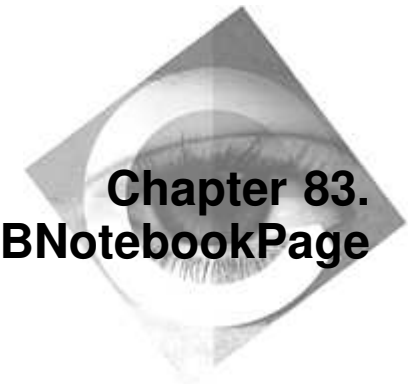
Features — IVBInfoArea

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	alignment ■	anyEvent
convertToGUIStyle ■	asDebugInfo	*backgroundColor*
disable	asString	*borderColor*
disableFillBackground	autoDeleteObject ■	commandEvent
disableGroup	autoDestroyWindow ■	*disabledBackgroundColor*
disableHalftone	backgroundColor ■ ►	*disabledForegroundColor*
disableNotification	bidiSupported	*disabledText*
disableStrikeout	borderColor ■ ►	*enabled*
disableTabStop	characterSize	*fillBackground*
disableUnderscore	clipboardHasTextFormat	*fillColor*
disableUpdate	defaultPushButton	*focus*
dispatchRemainingHandlers ■	disabledBackgroundColor ■ ►	*font*
enable ■	disabledForegroundColor ■ ►	*foregroundColor*
enableUpdate ■	disabledText ■ ►	gotFocusEvent ■
handleException ■	displaySize	*halftone*
hide	enabled ■ ►	*hiliteBackgroundColor*
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	fillBackground ■ ►	*inactiveColor*
matchForMnemonic ■	fillColor ■ ►	*inactiveText*
notifyObservers ■	focus ►	inputDisabledEvent ■
positionBehindSibling ■	font ■ ►	inputEnabledEvent ■
positionBehindSiblings	foregroundColor ■ ►	*isDigits*
positionOnSiblings	frameWindow	*limit*
postEvent ■	group ■	lostFocusEvent ■
refresh ■	halftone ■ ►	*missingText*
releasePointer	handle	*position*
releasePresSpace ■	helpId ■	*resourceLibrary*
removeHelpText ■	hiliteBackgroundColor ■ ►	*shadowColor*
resetActiveColor	hiliteForegroundColor ■ ►	*size*
resetBackgroundColor	id ■	*strikeout*
resetBorderColor	inactiveColor ■ ►	systemCommandEvent
resetDisabledBackgroundColor	inactiveText ■ ►	*text*
resetDisabledForegroundColor	isDigits ►	*textLength*
resetFillColor	itemProvider ■	*underscore*
resetFont	layoutAdjustment	*valueAsDouble*
resetForegroundColor	limit ■ ►	*valueAsInt*
resetHiliteBackgroundColor	lineCount ■	*valueAsUnsigned*
resetHiliteForegroundColor	menu	visibilityDisabledEvent ■
resetInactiveColor	minimumSize ■	visibilityEnabledEvent ■
resetMinimumSize	missingText ■ ►	*visible*
resetShadowColor	nativeRect	
sendEvent ■	owner ■	
setFocus	parent ■	

IVBInfoArea

Action	Attribute	Event
setHelpText ■	parentSize	
setLayoutDistorted ■	pointerCaptured	
show ■	position ■ ►	
showSourceEmphasis ■	presSpace	
start ■	rect ■	
stop ■	resourceLibrary ■ ►	
	shadowColor ■ ►	
	showing	
	size ■ ►	
	strikeout ■ ►	
	stringTableOffset ■	
	tabStop ■	
	text ■ ►	
	textLength ►	
	this	
	underscore ■ ►	
	valid	
	valueAsDouble ■ ►	
	valueAsInt ■ ►	
	valueAsUnsigned ■ ►	
	visible ■ ►	
	visibleRectangle	

IVBInfoArea



Chapter 83. IVBNotebookPage

Description: IBM VB Notebook page
Part type: Nonvisual
Part file: vbbase
Header file: None

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBNotebookPage

Use an IVBNotebookPage* part for settings choices or for related information. A notebook page presents information that is generally related. The page can have a tab for direct user access.

The user can do the following with a notebook page:

- View or change information by using other controls
- Turn the page by clicking on page buttons

Related Parts — IVBNotebookPage

Notebook parts:

- "INotebook" on page 237
- "IVBNotebookPage"

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IVBNotebookPage

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting the Tab Type
- Defining Tab Text
- Defining Status Text
- Adding a Canvas
- Sizing the Canvas

Adding the Part

To add an IVBNotebookPage* part to an INotebook* part, select an add page choice on the contextual menu of either the INotebook* part or an IVBNotebookPage* part.

- To add a page to an empty notebook, select **Add initial page** on the INotebook* part's contextual menu.
- To add a page before or after the top page of a notebook, select **Add page** on the INotebook* part's contextual menu. Then, select **Before top page** or **After top page** on the submenu.
- To add a page before or after the currently selected page, select **Add page** on the IVBNotebookPage* part's contextual menu. Then, select **Before** or **After** on the submenu.

An IVBNotebookPage* part is placed in the notebook.

Selecting the Tab Type

To select a tab type for the page, set the **minorTab** and **majorTab** fields on the IVBNotebookPage* part's **Styles** settings page. By default, the page has a major tab.

For a minor tab, set the **minorTab** field to **On** and set the **majorTab** field to **Off**.

For no tabs, set both fields to **Off**.

Do not set both fields to **On**.

Defining Tab Text

To define text for a tab, enter the text you want in the **Tab text** field on the IVBNotebookPage* part's **General** settings page.

Defining Status Text

To define text for the status area at the bottom of the page, enter the text you want in the **Status text** field on the IVBNotebookPage* part's **General** settings page.

This setting is not applicable for the native Windows notebook. The notebook does not have a status area.

Adding a Canvas

A notebook page accepts only one part, so you should place one of the canvases or a view port on the page. If you use a view port, place a canvas in the view port. After you have a canvas on the notebook page, you can place controls on the canvas.

To add a canvas part. select it from the parts palette and drop it on the IVBNotebookPage* part.

Sizing the Canvas

To resize the canvas, select the canvas part. Then, drag the corner handle to expand the canvas. This allows you to work with control parts you place on the canvas.

Leave part of the IVBNotebookPage* part exposed so you can still access the part. If the **autoPageSize** field on the **Styles** settings page is set to **On**, the canvas fills the page in generated code. The **autoPageSize** field is set to **On** by default.

Preferred Features — IVBNotebookPage

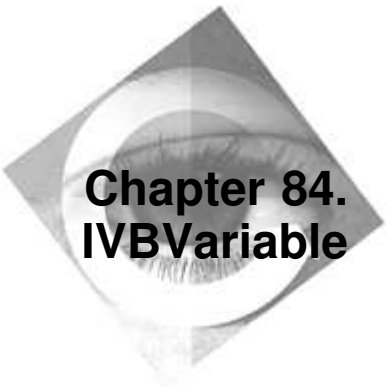
notebook
pageHandle
statusText
tabBitmap
tabText
this
window

Features — IVBNotebookPage

Action	Attribute	Event
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	
operator = ■	enabledForNotification ■	
	notebook	
	pageHandle	
	statusText ■	
	tabBitmap ■	

IVBNotebookPage

Action	Attribute	Event
	tabText ∎	
	this	
	window ∎	



Chapter 84. IVBVariable

Description: IBM VB placeholder for part instance
Part type: Nonvisual
Part file: vbbase
Header file: None

Derivation:
IVBVariable

Use an IVBVariable* part to pass data between composite parts or for holding objects. IVBVariable* parts are commonly used to hold objects such as the following:

- New objects from IVBFactory* parts
- Attributes of parts
- Selected collection elements from list boxes and containers

Related Parts — IVBVariable

Model parts:

- "IVBFactory" on page 371
- "IVBVariable"
- "IVSequence" on page 403

Building Guidelines — IVBVariable

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Object Type
- Changing the Name
- Promoting Features to the Interface

© Copyright IBM Corp. 1995, 1996

IVBVariable

Adding the Part

To add an IVBVariable* part to a composite part, select an IVBVariable* part from the parts palette. Then, drop the part on the free-form surface.

You can also add an IVBVariable* part for a particular part type using the add part window. Type the part type that you want the variable to hold. Then, add the part as a Variable.

Setting the Object Type

To specify the object type that the variable can hold, open the IVBVariable* part's contextual menu. Then, select **Change type** and enter the object type you want.

The IVBVariable* part is implemented in your application as follows:

- A part type, such as IEntryField, is implemented by an IVBVariablePart* part.
- A part pointer type, such as IEntryField*, is implemented by an IVBVariablePartPointer* part.
- A class type, such as IDate, is implemented by an IVBVariableClass* part.
- A class pointer type, such as IDate*, is implemented by an IVBVariableClassPointer* part.

After you change the variable type, you can connect to part interface features of the new type.

Changing the Name

To change the variable name to something meaningful for your application, open the IVBVariable* part's contextual menu. Then, select **Change name** and enter the name you want.

Promoting Features to the Interface

To promote variable features to the interface of your composite part, open the IVBVariable* part's contextual menu. Then, select **Promote part feature** and select or enter the features you want to promote.

Each promoted feature is given a name based on the variable name and the variable feature name. For example, if you promote the *text* attribute from an IEntryField* variable named Selection, the promoted feature is named *selectionText*.

Give the IVBVariable* part a meaningful name before you promote features. This ensures that feature names are based on a meaningful variable name.

Preferred Features — IVBVariable

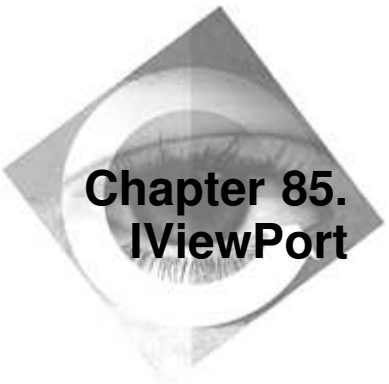
enabledForNotification
Returns `true` if an object is sending notifications to its observers.

this Current object instance attribute.

Features — IVBVariable

Action	Attribute	Event
deleteTarget	asDebugInfo	anyEvent
disableNotification	asString	***targetPtr***
notifyObservers ■	autoDeleteTarget ■	***this***
operator = ■	enabledForNotification ■	
	targetPtr ►	
	this ►	

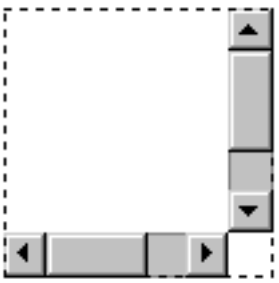
IVBVariable



Chapter 85. IViewPort

Description: IBM view port to scroll nonresizable windows
Part type: Visual
Part file: vbbase
Header file: ivport.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
IViewPort



Use an IViewPort* part to provide a scrollable view.

You can use an ISetCanvas* part in another composer part, such as the following:

- IFrameWindow* client area
- IMultiCellCanvas*
- ISplitCanvas*
- IVBNotebookPage*

You can place one part in the view port, typically one of the canvas parts.

Portability — IViewPort

Features not defined for Windows:

- messageQueue

Features not active on Windows:

- activeColor
- borderColor
- disabledBackgroundColor

© Copyright IBM Corp. 1995, 1996

- disabledForegroundColor
- hiliteBackgroundColor
- hiliteForegroundColor
- inactiveColor
- shadowColor
- resetActiveColor
- resetBorderColor
- resetDisabledBackgroundColor
- resetDisabledForegroundColor
- resetHiliteBackgroundColor
- resetHiliteForegroundColor
- resetInactiveColor
- resetShadowColor

Related Parts — IViewPort

Client parts:

- "ICanvas" on page 83
- "IDrawingCanvas" on page 145
- "IMultiCellCanvas" on page 225
- "IMultiLineEdit" on page 231
- "INotebook" on page 237
- "ISetCanvas" on page 293
- "ISplitCanvas" on page 307
- "IVBContainerControl" on page 359
- "IViewPort" on page 399

Building Guidelines — IViewPort

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding a Canvas
- Sizing the Canvas

Adding the Part

To add an IViewPort* part to a composite part, select an IViewPort* part from the parts palette. Then, drop the part on one of the following targets:

- An IFrameWindow* part
- A canvas part
- An IVBNotebookPage* part

Adding a Canvas

A view port accepts only one part, so place one of the canvases in the view port. You can place controls on the canvas.

To add a canvas part. select it from the parts palette and drop it on the IViewPort* part.

If you drop an IMultiCellCanvas* part on the view port, set the **expandableViewWindow** setting on the IViewPort* part's **Styles** settings page. This enables the IViewPort* part to automatically add or remove scroll bars based on the minimum size of cells in the IMultiCellCanvas* part.

Sizing the Canvas

To resize the canvas, select the canvas part. Then, drag the corner handle to expand the canvas. This allows you to work with control parts you place on the canvas.

Preferred Features — IViewPort

menu Popup menu link.

setFocus Sets the input focus to the window.

this A Pointer to the instance.

viewWindow Returns the handle of the window bounded by the view port.

viewWindowDrawRectangle
Returns the area of the view window that is currently visible in the view port.

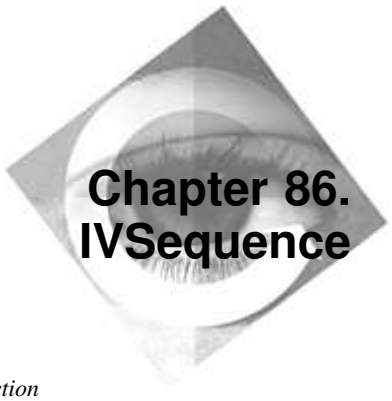
viewWindowSize
Returns the logical size of the window being scrolled.

Features — IViewPort

Action	Attribute	Event
applyBidiSettings ■	activeColor ■ ►	*activeColor*
capturePointer ■	asDebugInfo	anyEvent
convertToGUIStyle ■	asString	*backgroundColor*
disable	autoDeleteObject ■	*borderColor*
disableGroup	autoDestroyWindow ■	commandEvent
disableNotification	backgroundColor ■ ►	*disabledBackgroundColor*
disableTabStop	bidiSupported	*disabledForegroundColor*
disableUpdate	borderColor ■ ►	*enabled*
dispatchRemainingHandlers ■	characterSize	*focus*
enable ■	defaultPushButton	*font*
enableUpdate ■	disabledBackgroundColor ■ ►	*foregroundColor*
handleException ■	disabledForegroundColor ■ ►	gotFocusEvent ■
hide	enabled ■ ►	*hiliteBackgroundColor*

IViewPort

Action	Attribute	Event
hideSourceEmphasis	enabledForNotification ■	*hiliteForegroundColor*
isLayoutDistorted ■	focus ►	*inactiveColor*
matchForMnemonic ■	font ■ ►	inputDisabledEvent ■
notifyObservers ■	foregroundColor ■ ►	inputEnabledEvent ■
positionBehindSibling ■	frameWindow	lostFocusEvent ■
positionBehindSiblings	group ■	*position*
positionOnSiblings	handle	*shadowColor*
postEvent ■	helpId ■	*size*
refresh ■	hiliteBackgroundColor ■ ►	systemCommandEvent
releasePointer	hiliteForegroundColor ■ ►	***viewWindowSize***
releasePresSpace ■	**horizontalScrollBar**	visibilityDisabledEvent ■
resetActiveColor	id ■	visibilityEnabledEvent ■
resetBackgroundColor	inactiveColor ■ ►	*visible*
resetBorderColor	itemProvider ■	
resetDisabledBackgroundColor	layoutAdjustment	
resetDisabledForegroundColor	menu	
resetFont	minimumSize ■	
resetForegroundColor	nativeRect	
resetHiliteBackgroundColor	origDefaultButtonHandle	
resetHiliteForegroundColor	owner ■	
resetInactiveColor	parent ■	
resetMinimumSize	parentSize	
resetShadowColor	pointerCaptured	
scrollViewHorizontallyTo ■	position ■ ►	
scrollViewVerticallyTo ■	presSpace	
sendEvent ■	rect ■	
setFocus	shadowColor ■ ►	
setLayoutDistorted ■	showing	
show ■	size ■ ►	
showSourceEmphasis ■	tabStop ■	
	this	
	valid	
	verticalScrollBar	
	viewWindow	
	viewWindowDrawRectangle	
	viewWindowSize ■ ►	
	visible ■ ►	
	visibleRectangle	



Chapter 86. IVSequence

Description: IBM VSequence template collection class
Part type: Nonvisual
Part file: vbbase
Header file: ivseq.h

Derivation:
IACollection
IAOrderedCollection
IASequentialCollection
IASequence
IRSequence
IVSequenceOnBase
IVGSequence
IVSequence

Use an IVSequence* part to manage objects in an ordered collection.

The user interacts with objects in the collection through other parts, such as list boxes and combination boxes. The user does not interact directly with a sequence.

Related Parts — IVSequence

Collection parts:

- "ICollectionViewComboBox" on page 105
- "ICollectionViewListBox" on page 113
- "IVBContainerControl" on page 359
- "IVSequence"

Model parts:

- "IVBFactory" on page 371
- "IVBVariable" on page 395
- "IVSequence"

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IVSequence

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Object Type
- Setting Initial Contents

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Adding an Object
- Removing an Object

Adding the Part

To add an IVSequence* part to a composite part, select an IVSequence* part from the parts palette. Then, drop the part on the free-form surface.

Setting the Object Type

To define the type of objects in the sequence, enter a pointer to the part type (part type and *) in the **Item type** field on the IVSequence* part's **General** settings page.

Setting Initial Contents

To specify initial objects in the collection, enter the choices in the **Contents** field on the IVSequence* part's **General** settings page. Type code that creates pointers to objects in the sequence. For example, if the sequence holds ICustomer* objects, type something like the following:

```
new ICustomer("George")
new ICustomer("Dave")
```

Adding an Object

To add a given object to the collection when a push button is pressed, connect the IPushButton* part's *buttonClickEvent* feature to the IVSequence* part's *add* action. Then, connect the *this* attribute of the object to the *buttonClickEvent*-to-*add* connection, providing the element parameter for the *add* action.

Removing an Object

You might need to remove an object from the collection based on its index. For example, the user might click on a push button to remove the selected object item from an ICollectionViewComboBox* part. An ICollectionViewComboBox* part's *selection* attribute provides the index.

To remove an object with a known index from the collection when a push button is pressed, connect the IPushButton* part's *buttonClickEvent* feature to the IVSequence* part's *removeAtPosition* action. Then, connect the index to the *buttonClickEvent*-to-*removeAtPosition* connection, providing the position parameter for the *removeAtPosition* action.

Preferred Features — IVSequence

add Returns `True` if the element was added.

addAsFirst Adds the element to the collection as the first element in sequential order.

addAsLast Adds the element to the collection as the last element in sequential order.

addAtPosition Adds the element at the given position to the collection.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

Features — IVSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	
copy ■	lastElement	
disableNotification	maxNumberOfElements	
elementAtPosition ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		

IVSequence

Part 3. Sample Parts in vbcc.vbb

The following chapters provide information on sample collection parts that you can modify to create your own parts. These parts are provided without service support.

To use these parts for the first time, import part information from the vbw3cc.vbe part information file in the ibmcppw\samples\visbuild\vbsample directory. The vbcc.vbb part file is created in your ibmcppw\working directory.

To use these parts after the first time, load the vbcc.vbb part file from your ibmcppw\working directory.

Chapter 87. IElemPointer 411
Preferred Features — IElemPointer 411
Features — IElemPointer 411

Chapter 88. IMngPointer 413
Preferred Features — IMngPointer 413
Features — IMngPointer 413

Chapter 89. IVBag 415
Preferred Features — IVBag 415
Features — IVBag 415

Chapter 90. IVBagOnBSTKeySortedSet 417
Preferred Features — IVBagOnBSTKeySortedSet 417
Features — IVBagOnBSTKeySortedSet 417

Chapter 91. IVBagOnHashKeySet 419
Preferred Features — IVBagOnHashKeySet 419
Features — IVBagOnHashKeySet 419

Chapter 92. IVBagOnSortedDilutedSequence 421
Preferred Features — IVBagOnSortedDilutedSequence 421
Features — IVBagOnSortedDilutedSequence 421

Chapter 93. IVBagOnSortedLinkedSequence 423
Preferred Features — IVBagOnSortedLinkedSequence 423
Features — IVBagOnSortedLinkedSequence 423

Chapter 94. IVBagOnSortedTabularSequence 425

© Copyright IBM Corp. 1995, 1996

Sample Parts in vbcc.vbb

Preferred Features — IVBagOnSortedTabularSequence 425
Features — IVBagOnSortedTabularSequence 425

Chapter 95. IVDeque 427
Preferred Features — IVDeque 427
Features — IVDeque 427

Chapter 96. IVDequeOnDilutedSequence 429
Preferred Features — IVDequeOnDilutedSequence 429
Features — IVDequeOnDilutedSequence 429

Chapter 97. IVDilutedSequence 431
Preferred Features — IVDilutedSequence 431
Features — IVDilutedSequence 431

Chapter 98. IVEqualitySequence 433
Preferred Features — IVEqualitySequence 433
Features — IVEqualitySequence 434

Chapter 99. IVEqualitySequenceOnDilutedSequence 435
Preferred Features — IVEqualitySequenceOnDilutedSequence 435
Features — IVEqualitySequenceOnDilutedSequence 436

Chapter 100. IVEqualitySequenceOnTabularSequence 437
Preferred Features — IVEqualitySequenceOnTabularSequence 437
Features — IVEqualitySequenceOnTabularSequence 438

Chapter 101. IVHeap 439
Preferred Features — IVHeap 439
Features — IVHeap 439

Chapter 102. IVHeapOnDilutedSequence 441
Preferred Features — IVHeapOnDilutedSequence 441
Features — IVHeapOnDilutedSequence 441

Chapter 103. IVLinkedSequence 443
Preferred Features — IVLinkedSequence 443
Features — IVLinkedSequence 443

Chapter 104. IVQueue 445
Preferred Features — IVQueue 445
Features — IVQueue 445

Sample Parts in vbcc.vbb

Chapter 105. IVQueueOnDilutedSequence 447
Preferred Features — IVQueueOnDilutedSequence 447
Features — IVQueueOnDilutedSequence 447

Chapter 106. IVQueueOnTabularSequence 449
Preferred Features — IVQueueOnTabularSequence 449
Features — IVQueueOnTabularSequence 449

Chapter 107. IVSet 451
Preferred Features — IVSet 451
Features — IVSet 451

Chapter 108. IVSetOnBSTKeySortedSet 453
Preferred Features — IVSetOnBSTKeySortedSet 453
Features — IVSetOnBSTKeySortedSet 453

Chapter 109. IVSetOnHashKeySet 455
Preferred Features — IVSetOnHashKeySet 455
Features — IVSetOnHashKeySet 455

Chapter 110. IVSetOnSortedDilutedSequence 457
Preferred Features — IVSetOnSortedDilutedSequence 457
Features — IVSetOnSortedDilutedSequence 457

Chapter 111. IVSetOnSortedLinkedSequence 459
Preferred Features — IVSetOnSortedLinkedSequence 459
Features — IVSetOnSortedLinkedSequence 459

Chapter 112. IVSetOnSortedTabularSequence 461
Preferred Features — IVSetOnSortedTabularSequence 461
Features — IVSetOnSortedTabularSequence 461

Chapter 113. IVSortedBag 463
Preferred Features — IVSortedBag 463
Features — IVSortedBag 463

Chapter 114. IVSortedBagOnSortedDilutedSequence 465
Preferred Features — IVSortedBagOnSortedDilutedSequence 465
Features — IVSortedBagOnSortedDilutedSequence 465

Chapter 115. IVSortedBagOnSortedLinkedSequence 467
Preferred Features — IVSortedBagOnSortedLinkedSequence 467
Features — IVSortedBagOnSortedLinkedSequence 467

Sample Parts in vbcc.vbb

Chapter 116. IVSortedBagOnSortedTabularSequence 469
Preferred Features — IVSortedBagOnSortedTabularSequence 469
Features — IVSortedBagOnSortedTabularSequence 469

Chapter 117. IVSortedSet 471
Preferred Features — IVSortedSet 471
Features — IVSortedSet 471

Chapter 118. IVSortedSetOnBSTKeySortedSet 473
Preferred Features — IVSortedSetOnBSTKeySortedSet 473
Features — IVSortedSetOnBSTKeySortedSet 473

Chapter 119. IVSortedSetOnSortedLinkedSequence 475
Preferred Features — IVSortedSetOnSortedLinkedSequence 475
Features — IVSortedSetOnSortedLinkedSequence 475

Chapter 120. IVSortedSetOnSortedTabularSequence 477
Preferred Features — IVSortedSetOnSortedTabularSequence 477
Features — IVSortedSetOnSortedTabularSequence 477

Chapter 121. IVStack 479
Preferred Features — IVStack 479
Features — IVStack 479

Chapter 122. IVStackOnTabularSequence 481
Preferred Features — IVStackOnTabularSequence 481
Features — IVStackOnTabularSequence 481

Chapter 123. IVTabularSequence 483
Preferred Features — IVTabularSequence 483
Features — IVTabularSequence 483

Chapter 87. IElemPointer

Description: IBM pointer template collection class
Part type: Class
Part file: vbcc
Header file: iptr.h

Derivation:
IElemPointer

Preferred Features — IElemPointer

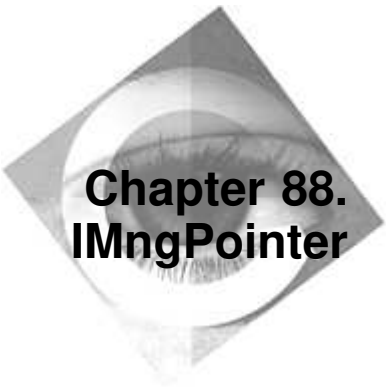
this A Pointer to the instance.

Features — IElemPointer

Action	Attribute	Type
	this	`IElemPointer*`

© Copyright IBM Corp. 1995, 1996

IElemPointer



Chapter 88. IMngPointer

Description: IBM pointer template collection class
Part type: Class
Part file: vbcc
Header file: iptr.h

Derivation:
IMngPointer

Preferred Features — IMngPointer

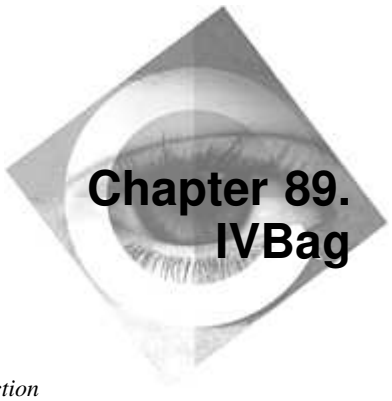
this A Pointer to the instance.

Features — IMngPointer

Action	Attribute	Type
operator *	**this**	`IMngPointer*`
operator = ■		

© Copyright IBM Corp. 1995, 1996

IMngPointer



Chapter 89. IVBag

Description: IBM VBag template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivbag.h

Derivation:
IACollection
IAEqualityCollection
IABag
IRBag
IVBagOnBase
IVGBag
IVBag

Preferred Features — IVBag

add Returns `True` if the element was added.
contains Returns `True` if the collection contains an element equal to the given element.
copy Copies the given collection to this collection
numberOfElements Returns the number of elements the collection contains.
remove Removes an element in the collection that is equal to the given element.
this A Pointer to the instance.

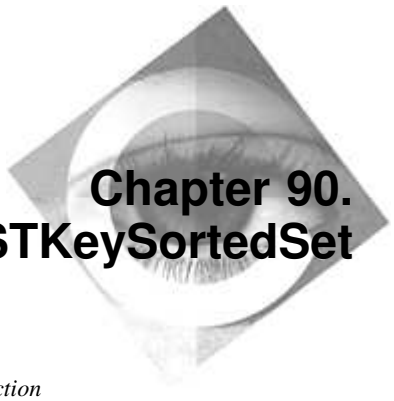
Features — IVBag

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfDifferentElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		

© Copyright IBM Corp. 1995, 1996

IVBag

Action	Attribute	Event
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 90. IVBagOnBSTKeySortedSet

Description: IBM VBag On BST Key Sorted Set template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivbag.h

Derivation:
IACollection
IAEqualityCollection
IABag
IRBag
IVBagOnBase
IVGBagOnBSTKeySortedSet
IVBagOnBSTKeySortedSet

Preferred Features — IVBagOnBSTKeySortedSet

add Returns `True` if the element was added.
contains Returns `True` if the collection contains an element equal to the given element.
copy Copies the given collection to this collection
numberOfElements Returns the number of elements the collection contains.
remove Removes an element in the collection that is equal to the given element.
this A Pointer to the instance.

Features — IVBagOnBSTKeySortedSet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfDifferentElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		

© Copyright IBM Corp. 1995, 1996

IVBagOnBSTKeySortedSet

Action	Attribute	Event
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		

Chapter 91. IVBagOnHashKeySet

Description: IBM VBag On Hash Key Set template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivbag.h

Derivation:
IACollection
IAEqualityCollection
IABag
IRBag
IVBagOnBase
IVGBagOnHashKeySet
IVBagOnHashKeySet

Preferred Features — IVBagOnHashKeySet

add	Returns `True` if the element was added.
contains	Returns `True` if the collection contains an element equal to the given element.
copy	Copies the given collection to this collection
numberOfElements	Returns the number of elements the collection contains.
remove	Removes an element in the collection that is equal to the given element.
this	A Pointer to the instance.

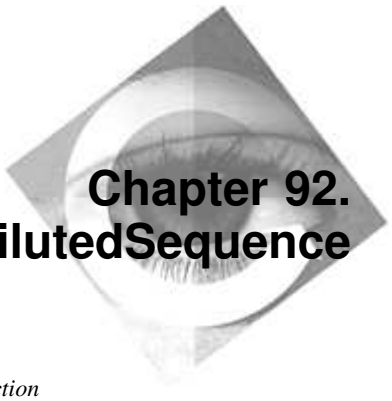
Features — IVBagOnHashKeySet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfDifferentElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		

© Copyright IBM Corp. 1995, 1996

IVBagOnHashKeySet

Action	Attribute	Event
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 92. IVBagOnSortedDilutedSequence

Description: IBM VBag On Sorted Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivbag.h

Derivation:
IACollection
IAEqualityCollection
IABag
IRBag
IVBagOnBase
IVGBagOnSortedDilutedSequence
IVBagOnSortedDilutedSequence

Preferred Features — IVBagOnSortedDilutedSequence

add Returns `True` if the element was added.
contains Returns `True` if the collection contains an element equal to the given element.
copy Copies the given collection to this collection
numberOfElements Returns the number of elements the collection contains.
remove Removes an element in the collection that is equal to the given element.
this A Pointer to the instance.

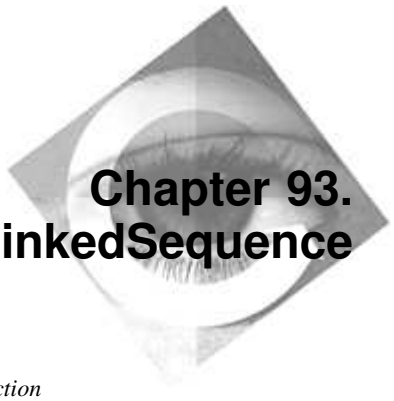
Features — IVBagOnSortedDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfDifferentElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		

© Copyright IBM Corp. 1995, 1996

IVBagOnSortedDilutedSequence

Action	Attribute	Event
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 93. IVBagOnSortedLinkedSequence

Description: IBM VBag On Sorted Linked Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivbag.h

Derivation:
IACollection
IAEqualityCollection
IABag
IRBag
IVBagOnBase
IVGBagOnSortedLinkedSequence
IVBagOnSortedLinkedSequence

Preferred Features — IVBagOnSortedLinkedSequence

add	Returns `True` if the element was added.
contains	Returns `True` if the collection contains an element equal to the given element.
copy	Copies the given collection to this collection
numberOfElements	Returns the number of elements the collection contains.
remove	Removes an element in the collection that is equal to the given element.
this	A Pointer to the instance.

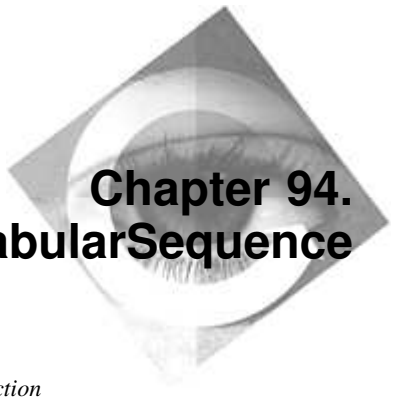
Features — IVBagOnSortedLinkedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfDifferentElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		

© Copyright IBM Corp. 1995, 1996

IVBagOnSortedLinkedSequence

Action	Attribute	Event
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 94. IVBagOnSortedTabularSequence

Description: IBM VBag On Sorted Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivbag.h

Derivation:
IACollection
IAEqualityCollection
IABag
IRBag
IVBagOnBase
IVGBagOnSortedTabularSequence
IVBagOnSortedTabularSequence

Preferred Features — IVBagOnSortedTabularSequence

add	Returns `True` if the element was added.
contains	Returns `True` if the collection contains an element equal to the given element.
copy	Copies the given collection to this collection
numberOfElements	Returns the number of elements the collection contains.
remove	Removes an element in the collection that is equal to the given element.
this	A Pointer to the instance.

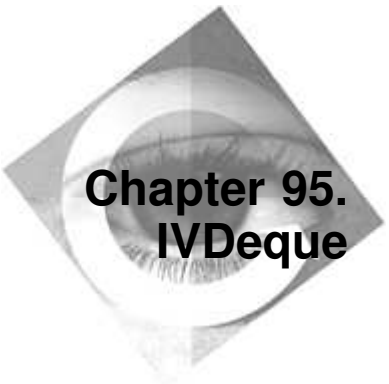
Features — IVBagOnSortedTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfDifferentElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		

© Copyright IBM Corp. 1995, 1996

IVBagOnSortedTabularSequence

Action	Attribute	Event
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 95. IVDeque

Description: IBM VDeque template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivdeque.h

Derivation:
IADeque
IRDeque
IVDequeOnBase
IVGDeque
IVDeque

Preferred Features — IVDeque

add Returns `True` if the element was added.

addAsFirst Adds the element to the collection as the first element in sequential order.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

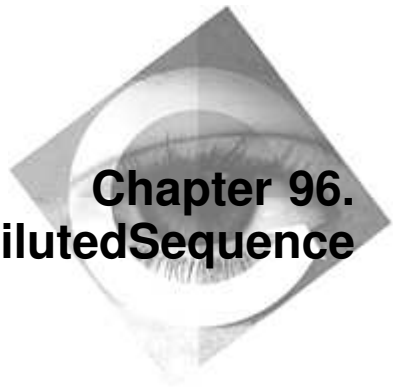
this A Pointer to the instance.

Features — IVDeque

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
allElementsDo ■	enabledForNotification ■	replacedEvent
compare ■	firstElement	
copy ■	full	
disableNotification	lastElement	
elementAtPosition ■	maxNumberOfElements	
operator = ■	numberOfElements	
removeAll	**this**	
removeFirst		
removeLast		

© Copyright IBM Corp. 1995, 1996

IVDeque



Chapter 96. IVDequeOnDilutedSequence

Description: IBM VDeque On Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivdeque.h

Derivation:
IADeque
IRDeque
IVDequeOnBase
IVGDequeOnDilutedSequence
IVDequeOnDilutedSequence

Preferred Features — IVDequeOnDilutedSequence

add Returns `True` if the element was added.

addAsFirst Adds the element to the collection as the first element in sequential order.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

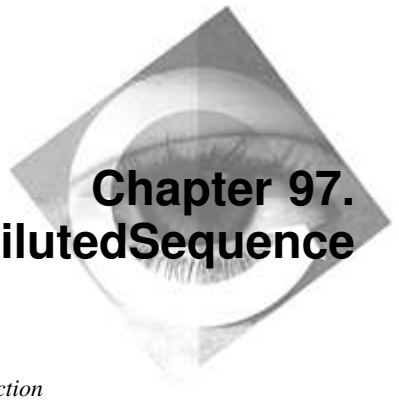
this A Pointer to the instance.

Features — IVDequeOnDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
allElementsDo ■	enabledForNotification ■	replacedEvent
compare ■	firstElement	
copy ■	full	
disableNotification	lastElement	
elementAtPosition ■	maxNumberOfElements	
operator = ■	numberOfElements	
removeAll	**this**	
removeFirst		
removeLast		

© Copyright IBM Corp. 1995, 1996

IVDequeOnDilutedSequence



Chapter 97. IVDilutedSequence

Description: IBM VDiluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivseq.h

Derivation:
IACollection
IAOrderedCollection
IASequentialCollection
IASequence
IRSequence
IVSequenceOnBase
IVGDilutedSequence
IVDilutedSequence

Preferred Features — IVDilutedSequence

add Returns `True` if the element was added.
addAsFirst Adds the element to the collection as the first element in sequential order.
addAsLast Adds the element to the collection as the last element in sequential order.
addAtPosition Adds the element at the given position to the collection.
copy Copies the given collection to this collection
elementAtPosition
Returns a reference to the element at the given position in the collection.
numberOfElements
Returns the number of elements the collection contains.
removeAtPosition
Removes the element from the collection, which is at the given position.
this A Pointer to the instance.

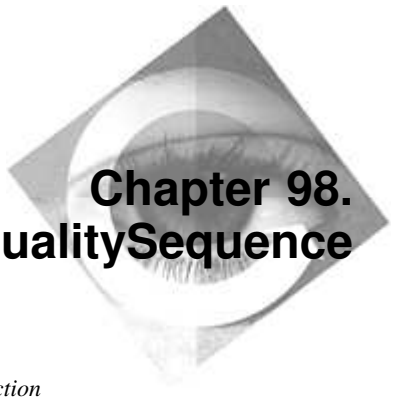
Features — IVDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	

© Copyright IBM Corp. 1995, 1996

IVDilutedSequence

Action	Attribute	Event
compare ■	full	
copy ■	lastElement	
disableNotification	maxNumberOfElements	
elementAtPosition ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		



Chapter 98. IVEqualitySequence

Description: IBM VEquality Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: iveqseq.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySequence
IREqualitySequence
IVEqualitySequenceOnBase
IVGEqualitySequence
IVEqualitySequence

Preferred Features — IVEqualitySequence

add Returns `True` if the element was added.

addAsFirst Adds the element to the collection as the first element in sequential order.

addAsLast Adds the element to the collection as the last element in sequential order.

addAtPosition Adds the element at the given position to the collection.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

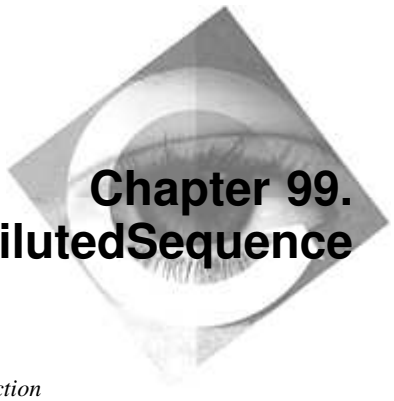
this A Pointer to the instance.

© Copyright IBM Corp. 1995, 1996

IVEqualitySequence

Features — IVEqualitySequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		



Chapter 99. IVEqualitySequenceOnDilutedSequence

Description: IBM VEquality Sequence On Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: iveqseq.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySequence
IREqualitySequence
IVEqualitySequenceOnBase
IVGEqualitySequenceOnDilutedSequence
IVEqualitySequenceOnDilutedSequence

Preferred Features — IVEqualitySequenceOnDilutedSequence

add Returns `True` if the element was added.

addAsFirst Adds the element to the collection as the first element in sequential order.

addAsLast Adds the element to the collection as the last element in sequential order.

addAtPosition Adds the element at the given position to the collection.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

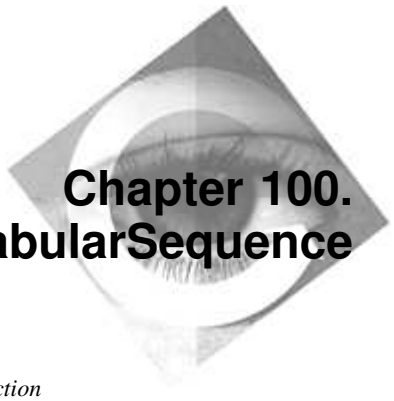
this A Pointer to the instance.

© Copyright IBM Corp. 1995, 1996

IVEqualitySequenceOnDilutedSequence

Features — IVEqualitySequenceOnDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		



Chapter 100. IVEqualitySequenceOnTabularSequence

Description: IBM VEquality Sequence On Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: iveqseq.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySequence
IREqualitySequence
IVEqualitySequenceOnBase
IVGEqualitySequenceOnTabularSequence
IVEqualitySequenceOnTabularSequence

Preferred Features — IVEqualitySequenceOnTabularSequence

add Returns `True` if the element was added.

addAsFirst Adds the element to the collection as the first element in sequential order.

addAsLast Adds the element to the collection as the last element in sequential order.

addAtPosition Adds the element at the given position to the collection.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

© Copyright IBM Corp. 1995, 1996

IVEqualitySequenceOnTabularSequence

Features — IVEqualitySequenceOnTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		

Chapter 101. IVHeap

Description: IBM VHeap template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivheap.h

Derivation:
IACollection
IAHeap
IRHeap
IVHeapOnBase
IVGHeap
IVHeap

Preferred Features — IVHeap

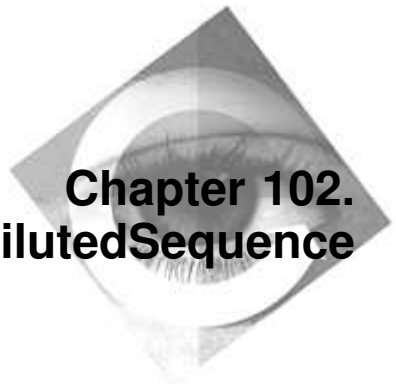
add Returns `True` if the element was added.
copy Copies the given collection to this collection
numberOfElements Returns the number of elements the collection contains.
this A Pointer to the instance.

Features — IVHeap

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
allElementsDo ■	consistent	modifiedEvent
copy ■	empty	removedEvent
disableNotification	enabledForNotification ■	replacedEvent
operator = ■	full	
removeAll	maxNumberOfElements	
	numberOfElements	
	this	

© Copyright IBM Corp. 1995, 1996

IVHeap



Chapter 102. IVHeapOnDilutedSequence

Description: IBM VHeap On Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivheap.h

Derivation:
IACollection
IAHeap
IRHeap
IVHeapOnBase
IVGHeapOnDilutedSequence
IVHeapOnDilutedSequence

Preferred Features — IVHeapOnDilutedSequence

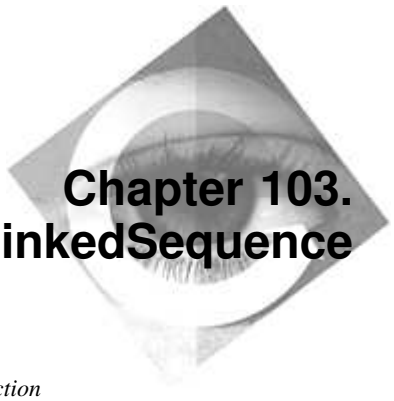
add Returns `True` if the element was added.
copy Copies the given collection to this collection
numberOfElements
Returns the number of elements the collection contains.
this A Pointer to the instance.

Features — IVHeapOnDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
allElementsDo ■	consistent	modifiedEvent
copy ■	empty	removedEvent
disableNotification	enabledForNotification ■	replacedEvent
operator = ■	full	
removeAll	maxNumberOfElements	
	numberOfElements	
	this	

© Copyright IBM Corp. 1995, 1996

IVHeapOnDilutedSequence



Chapter 103. IVLinkedSequence

Description: IBM VLinked Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivseq.h

Derivation:
IACollection
IAOrderedCollection
IASequentialCollection
IASequence
IRSequence
IVSequenceOnBase
IVGLinkedSequence
IVLinkedSequence

Preferred Features — IVLinkedSequence

add Returns `True` if the element was added.
addAsFirst Adds the element to the collection as the first element in sequential order.
addAsLast Adds the element to the collection as the last element in sequential order.
addAtPosition Adds the element at the given position to the collection.
copy Copies the given collection to this collection
elementAtPosition
Returns a reference to the element at the given position in the collection.
numberOfElements
Returns the number of elements the collection contains.
removeAtPosition
Removes the element from the collection, which is at the given position.
this A Pointer to the instance.

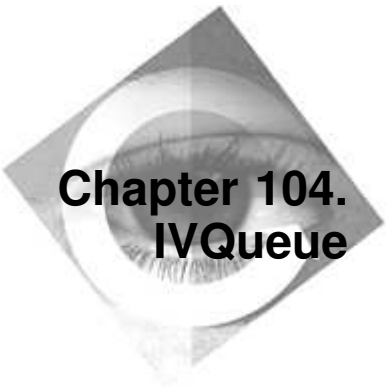
Features — IVLinkedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	

© Copyright IBM Corp. 1995, 1996

IVLinkedSequence

Action	Attribute	Event
compare ■	full	
copy ■	lastElement	
disableNotification	maxNumberOfElements	
elementAtPosition ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		



Chapter 104. IVQueue

Description: IBM VQueue template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivqueue.h

Derivation:
IAQueue
IRQueue
IVQueueOnBase
IVGQueue
IVQueue

Preferred Features — IVQueue

add Returns `True` if the element was added.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

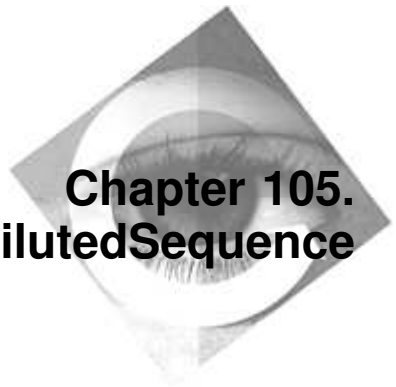
this A Pointer to the instance.

Features — IVQueue

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsLast ■	consistent	modifiedEvent
allElementsDo ■	empty	removedEvent
compare ■	enabledForNotification ■	replacedEvent
copy ■	firstElement	
dequeue ■	full	
disableNotification	lastElement	
elementAtPosition ■	maxNumberOfElements	
enqueue ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeFirst		

© Copyright IBM Corp. 1995, 1996

IVQueue



Chapter 105. IVQueueOnDilutedSequence

Description: IBM VQueue On Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivqueue.h

Derivation:
IAQueue
IRQueue
IVQueueOnBase
IVGQueueOnDilutedSequence
IVQueueOnDilutedSequence

Preferred Features — IVQueueOnDilutedSequence

add Returns `True` if the element was added.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

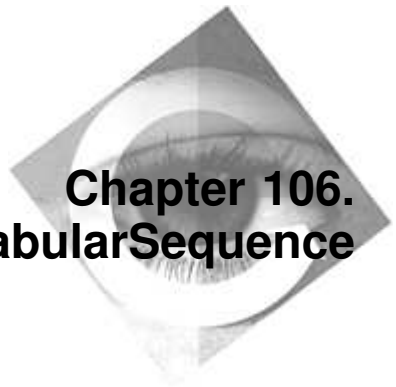
this A Pointer to the instance.

Features — IVQueueOnDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsLast ■	consistent	modifiedEvent
allElementsDo ■	empty	removedEvent
compare ■	enabledForNotification ■	replacedEvent
copy ■	firstElement	
dequeue ■	full	
disableNotification	lastElement	
elementAtPosition ■	maxNumberOfElements	
enqueue ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeFirst		

© Copyright IBM Corp. 1995, 1996

IVQueueOnDilutedSequence



Chapter 106. IVQueueOnTabularSequence

Description: IBM VQueue On Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivqueue.h

Derivation:
IAQueue
IRQueue
IVQueueOnBase
IVGQueueOnTabularSequence
IVQueueOnTabularSequence

Preferred Features — IVQueueOnTabularSequence

add Returns `True` if the element was added.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

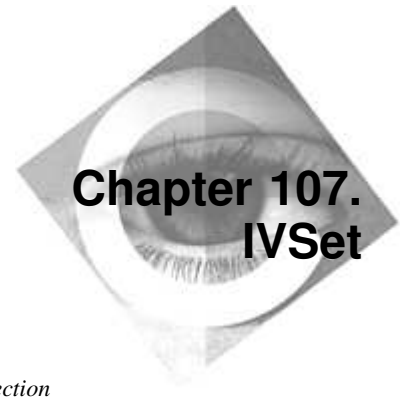
this A Pointer to the instance.

Features — IVQueueOnTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsLast ■	consistent	modifiedEvent
allElementsDo ■	empty	removedEvent
compare ■	enabledForNotification ■	replacedEvent
copy ■	firstElement	
dequeue ■	full	
disableNotification	lastElement	
elementAtPosition ■	maxNumberOfElements	
enqueue ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeFirst		

© Copyright IBM Corp. 1995, 1996

IVQueueOnTabularSequence



Chapter 107. IVSet

Description: IBM VSet template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivset.h

Derivation:
IACollection
IAEqualityCollection
IASet
IRSet
IVSetOnBase
IVGSet
IVSet

Preferred Features — IVSet

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

numberOfElements Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

this A Pointer to the instance.

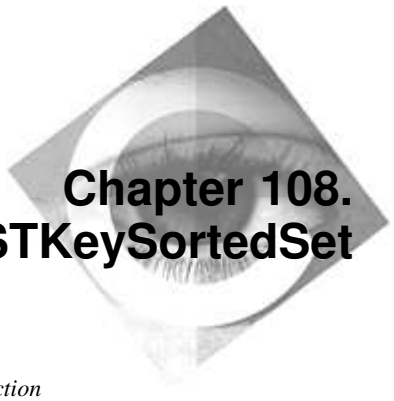
Features — IVSet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfElements	
copy ■	**this**	
differenceWith ■		
disableNotification		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		

© Copyright IBM Corp. 1995, 1996

IVSet

Action	Attribute	Event
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 108. IVSetOnBSTKeySortedSet

Description: IBM VSet On BSTKey Sorted Set template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivset.h

Derivation:
IACollection
IAEqualityCollection
IASet
IRSet
IVSetOnBase
IVGSetOnBSTKeySortedSet
IVSetOnBSTKeySortedSet

Preferred Features — IVSetOnBSTKeySortedSet

add	Returns `True` if the element was added.
contains	Returns `True` if the collection contains an element equal to the given element.
copy	Copies the given collection to this collection
numberOfElements	Returns the number of elements the collection contains.
remove	Removes an element in the collection that is equal to the given element.
this	A Pointer to the instance.

Features — IVSetOnBSTKeySortedSet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfElements	
copy ■	**this**	
differenceWith ■		
disableNotification		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		

© Copyright IBM Corp. 1995, 1996

IVSetOnBSTKeySortedSet

Action	Attribute	Event
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		

Chapter 109. IVSetOnHashKeySet

Description: IBM VSet On Hash Key Set template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivset.h

Derivation:
IACollection
IAEqualityCollection
IASet
IRSet
IVSetOnBase
IVGSetOnHashKeySet
IVSetOnHashKeySet

Preferred Features — IVSetOnHashKeySet

add Returns `True` if the element was added.
contains Returns `True` if the collection contains an element equal to the given element.
copy Copies the given collection to this collection
numberOfElements Returns the number of elements the collection contains.
remove Removes an element in the collection that is equal to the given element.
this A Pointer to the instance.

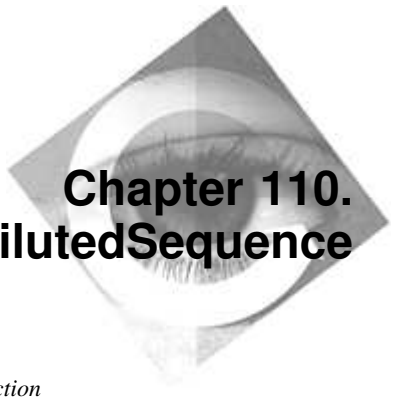
Features — IVSetOnHashKeySet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfElements	
copy ■	**this**	
differenceWith ■		
disableNotification		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		

© Copyright IBM Corp. 1995, 1996

IVSetOnHashKeySet

Action	Attribute	Event
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 110. IVSetOnSortedDilutedSequence

Description: IBM VSet On Sorted Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivset.h

Derivation:
IACollection
IAEqualityCollection
IASet
IRSet
IVSetOnBase
IVGSetOnSortedDilutedSequence
IVSetOnSortedDilutedSequence

Preferred Features — IVSetOnSortedDilutedSequence

add Returns `True` if the element was added.
contains Returns `True` if the collection contains an element equal to the given element.
copy Copies the given collection to this collection
numberOfElements Returns the number of elements the collection contains.
remove Removes an element in the collection that is equal to the given element.
this A Pointer to the instance.

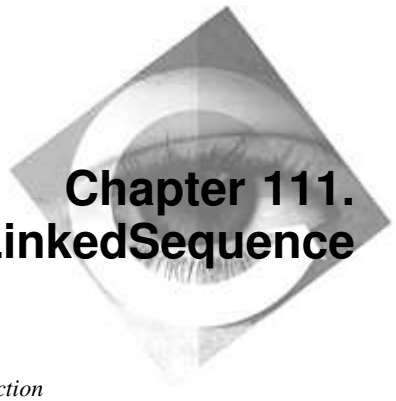
Features — IVSetOnSortedDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfElements	
copy ■	**this**	
differenceWith ■		
disableNotification		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		

© Copyright IBM Corp. 1995, 1996

IVSetOnSortedDilutedSequence

Action	Attribute	Event
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 111. IVSetOnSortedLinkedSequence

Description: IBM VSet On Sorted Linked Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivset.h

Derivation:
IACollection
IAEqualityCollection
IASet
IRSet
IVSetOnBase
IVGSetOnSortedLinkedSequence
IVSetOnSortedLinkedSequence

Preferred Features — IVSetOnSortedLinkedSequence

add	Returns `True` if the element was added.
contains	Returns `True` if the collection contains an element equal to the given element.
copy	Copies the given collection to this collection
numberOfElements	Returns the number of elements the collection contains.
remove	Removes an element in the collection that is equal to the given element.
this	A Pointer to the instance.

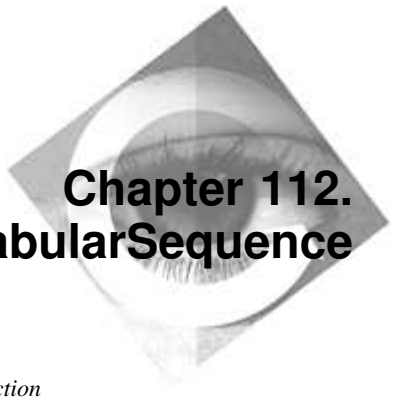
Features — IVSetOnSortedLinkedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfElements	
copy ■	**this**	
differenceWith ■		
disableNotification		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		

© Copyright IBM Corp. 1995, 1996

IVSetOnSortedLinkedSequence

Action	Attribute	Event
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 112. IVSetOnSortedTabularSequence

Description: IBM VSet On Sorted Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivset.h

Derivation:
IACollection
IAEqualityCollection
IASet
IRSet
IVSetOnBase
IVGSetOnSortedTabularSequence
IVSetOnSortedTabularSequence

Preferred Features — IVSetOnSortedTabularSequence

add	Returns `True` if the element was added.
contains	Returns `True` if the collection contains an element equal to the given element.
copy	Copies the given collection to this collection
numberOfElements	Returns the number of elements the collection contains.
remove	Removes an element in the collection that is equal to the given element.
this	A Pointer to the instance.

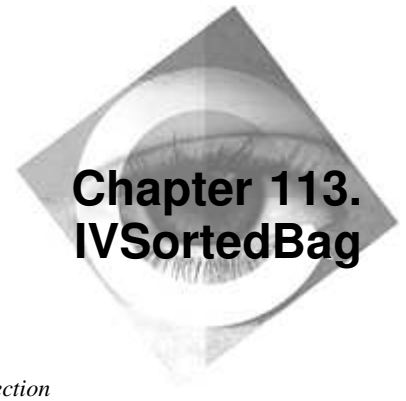
Features — IVSetOnSortedTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	full	
contains ■	maxNumberOfElements	
containsAllFrom ■	numberOfElements	
copy ■	**this**	
differenceWith ■		
disableNotification		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		

© Copyright IBM Corp. 1995, 1996

IVSetOnSortedTabularSequence

Action	Attribute	Event
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
unionWith ■		



Chapter 113. IVSortedBag

Description: IBM VSorted Bag template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtbag.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedBag
IRSortedBag
IVSortedBagOnBase
IVGSortedBag
IVSortedBag

Preferred Features — IVSortedBag

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

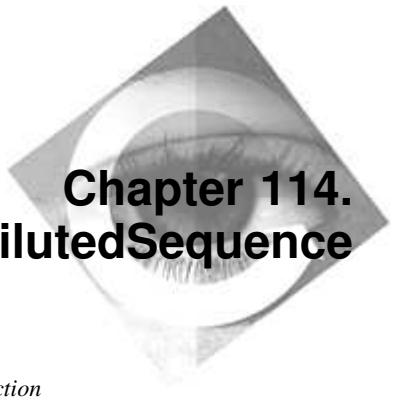
Features — IVSortedBag

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedBag

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfDifferentElements	
differenceWith ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 114. IVSortedBagOnSortedDilutedSequence

Description: IBM VSorted Bag On Sorted Diluted Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtbag.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedBag
IRSortedBag
IVSortedBagOnBase
IVGSortedBagOnSortedDilutedSequence
IVSortedBagOnSortedDilutedSequence

Preferred Features — IVSortedBagOnSortedDilutedSequence

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

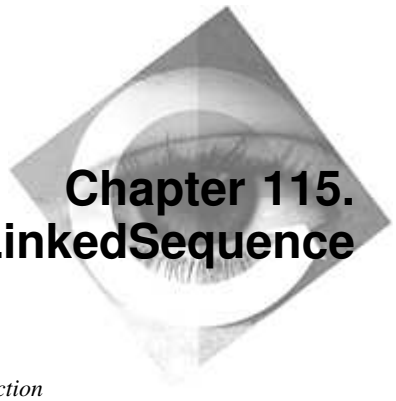
Features — IVSortedBagOnSortedDilutedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedBagOnSortedDilutedSequence

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfDifferentElements	
differenceWith ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 115. IVSortedBagOnSortedLinkedSequence

Description: IBM VSorted Bag On Sorted Linked Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtbag.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedBag
IRSortedBag
IVSortedBagOnBase
IVGSortedBagOnSortedLinkedSequence
IVSortedBagOnSortedLinkedSequence

Preferred Features — IVSortedBagOnSortedLinkedSequence

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

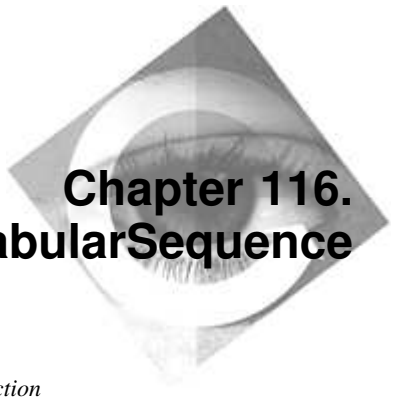
Features — IVSortedBagOnSortedLinkedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedBagOnSortedLinkedSequence

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfDifferentElements	
differenceWith ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 116. IVSortedBagOnSortedTabularSequence

Description: IBM VSorted Bag On Sorted Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtbag.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedBag
IRSortedBag
IVSortedBagOnBase
IVGSortedBagOnSortedTabularSequence
IVSortedBagOnSortedTabularSequence

Preferred Features — IVSortedBagOnSortedTabularSequence

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

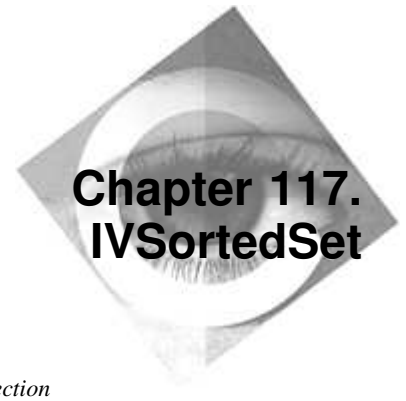
Features — IVSortedBagOnSortedTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedBagOnSortedTabularSequence

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfDifferentElements	
differenceWith ■	numberOfElements	
disableNotification	**this**	
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
numberOfOccurrences ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 117. IVSortedSet

Description: IBM VSorted Set template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtset.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedSet
IRSortedSet
IVSortedSetOnBase
IVGSortedSet
IVSortedSet

Preferred Features — IVSortedSet

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

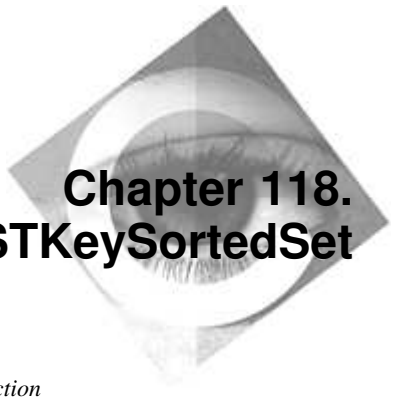
Features — IVSortedSet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedSet

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 118. IVSortedSetOnBSTKeySortedSet

Description: IBM VSorted Set On BSTKey Sorted Set template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtset.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedSet
IRSortedSet
IVSortedSetOnBase
IVGSortedSetOnBSTKeySortedSet
IVSortedSetOnBSTKeySortedSet

Preferred Features — IVSortedSetOnBSTKeySortedSet

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

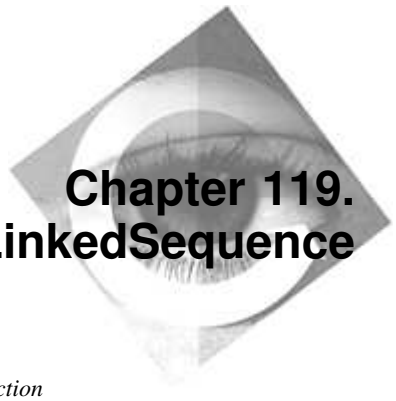
Features — IVSortedSetOnBSTKeySortedSet

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedSetOnBSTKeySortedSet

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 119. IVSortedSetOnSortedLinkedSequence

Description: IBM VSorted Set On Sorted Linked Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtset.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedSet
IRSortedSet
IVSortedSetOnBase
IVGSortedSetOnSortedLinkedSequence
IVSortedSetOnSortedLinkedSequence

Preferred Features — IVSortedSetOnSortedLinkedSequence

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

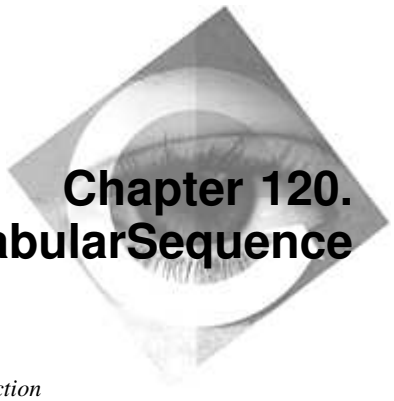
Features — IVSortedSetOnSortedLinkedSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedSetOnSortedLinkedSequence

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		



Chapter 120. IVSortedSetOnSortedTabularSequence

Description: IBM VSorted Set On Sorted Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivsrtset.h

Derivation:
IACollection
IAEqualityCollection
IAEqualitySortedCollection
IASortedSet
IRSortedSet
IVSortedSetOnBase
IVGSortedSetOnSortedTabularSequence
IVSortedSetOnSortedTabularSequence

Preferred Features — IVSortedSetOnSortedTabularSequence

add Returns `True` if the element was added.

contains Returns `True` if the collection contains an element equal to the given element.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

remove Removes an element in the collection that is equal to the given element.

removeAtPosition
Removes the element from the collection, which is at the given position.

this A Pointer to the instance.

Features — IVSortedSetOnSortedTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addDifference ■	consistent	modifiedEvent
addIntersection ■	empty	removedEvent
addUnion ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	
compare ■	full	

© Copyright IBM Corp. 1995, 1996

IVSortedSetOnSortedTabularSequence

Action	Attribute	Event
contains ■	lastElement	
containsAllFrom ■	maxNumberOfElements	
copy ■	numberOfElements	
differenceWith ■	**this**	
disableNotification		
elementAtPosition ■		
intersectionWith ■		
locateOrAdd ■		
operator != ■		
operator = ■		
operator == ■		
remove ■		
removeAll		
removeAllOccurrences ■		
removeAtPosition ■		
removeFirst		
removeLast		
unionWith ■		

Chapter 121. IVStack

Description: IBM VStack template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivstack.h

Derivation:
IAStack
IRStack
IVStackOnBase
IVGStack
IVStack

Preferred Features — IVStack

add Returns `True` if the element was added.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

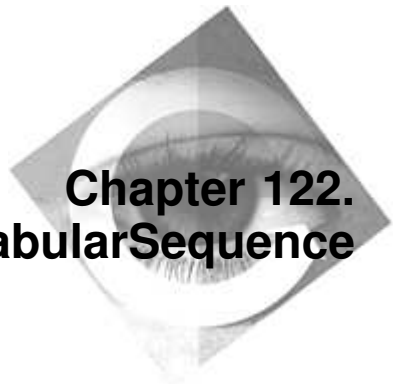
this A Pointer to the instance.

Features — IVStack

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsLast ■	consistent	modifiedEvent
allElementsDo ■	empty	removedEvent
compare ■	enabledForNotification ■	replacedEvent
copy ■	firstElement	
disableNotification	full	
elementAtPosition ■	lastElement	
operator = ■	maxNumberOfElements	
pop	numberOfElements	
push ■	**this**	
removeAll	top	
removeLast		

© Copyright IBM Corp. 1995, 1996

IVStack



Chapter 122. IVStackOnTabularSequence

Description: IBM VStack On Tabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivstack.h

Derivation:
IAStack
IRStack
IVStackOnBase
IVGStackOnTabularSequence
IVStackOnTabularSequence

Preferred Features — IVStackOnTabularSequence

add Returns `True` if the element was added.

addAsLast Adds the element to the collection as the last element in sequential order.

copy Copies the given collection to this collection

elementAtPosition
Returns a reference to the element at the given position in the collection.

numberOfElements
Returns the number of elements the collection contains.

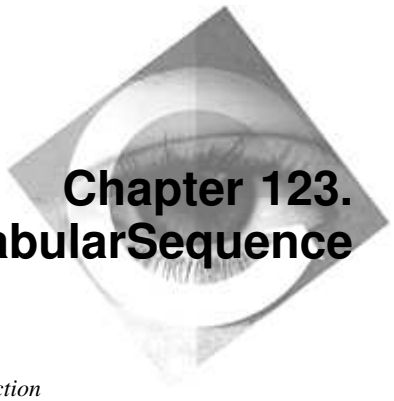
this A Pointer to the instance.

Features — IVStackOnTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsLast ■	consistent	modifiedEvent
allElementsDo ■	empty	removedEvent
compare ■	enabledForNotification ■	replacedEvent
copy ■	firstElement	
disableNotification	full	
elementAtPosition ■	lastElement	
operator = ■	maxNumberOfElements	
pop	numberOfElements	
push ■	**this**	
removeAll	top	
removeLast		

© Copyright IBM Corp. 1995, 1996

IVStackOnTabularSequence



Chapter 123. IVTabularSequence

Description: IBM VTabular Sequence template collection class
Part type: Nonvisual
Part file: vbcc
Header file: ivseq.h

Derivation:
IACollection
IAOrderedCollection
IASequentialCollection
IASequence
IRSequence
IVSequenceOnBase
IVGTabularSequence
IVTabularSequence

Preferred Features — IVTabularSequence

add Returns `True` if the element was added.
addAsFirst Adds the element to the collection as the first element in sequential order.
addAsLast Adds the element to the collection as the last element in sequential order.
addAtPosition Adds the element at the given position to the collection.
copy Copies the given collection to this collection
elementAtPosition
Returns a reference to the element at the given position in the collection.
numberOfElements
Returns the number of elements the collection contains.
removeAtPosition
Removes the element from the collection, which is at the given position.
this A Pointer to the instance.

Features — IVTabularSequence

Action	Attribute	Event
add ■	anyElement	addedEvent
addAllFrom ■	bounded	anyEvent
addAsFirst ■	consistent	modifiedEvent
addAsLast ■	empty	removedEvent
addAtPosition ■	enabledForNotification ■	replacedEvent
allElementsDo ■	firstElement	

© Copyright IBM Corp. 1995, 1996

IVTabularSequence

Action	Attribute	Event
compare ■	full	
copy ■	lastElement	
disableNotification	maxNumberOfElements	
elementAtPosition ■	numberOfElements	
operator = ■	**this**	
removeAll		
removeAtPosition ■		
removeFirst		
removeLast		
sort ■		

Part 4. Parts in vbmm.vbb

The following chapters provide information on Visual Builder multimedia parts that you can use in your applications.

To use these parts, load the vbmm.vbb part file from the ibmcppw\ivb directory. Also, ensure that you have multimedia support installed on your system. Windows automatically installs multimedia support.

Chapter 124. IMM24FramesPerSecondTime 489
Preferred Features — IMM24FramesPerSecondTime 489
Features — IMM24FramesPerSecondTime 489

Chapter 125. IMM25FramesPerSecondTime 491
Preferred Features — IMM25FramesPerSecondTime 491
Features — IMM25FramesPerSecondTime 491

Chapter 126. IMM30FramesPerSecondTime 493
Preferred Features — IMM30FramesPerSecondTime 493
Features — IMM30FramesPerSecondTime 493

Chapter 127. IMMAmpMixer 495
Portability — IMMAmpMixer 495
Related Parts — IMMAmpMixer 495
Building Guidelines — IMMAmpMixer 496
Preferred Features — IMMAmpMixer 497
Features — IMMAmpMixer 497

Chapter 128. IMMAudioBuffer 499
Portability — IMMAudioBuffer 499
Preferred Features — IMMAudioBuffer 499
Features — IMMAudioBuffer 499

Chapter 129. IMMAudioCD 501
Portability — IMMAudioCD 501
Related Parts — IMMAudioCD 501
Building Guidelines — IMMAudioCD 502
Preferred Features — IMMAudioCD 503
Features — IMMAudioCD 504

Chapter 130. IMMAudioCDContents 507

© Copyright IBM Corp. 1995, 1996

Parts in vbmm.vbb

Preferred Features — IMMAudioCDContents 507
Features — IMMAudioCDContents 507

Chapter 131. IMMDigitalVideo 509
Portability — IMMDigitalVideo 509
Related Parts — IMMDigitalVideo 510
Building Guidelines — IMMDigitalVideo 510
Preferred Features — IMMDigitalVideo 512
Features — IMMDigitalVideo 512

Chapter 132. IMMErrorInfo 515
Preferred Features — IMMErrorInfo 515
Features — IMMErrorInfo 515

Chapter 133. IMMHourMinSecFrameTime 517
Preferred Features — IMMHourMinSecFrameTime 517
Features — IMMHourMinSecFrameTime 517

Chapter 134. IMMHourMinSecTime 519
Preferred Features — IMMHourMinSecTime 519
Features — IMMHourMinSecTime 519

Chapter 135. IMMMasterAudio 521
Related Parts — IMMMasterAudio 521
Building Guidelines — IMMMasterAudio 522
Preferred Features — IMMMasterAudio 523
Features — IMMMasterAudio 523

Chapter 136. IMMMillisecondTime 525
Preferred Features — IMMMillisecondTime 525
Features — IMMMillisecondTime 525

Chapter 137. IMMMinSecFrameTime 527
Preferred Features — IMMMinSecFrameTime 527
Features — IMMMinSecFrameTime 527

Chapter 138. IMMPlayerPanel 529
Related Parts — IMMPlayerPanel 529
Building Guidelines — IMMPlayerPanel 530
Preferred Features — IMMPlayerPanel 531
Features — IMMPlayerPanel 531

Chapter 139. IMMSequencer 533

Portability — IMMSequencer 533
Related Parts — IMMSequencer 533
Building Guidelines — IMMSequencer 534
Preferred Features — IMMSequencer 535
Features — IMMSequencer 536

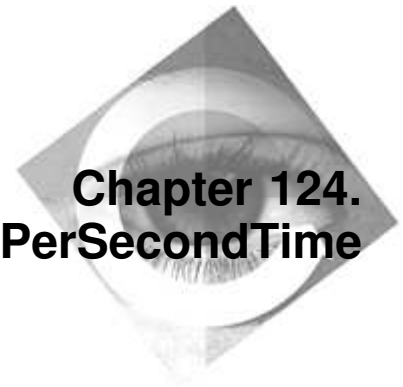
Chapter 140. IMMSpeed 537
Preferred Features — IMMSpeed 537
Features — IMMSpeed 537

Chapter 141. IMMTime 539
Preferred Features — IMMTime 539
Features — IMMTime 539

Chapter 142. IMMTrackMinSecFrameTime 541
Preferred Features — IMMTrackMinSecFrameTime 541
Features — IMMTrackMinSecFrameTime 541

Chapter 143. IMMWaveAudio 543
Portability — IMMWaveAudio 543
Related Parts — IMMWaveAudio 544
Building Guidelines — IMMWaveAudio 544
Preferred Features — IMMWaveAudio 546
Features — IMMWaveAudio 546

Parts in vbmm.vbb



Chapter 124. IMM24FramesPerSecondTime

Description: IBM support of 24-frames-per-second video
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMHourMinSecFrameTime
IMM24FramesPerSecondTime

Preferred Features — IMM24FramesPerSecondTime

frames	Returns the frames component of the time.
minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMM24FramesPerSecondTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	frames	`unsigned long`
operator -= ■	framesPerSecond	`unsigned long`
operator < ■	hours	`unsigned long`
operator <= ■	hundredths	`unsigned long`
operator = ■	minutes	`unsigned long`
operator == ■	ordinal ■	`unsigned long`
operator > ■	seconds	`unsigned long`
operator >= ■	**this**	`IMM24FramesPerSecondTime*`
operator unsigned long	thousandths	`unsigned long`
	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMM24FramesPerSecondTime

Chapter 125. IMM25FramesPerSecondTime

Description: IBM support of 25-frames-per-second video
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMHourMinSecFrameTime
IMM25FramesPerSecondTime

Preferred Features — IMM25FramesPerSecondTime

frames	Returns the frames component of the time.
minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMM25FramesPerSecondTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	frames	`unsigned long`
operator -= ■	framesPerSecond	`unsigned long`
operator < ■	hours	`unsigned long`
operator <= ■	hundredths	`unsigned long`
operator = ■	minutes	`unsigned long`
operator == ■	ordinal ■	`unsigned long`
operator > ■	seconds	`unsigned long`
operator >= ■	**this**	`IMM25FramesPerSecondTime*`
operator unsigned long	thousandths	`unsigned long`
	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMM25FramesPerSecondTime

Chapter 126. IMM30FramesPerSecondTime

Description: IBM support of 30-frames-per-second video
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMHourMinSecFrameTime
IMM30FramesPerSecondTime

Preferred Features — IMM30FramesPerSecondTime

frames	Returns the frames component of the time.
minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMM30FramesPerSecondTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	frames	`unsigned long`
operator -= ■	framesPerSecond	`unsigned long`
operator < ■	hours	`unsigned long`
operator <= ■	hundredths	`unsigned long`
operator = ■	minutes	`unsigned long`
operator == ■	ordinal ■	`unsigned long`
operator > ■	seconds	`unsigned long`
operator >= ■	**this**	`IMM30FramesPerSecondTime*`
operator unsigned long	thousandths	`unsigned long`
	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMM30FramesPerSecondTime

Chapter 127. IMMAmpMixer

Description: IBM amplifier mixer
Part type: Nonvisual
Part file: vbmm
Header file: immamix.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IMMDevice
IMMAmpMixer

Use an IMMAmpMixer* part to provide standard audio control functions, such as volume and balance. The amplifier-mixer enables audio signals to be transferred to speakers or other sound devices.

Note: To use an IMMAmpMixer* part, ensure that you have installed multimedia support on your system.

The user can operate the amplifier-mixer through visual controls you provide. Circular sliders are well suited for visual control of an amplifier-mixer.

Portability — IMMAmpMixer

Features not defined for Windows:

- prerollTime
- prerollType

Related Parts — IMMAmpMixer

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer"
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Multimedia player parts:

© Copyright IBM Corp. 1995, 1996

- "IMMAudioCD" on page 501
- "IMMDigitalVideo" on page 509
- "IMMSequencer" on page 533
- "IMMWaveAudio" on page 543

Building Guidelines — IMMAmpMixer

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting Formats
- Setting Initial Audio Values

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing Audio Values

Adding the Part

To add an IMMAmpMixer* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Selecting Formats

To specify speed and time formats rather than using system-provided formats, do the following:

- Select a speed format in the **speedFormat** field on the IMMAmpMixer* part's settings page.
- Select a time format in the **timeFormat** field on the IMMAmpMixer* part's settings page.

Setting Initial Audio Values

To define initial amplification and mixing settings, specify values for the following audio controls on the IMMAmpMixer* part's settings page:

- **balance**
- **bass**
- **gain**
- **pitch**

- **treble**
- **volume**

The field values range from 0 to 100. Generally, 0 is the minimum setting of the control and 100 is the maximum setting. For **balance**, 0 is the maximum left speaker setting, and 100 is the maximum right speaker setting.

Changing Audio Values

To let the user change amplification and mixing settings, provide audio control dials. For example, provide a volume dial as follows:

- Place an ICircularSlider* part on a canvas.
- Define the circular slider as a volume control with values from 0 to 100.
- Connect the ICircularSlider* part's *value* attribute to the IMMAmpMixer* part's *volume* attribute.

Preferred Features — IMMAmpMixer

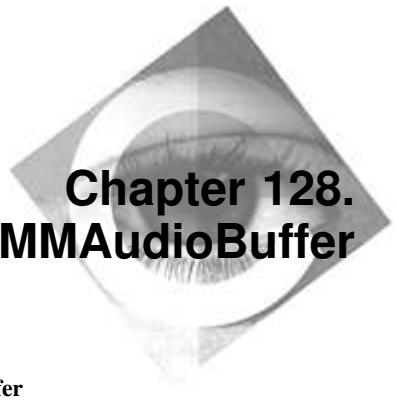
close	Closes a device.
open	Opens the device or file based on the passed in string.
this	A Pointer to the instance.

Features — IMMAmpMixer

Action	Attribute	Event
acquire ■	acquired	anyEvent
close ■	aliasName	commandNotifyEvent
connectedDeviceId ■	asDebugInfo	cuePointEvent
deletePendingEvents ■	asString	deviceEvent
disableAudio ■	audioEnabled ■	passDeviceEvent
disableConnector ■	**balance** ■	positionChangeEvent ■
disableMonitoring ■	**bass** ■	
disableNotification	closeOnDestroy ■	
enableConnector ■	description	
isConnectionSupported ■	deviceId	
isConnectorEnabled ■	deviceName	
notifyObservers ■	deviceType	
open ■	enabledForNotification ■	
openOnThread ■	**gain** ■	
operator = ■	isOpen	
release ■	mode	
supportsCommand ■	**monitoringEnabled** ■	
	pitch ■	
	requiresFiles	
	speedFormat ■	

IMMAmpMixer

Action	Attribute	Event
	supportsAudio	
	supportsDigitalTransfer	
	supportsDisableEject	
	supportsEject	
	supportsPlay	
	supportsRecord	
	supportsRecordInsertion	
	supportsSave	
	supportsStreaming	
	supportsVideo	
	supportsVolumeAdjustment	
	this	
	timeFormat ■	
	treble ■	
	volume ■	



Chapter 128. IMMAudioBuffer

Description: IBM audio buffer
Part type: Class
Part file: vbmm
Header file: immabuf.hpp

Derivation:
IBase
IMMAudioBuffer

Portability — IMMAudioBuffer

Features not defined for Windows:

- bitsPerSample
- blockAlignment
- bytesPerSecond
- channels
- contentType
- data
- format
- headerData
- length
- mediaType
- samplesPerSecond
- setData

Preferred Features — IMMAudioBuffer

this A Pointer to the instance.

Features — IMMAudioBuffer

Action	Attribute	Type
operator = ∎	asDebugInfo	`IString`
	asString	`IString`
	this	`IMMAudioBuffer*`

© Copyright IBM Corp. 1995, 1996

IMMAudioBuffer

Chapter 129. IMMAudioCD

Description: IBM audio compact disc device
Part type: Nonvisual
Part file: vbmm
Header file: immcdda.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IMMDevice
IMMPlayableDevice
IMMRemovableMedia
IMMAudioCD

Use an IMMAudioCD* part to provide standard audio compact disc (CD) player functions.

Note: To use an IMMAudioCD* part, ensure that you have installed multimedia support on your system.

The user can operate the CD player through visual controls. You can easily provide button controls with an IMMPlayerPanel* part. You can provide volume control for all audio devices with an IMMMasterAudio* part.

Portability — IMMAudioCD

Features not defined for Windows:

- lockDoor
- unlockDoor
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

Related Parts — IMMAudioCD

Multimedia player parts:

- "IMMAudioCD"
- "IMMDigitalVideo" on page 509

© Copyright IBM Corp. 1995, 1996

- "IMMSequencer" on page 533
- "IMMWaveAudio" on page 543

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — IMMAudioCD

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting Formats
- Setting the Initial Volume

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Volume
- Starting Player Actions

Adding the Part

To add an IMMAudioCD* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Selecting Formats

To specify speed and time formats rather than using system-provided formats, do the following:

- Select a speed format in the **speedFormat** field on the IMMAudioCD* part's settings page.
- Select a time format in the **timeFormat** field on the IMMAudioCD* part's settings page.

Setting the Initial Volume

To define an initial volume setting, enter a value in the **volume** field on the IMMAudioCD* part's settings page. Values range from 0 for no volume to 100 for the maximum volume.

If you are providing an IMMMasterAudio* part for volume control and are not providing a volume dial for the CD player, set the **volume** field value to 100.

Changing the Volume

To let the user change the CD player volume, provide a volume dial as follows:

- Place an ICircularSlider* part on a canvas.
- Define the circular slider as a volume control with values from 0 to 100.
- Connect the ICircularSlider* part's *value* attribute to the IMMAudioCD* part's *volume* attribute.

If you are providing volume control with an IMMMasterAudio* part as well as with the CD player, the master audio volume setting determines the maximum volume produced. The player volume is the product of the master audio volume setting and the player volume setting. For example, if the master audio volume is set at 50 and the player volume is set at 80, the player volume is 40 percent of its potential maximum.

Starting Player Actions

To let the user start CD player actions, provide animated buttons for the actions. For example, provide a play button as follows:

- Place an IAnimatedButton* part on a canvas.
- Define the animated button as a latchable play button.
- Connect the IAnimatedButton* part's *buttonClickEvent* feature to the IMMAudioCD* part's *play* action.

Preferred Features — IMMAudioCD

close	Closes a device.
closeDoor	Retracts the tray and closes the door, if possible.
contents	Returns a copy of the CD's table of contents.
goToEntry	Moves the current position to the passed in location.
open	Opens the device or file based on the passed in string.
openDoor	Opens the door and ejects the tray, if possible.
pause	If the device is playing, then it pauses the device.
play	Starts playing the device from the passed in start position to the passed in end position.

position	Returns the current position in the current time format as an `unsigned long`.
seekToEnd	Moves the current position to the end of the data in the device.
seekToStart	Moves the current position to the start of the data in the device.
stepFrame	Steps the play one or more time units forward or backward.
stop	Stops playback of the device.
this	A Pointer to the instance.
trackBackward	Moves the current position backwards the passed in number of tracks.
trackForward	Move the current position forwards the passed in number of tracks.

Features — IMMAudioCD

Action	Attribute	Event
acquire ■	acquired	anyEvent
close ■	aliasName	commandNotifyEvent
closeDoor ■	asDebugInfo	cuePointEvent
connectedDeviceId ■	asString	deviceEvent
cueForPlayback ■	audioEnabled ■	mediaLoadedEvent
deletePendingEvents ■	**autoPlayEnabled** ■	passDeviceEvent
disableAudio ■	closeOnDestroy ■	positionChangeEvent ■
disableAutoPlay	**contents**	**positionTimerEvent** ■
disableConnector ■	**continuousPlayEnabled** ■	**trackStartedEvent**
disableContinuousPlay	description	
disableNotification	deviceId	
enableConnector ■	deviceName	
goToEntry ■	deviceType	
isConnectionSupported ■	**discId**	
isConnectorEnabled ■	**discTitle** ■	
notifyObservers ■	enabledForNotification ■	
open ■	isOpen	
openDoor ■	length	
openOnThread ■	mediaPresent	
operator = ■	mode	
pause ■	**numberOfTracks**	
play ■	position	
release ■	**profile** ■	
resume ■	requiresFiles	
seek ■	speedFormat ■	
seekToEnd ■	supportsAudio	
seekToStart ■	supportsDigitalTransfer	
setProgram ■	supportsDisableEject	
setTrackTitle ■	supportsEject	
startPositionTracking ■	supportsPlay	
startScanningBackward	supportsRecord	
startScanningForward	supportsRecordInsertion	

IMMAudioCD

Action	Attribute	Event
stepFrame ■	supportsSave	
stop ■	supportsStreaming	
stopPositionTracking ■	supportsVideo	
supportsCommand ■	supportsVolumeAdjustment	
trackBackward ■	**this**	
trackForward ■	timeFormat ■	
trackTitle ■	**upc**	
	volume ■	

IMMAudioCD

Chapter 130. IMMAudioCDContents

Description: IBM table of contents for audio compact disc
Part type: Class
Part file: vbmm
Header file: immcdda.hpp

Derivation:
IBase
IVBase
IMMAudioCDContents

Preferred Features — IMMAudioCDContents

this A Pointer to the instance.

Features — IMMAudioCDContents

Action	Attribute	Type
addEntryAsFirst ■	asDebugInfo	`IString`
operator = ■	asString	`IString`
	discId	`IString`
	numberOfEntries	`unsigned long`
	numberOfTracks	`unsigned long`
	this	`IMMAudioCDContents*`
	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMMAudioCDContents

Chapter 131. IMMDigitalVideo

Description: IBM digital video
Part type: Nonvisual
Part file: vbmm
Header file: immdigvd.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IMMDevice
IMMPlayableDevice
IMMFileMedia
IMMRecordable
IMMConfigurableAudio
IMMDigitalVideo

Use an IMMDigitalVideo* part to provide standard video player functions.

Note: To use an IMMDigitalVideo* part, ensure that you have installed multimedia support on your system.

The user can operate the video player through visual controls. You can easily provide button controls with an IMMPlayerPanel* part. You can provide volume control for all audio devices with an IMMMasterAudio* part.

Portability — IMMDigitalVideo

Features not defined for Windows:

- fastSpeed
- monitoringEnabled
- monitorHandle
- slowSpeed
- disableMonitoring
- refresh
- setMonitorWindow
- useDefaultMonitorWindow
- canRedo
- canUndo
- redo

© Copyright IBM Corp. 1995, 1996

- undo
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

Related Parts — IMMDigitalVideo

Multimedia player parts:

- "IMMAudioCD" on page 501
- "IMMDigitalVideo" on page 509
- "IMMSequencer" on page 533
- "IMMWaveAudio" on page 543

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — IMMDigitalVideo

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting Formats
- Defining Audio Channels
- Setting the Initial Volume

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Volume
- Starting Player Actions

Adding the Part

To add an IMMDigitalVideo* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.

- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Selecting Formats

To specify audio, speed, and time formats rather than using system-provided formats, do the following:

- Select an audio format in the **format** field on the IMMDigitalVideo* part's settings page.
- Select a speed format in the **speedFormat** field on the IMMDigitalVideo* part's settings page.
- Select a time format in the **timeFormat** field on the IMMDigitalVideo* part's settings page.

Defining Audio Channels

To define the number of audio output channels for the video player, enter a number in the **channels** field on the IMMDigitalVideo* part's settings page. Enter 1 for monophonic (mono) output or 2 for stereophonic (stereo) output.

Setting the Initial Volume

To define an initial volume setting, enter a value in the **volume** field on the IMMDigitalVideo* part's settings page. Values range from 0 for no volume to 100 for the maximum volume.

If you are providing an IMMMasterAudio* part for volume control and are not providing a volume dial for the video player, set the **volume** field value to 100.

Changing the Volume

To let the user change the video player volume, provide a volume dial as follows:

- Place an ICircularSlider* part on a canvas.
- Define the circular slider as a volume control with values from 0 to 100.
- Connect the ICircularSlider* part's *value* attribute to the IMMDigitalVideo* part's *volume* attribute.

If you are providing volume control with an IMMMasterAudio* part as well as with the video player, the master audio volume setting determines the maximum volume produced. The player volume is the product of the master audio volume setting and the player volume setting. For example, if the master audio volume is set at 50 and the player volume is set at 80, the player volume is 40 percent of its potential maximum.

Starting Player Actions

To let the user start video player actions, provide animated buttons for the actions. For example, provide a play button as follows:

- Place an IAnimatedButton* part on a canvas.
- Define the animated button as a latchable play button.
- Connect the IAnimatedButton* part's *buttonClickEvent* feature to the IMMDigitalVideo* part's *play* action.

Preferred Features — IMMDigitalVideo

close	Closes a device.
copy	Copies data from the passed in start position to the passed in end position into the clipboard.
cut	Cuts the data from the passed in start position to the passed in end position into the clipboard.
filename	Returns the currently loaded filename.
open	Opens the device or file based on the passed in string.
paste	Replaces the data with data from the clipboard, from the passed in start position to the passed in end position.
pause	If the device is playing, then it pauses the device.
play	Starts playing the device from the passed in start position to the passed in end position.
position	Returns the current position in the current time format as an `unsigned long`.
record	Starts recording.
seekToEnd	Moves the current position to the end of the data in the device.
seekToStart	Moves the current position to the start of the data in the device.
stepFrame	Steps the play one or more time units forward or backward.
stop	Stops playback of the device.
this	A Pointer to the instance.

Features — IMMDigitalVideo

Action	Attribute	Event
acquire ■	acquired	anyEvent
close ■	aliasName	commandNotifyEvent
connectedDeviceId ■	asDebugInfo	cuePointEvent
copy ■	asString	deviceEvent
cueForPlayback ■	audioEnabled ■	passDeviceEvent
cueForRecording ■	bitsPerSample ■	positionChangeEvent ■
cut ■	blockAlignment ■	
deletePendingEvents ■	bytesPerSecond ■	
deleteSelection ■	channels ■	

IMMDigitalVideo

Action	Attribute	Event
disableAudio ■	closeOnDestroy ■	
disableConnector ■	description	
disableNotification	**destinationRectangle** ■	
enableConnector ■	deviceId	
isConnectionSupported ■	deviceName	
isConnectorEnabled ■	deviceType	
loadOnThread ■	enabledForNotification ■	
notifyObservers ■	filename ■	
open ■	**fileNormalSpeed**	
openOnThread ■	format ■	
operator = ■	**handle**	
paste ■	isOpen	
pause ■	length	
play ■	**maximumSpeed**	
playAt ■	**maximumWindows**	
playFast ■	**minimumSpeed**	
playScan ■	mode	
playSlow ■	**normalSpeed**	
record ■	**playingForward**	
release ■	position	
resume ■	readOnly	
save ■	requiresFiles	
saveAs ■	samplesPerSecond ■	
seek ■	**sourceRectangle**	
seekToEnd ■	**speed**	
seekToStart ■	speedFormat ■	
setWindow ■	supportsAudio	
startPositionTracking ■	supportsDigitalTransfer	
stepFrame ■	supportsDisableEject	
stop ■	supportsEject	
stopPositionTracking ■	**supportsOverlayGraphics**	
supportsCommand ■	supportsPlay	
useDefaultWindow ■	supportsRecord	
	supportsRecordInsertion	
	supportsReverse	
	supportsSave	
	supportsSizing	
	supportsStreaming	
	supportsStretchToFit	
	supportsVideo	
	supportsVolumeAdjustment	
	this	
	timeFormat ■	
	videoFileHeight	
	videoFileName	
	videoFileWidth	
	videoHeight	

IMMDigitalVideo

Action	Attribute	Event
	videoWidth	
	volume ■	

Chapter 132. IMMErrorInfo

Description: IBM multimedia error information
Part type: Class
Part file: vbmm
Header file: immexcpt.hpp

Derivation:
IBase
IVBase
IErrorInfo
IMMErrorInfo

Preferred Features — IMMErrorInfo

text	Returns the error text.
this	A Pointer to the instance.

Features — IMMErrorInfo

Action	Attribute	Type
operator const char *	asDebugInfo	`IString`
throwError ■	asString	`IString`
throwMMError ■	**available**	`Boolean`
	errorId	`unsigned long`
	text	`const char*`
	this	`IMMErrorInfo*`

© Copyright IBM Corp. 1995, 1996

IMMErrorInfo

Chapter 133. IMMHourMinSecFrameTime

Description: IBM support of hour/minute/second/frame time format
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMHourMinSecFrameTime

Preferred Features — IMMHourMinSecFrameTime

frames	Returns the frames component of the time.
minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMMHourMinSecFrameTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	**frames**	`unsigned long`
operator -= ■	**framesPerSecond**	`unsigned long`
operator < ■	hours	`unsigned long`
operator <= ■	hundredths	`unsigned long`
operator = ■	minutes	`unsigned long`
operator == ■	ordinal ■	`unsigned long`
operator > ■	seconds	`unsigned long`
operator >= ■	**this**	`IMMHourMinSecFrameTime*`
operator unsigned long	thousandths	`unsigned long`
	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMMHourMinSecFrameTime

Chapter 134. IMMHourMinSecTime

Description: IBM support of hour/minute/second time format
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMHourMinSecTime

Preferred Features — IMMHourMinSecTime

minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMMHourMinSecTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	hours	`unsigned long`
operator -= ■	hundredths	`unsigned long`
operator < ■	minutes	`unsigned long`
operator <= ■	ordinal ■	`unsigned long`
operator = ■	seconds	`unsigned long`
operator == ■	**this**	`IMMHourMinSecTime*`
operator > ■	thousandths	`unsigned long`
operator >= ■	valid	`Boolean`
operator unsigned long		

© Copyright IBM Corp. 1995, 1996

IMMHourMinSecTime

Chapter 135. IMMMasterAudio

Description: IBM master audio
Part type: Nonvisual
Part file: vbmm
Header file: immmaud.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IMMMasterAudio

Use an IMMMasterAudio* part to provide a master volume control. Your application can also use the control to enable and disable speakers and headphones. Be aware that a master audio control affects all sound output for the system.

Note: To use an IMMMasterAudio* part, ensure that you have installed multimedia support on your system.

The user can operate the master volume through a visual control you provide. A circular slider is well suited for visual control of volume.

Related Parts — IMMMasterAudio

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio"
- "IMMPlayerPanel" on page 529

Multimedia player parts:

- "IMMAudioCD" on page 501
- "IMMDigitalVideo" on page 509
- "IMMSequencer" on page 533
- "IMMWaveAudio" on page 543

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IMMMasterAudio

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Setting the Initial Volume

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Volume
- Starting Player Actions

Adding the Part

To add an IMMMasterAudio* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Setting the Initial Volume

To define an initial volume setting, enter a value in the **volume** field on the IMMMasterAudio* part's settings page. Values range from 0 for no volume to 100 for the maximum volume.

Changing the Volume

To let the user change the master audio volume, provide a volume dial as follows:

- Place an ICircularSlider* part on a canvas.
- Define the circular slider as a volume control with values from 0 to 100.
- Connect the ICircularSlider* part's *value* attribute to the IMMMasterAudio* part's *volume* attribute.

Starting Player Actions

To let the user start master audio actions, provide animated buttons for the actions. For example, provide a mute button as follows:

- Place an IAnimatedButton* part on a canvas.
- Define the animated button as a latchable mute button.
- Connect the IAnimatedButton* part's *buttonClickEvent* feature to IMMMasterAudio* part's **Custom logic**. Enter C++ code to mute and unmute the sound.

For example, you could enter the following code:

```
unsigned long volumeBeforeMute = 50;    // Set an initial previous volume
if (target->volume() == 0)              // Turn sound on if it is off
{
  target->setVolume(volumeBeforeMute);
}
else                                    // Turn sound off if it is on
{
  volumeBeforeMute = target->volume();
  target->setVolume(0);
}
```

Preferred Features — IMMMasterAudio

areHeadphonesEnabled
: Returns if the headphones' setting is enabled for the passed in source (either the current setting or the saved setting); otherwise, returns `false`.

areSpeakersEnabled
: Returns the `true` if the speakers' setting is enabled for the passed in source (either the current setting or the saved setting); otherwise, returns `false`.

this
: A Pointer to the instance.

volume
: Returns the master volume setting for either the current setting or the saved setting to a percent of the maximum audio level.

Features — IMMMasterAudio

Action	Attribute	Event
disableHeadphones ■	**areHeadphonesEnabled**	anyEvent
disableNotification	**areSpeakersEnabled**	
disableSpeakers ■	asDebugInfo	
enableHeadphones ■	asString	
enableSpeakers ■	enabledForNotification ■	
notifyObservers ■	**this**	
operator = ■	**volume** ■	
saveHeadphonesSetting ■		
saveSpeakersSetting ■		
saveVolume ■		

IMMMasterAudio

Chapter 136. IMMMillisecondTime

Description: IBM millisecond time class
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMMillisecondTime

Preferred Features — IMMMillisecondTime

minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMMMillisecondTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	hours	`unsigned long`
operator -= ■	hundredths	`unsigned long`
operator < ■	minutes	`unsigned long`
operator <= ■	ordinal ■	`unsigned long`
operator = ■	seconds	`unsigned long`
operator == ■	**this**	`IMMMillisecondTime*`
operator > ■	thousandths	`unsigned long`
operator >= ■	valid	`Boolean`
operator unsigned long		

© Copyright IBM Corp. 1995, 1996

IMMMillisecondTime

Chapter 137. IMMMinSecFrameTime

Description: IBM support of minute/second/frame time format
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMMinSecFrameTime

Preferred Features — IMMMinSecFrameTime

frames	Returns the frames component of the time.
minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMMMinSecFrameTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	**frames**	`unsigned long`
operator -= ■	hours	`unsigned long`
operator < ■	hundredths	`unsigned long`
operator <= ■	minutes	`unsigned long`
operator = ■	ordinal ■	`unsigned long`
operator == ■	seconds	`unsigned long`
operator > ■	**this**	`IMMMinSecFrameTime*`
operator >= ■	thousandths	`unsigned long`
operator unsigned long	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMMMinSecFrameTime

Chapter 138. IMMPlayerPanel

Description: IBM player panel
Part type: Visual
Part file: vbmm
Header file: immplypn.hpp

Derivation:
IBase
IVBase
INotifier
IWindow
IControl
ICanvas
IMultiCellCanvas
IMMPlayerPanel

Use an IMMPlayerPanel* part for easy composition and control of audio and video players.

Note: To use an IMMPlayerPanel* part, ensure that you have installed multimedia support on your system.

The user can operate an audio or video player through visual controls in the player panel.

Related Parts — IMMPlayerPanel

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel"

Multimedia player parts:

- "IMMAudioCD" on page 501
- "IMMDigitalVideo" on page 509
- "IMMSequencer" on page 533
- "IMMWaveAudio" on page 543

© Copyright IBM Corp. 1995, 1996

Building Guidelines — IMMPlayerPanel

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Adding the Player Panel to a Player

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Starting Player Actions
- Changing Related Buttons

Adding the Part

To add an IMMPlayerPanel* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Adding the Player Panel to a Player

To add the player panel to a player, specify the player part as the playable device for the panel.

For example, to use the player panel with an audio CD player, connect the IMMAudioCD* part's *this* attribute to the IMMPlayerPanel* part's *playableDevice* attribute.

Starting Player Actions

The IMMPlayerPanel* part automatically connects player panel buttons to corresponding actions in the player part you add the panel to.

For example, the play button is automatically connected to the *play* action of the player part. When the user presses the play button, the *play* action is started.

Changing Related Buttons

The IMMPlayerPanel* part automatically changes the state of related buttons when the user presses player panel buttons.

For example, if the user presses the stop button, the stop and pause buttons are disabled.

As another example, if the user presses the pause button while the device is playing, the pause button is latched down and the play button is unlatched.

Preferred Features — IMMPlayerPanel

menu	Popup menu link.
position	Returns the position of the window.
setFocus	Sets the input focus to the window.
this	A Pointer to the instance.

Features — IMMPlayerPanel

Action	Attribute	Event
addToCell ■	activeColor ■ ►	*activeColor*
applyBidiSettings ■	asDebugInfo	anyEvent
capturePointer ■	asString	*backgroundColor*
columnWidth ■	autoDeleteObject ■	*borderColor*
convertToGUIStyle ■	autoDestroyWindow ■	commandEvent
disable	backgroundColor ■ ►	*disabledBackgroundColor*
disableDragLines	bidiSupported	*disabledForegroundColor*
disableGridLines	borderColor ■ ►	*enabled*
disableGroup	characterSize	*focus*
disableNotification	defaultCell ■	*font*
disableTabStop	defaultPushButton	*foregroundColor*
disableUpdate	**deviceType**	gotFocusEvent ■
dispatchRemainingHandlers ■	disabledBackgroundColor ■ ►	*hiliteBackgroundColor*
enable ■	disabledForegroundColor ■ ►	*hiliteForegroundColor*
enableUpdate ■	dragLines ■	*inactiveColor*
handleException ■	enabled ■ ►	inputDisabledEvent ■
hide	enabledForNotification ■	inputEnabledEvent ■
hideSourceEmphasis	**fastForwardButton**	lostFocusEvent ■
isColumnExpandable ■	focus ►	*position*
isLayoutDistorted ■	font ■ ►	*shadowColor*
isRowExpandable ■	foregroundColor ■ ►	*size*
matchForMnemonic ■	frameWindow	systemCommandEvent
notifyObservers ■	gridLines ■	visibilityDisabledEvent ■
positionBehindSibling ■	group ■	visibilityEnabledEvent ■
positionBehindSiblings	handle	*visible*
positionOnSiblings	helpId ■	
postEvent ■	hiliteBackgroundColor ■ ►	
refresh ■	hiliteForegroundColor ■ ►	
releasePointer	id ■	
releasePresSpace ■	inactiveColor ■ ►	
removeFromCell ■	itemProvider ■	
resetActiveColor	layoutAdjustment	
resetBackgroundColor	menu	

IMMPlayerPanel

Action	Attribute	Event
resetBorderColor	minimumSize ■	
resetDisabledBackgroundColor	nativeRect	
resetDisabledForegroundColor	origDefaultButtonHandle	
resetFont	owner ■	
resetForegroundColor	parent ■	
resetHiliteBackgroundColor	parentSize	
resetHiliteForegroundColor	**pauseButton**	
resetInactiveColor	**playableDevice** ■	
resetMinimumSize	**playButton**	
resetShadowColor	pointerCaptured	
rowHeight ■	position ■ ►	
sendEvent ■	presSpace	
setColumnWidth ■	rect ■	
setFocus	**rewindButton**	
setLayoutDistorted ■	shadowColor ■ ►	
setRowHeight ■	showing	
show ■	size ■ ►	
showSourceEmphasis ■	**stepBackwardButton**	
windowInCell ■	**stepForwardButton**	
	stopButton	
	tabStop ■	
	this	
	valid	
	visible ■ ►	
	visibleRectangle	

Chapter 139. IMMSequencer

Description: IBM sequencer
Part type: Nonvisual
Part file: vbmm
Header file: immsequ.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IMMDevice
IMMPlayableDevice
IMMFileMedia
IMMSequencer

Use an IMMSequencer* part to provide standard audio player functions for Musical Instrument Digital Interface (MIDI) files.

Note: To use an IMMSequencer* part, ensure that you have installed multimedia support on your system.

The user can operate the sequencer through visual controls. You can easily provide button controls with an IMMPlayerPanel* part. You can provide volume control for all audio devices with an IMMMasterAudio* part.

Portability — IMMSequencer

Features not defined for Windows:

- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

Related Parts — IMMSequencer

Multimedia player parts:

- "IMMAudioCD" on page 501
- "IMMDigitalVideo" on page 509
- "IMMSequencer"
- "IMMWaveAudio" on page 543

© Copyright IBM Corp. 1995, 1996

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — IMMSequencer

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting Formats
- Setting the Initial Volume

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Volume
- Starting Player Actions

Adding the Part

To add an IMMSequencer* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.
- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Selecting Formats

To specify speed and time formats rather than using system-provided formats, do the following:

- Select a speed format in the **speedFormat** field on the IMMSequencer* part's settings page.
- Select a time format in the **timeFormat** field on the IMMSequencer* part's settings page.

Setting the Initial Volume

To define an initial volume setting, enter a value in the **volume** field on the IMMSequencer* part's settings page. Values range from 0 for no volume to 100 for the maximum volume.

If you are providing an IMMMasterAudio* part for volume control and are not providing a volume dial for the sequencer, set the **volume** field value to 100.

Changing the Volume

To let the user change the sequencer volume, provide a volume dial as follows:

- Place an ICircularSlider* part on a canvas.
- Define the circular slider as a volume control with values from 0 to 100.
- Connect the ICircularSlider* part's *value* attribute to the IMMSequencer* part's *volume* attribute.

If you are providing volume control with an IMMMasterAudio* part as well as with the sequencer, the master audio volume setting determines the maximum volume produced. The player volume is the product of the master audio volume setting and the player volume setting. For example, if the master audio volume is set at 50 and the player volume is set at 80, the player volume is 40 percent of its potential maximum.

Starting Player Actions

To let the user start sequencer actions, provide animated buttons for the actions. For example, provide a play button as follows:

- Place an IAnimatedButton* part on a canvas.
- Define the animated button as a latchable play button.
- Connect the IAnimatedButton* part's *buttonClickEvent* feature to the IMMSequencer* part's *play* action.

Preferred Features — IMMSequencer

close	Closes a device.
filename	Returns the currently loaded filename.
open	Opens the device or file based on the passed in string.
pause	If the device is playing, then it pauses the device.
play	Starts playing the device from the passed in start position to the passed in end position.
position	Returns the current position in the current time format as an `unsigned long`.
seekToEnd	Moves the current position to the end of the data in the device.
seekToStart	Moves the current position to the start of the data in the device.

stepFrame	Steps the play one or more time units forward or backward.
stop	Stops playback of the device.
this	A Pointer to the instance.

Features — IMMSequencer

Action	Attribute	Event
acquire ■	acquired	anyEvent
close ■	aliasName	commandNotifyEvent
connectedDeviceId ■	asDebugInfo	cuePointEvent
cueForPlayback ■	asString	deviceEvent
deletePendingEvents ■	audioEnabled ■	passDeviceEvent
disableAudio ■	closeOnDestroy ■	positionChangeEvent ■
disableConnector ■	description	
disableNotification	deviceId	
enableConnector ■	deviceName	
isConnectionSupported ■	deviceType	
isConnectorEnabled ■	enabledForNotification ■	
loadOnThread ■	filename ■	
notifyObservers ■	isOpen	
open ■	length	
openOnThread ■	mode	
operator = ■	position	
pause ■	readOnly	
play ■	requiresFiles	
release ■	speedFormat ■	
resume ■	supportsAudio	
seek ■	supportsDigitalTransfer	
seekToEnd ■	supportsDisableEject	
seekToStart ■	supportsEject	
startPositionTracking ■	supportsPlay	
stepFrame ■	supportsRecord	
stop ■	supportsRecordInsertion	
stopPositionTracking ■	supportsSave	
supportsCommand ■	supportsStreaming	
	supportsVideo	
	supportsVolumeAdjustment	
	this	
	timeFormat ■	
	volume ■	

Chapter 140. IMMSpeed

Description: IBM speed class
Part type: Class
Part file: vbmm
Header file: immspeed.hpp

Derivation:
IBase
IVBase
IMMSpeed

Preferred Features — IMMSpeed

speed	Returns the speed value as either a percentage or in frames-per-second.
this	A Pointer to the instance.

Features — IMMSpeed

Action	Attribute	Type
	asDebugInfo	`IString`
	asString	`IString`
	format	`IMMSpeed::Format`
	speed	`unsigned long`
	this	`IMMSpeed*`

© Copyright IBM Corp. 1995, 1996

IMMSpeed

Chapter 141. IMMTime

Description: IBM base class for all multimedia time support
Part type: Class
Part file: vbmm
Header file: immtime.hpp

Derivation:
IBase
IVBase
IMMTime

Preferred Features — IMMTime

minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.

Features — IMMTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	**asMMTime**	`unsigned long`
operator += ■	asString	`IString`
operator - ■	**hours**	`unsigned long`
operator -= ■	**hundredths**	`unsigned long`
operator < ■	**minutes**	`unsigned long`
operator <= ■	**ordinal ■**	`unsigned long`
operator = ■	**seconds**	`unsigned long`
operator == ■	**this**	`IMMTime*`
operator > ■	**thousandths**	`unsigned long`
operator >= ■	**valid**	`Boolean`
operator unsigned long		

© Copyright IBM Corp. 1995, 1996

IMMTime

Chapter 142. IMMTrackMinSecFrameTime

Description: IBM support of track/minute/second/frame time format
Part type: Class
Part file: vbmm
Header file: immttime.hpp

Derivation:
IBase
IVBase
IMMTime
IMMTrackMinSecFrameTime

Preferred Features — IMMTrackMinSecFrameTime

frames	Returns the frames component of the time.
minutes	Returns the minutes component of the time.
seconds	Returns the seconds component of the time.
this	A Pointer to the instance.
track	Returns the track component of the time.

Features — IMMTrackMinSecFrameTime

Action	Attribute	Type
operator != ■	asDebugInfo	`IString`
operator + ■	asMMTime	`unsigned long`
operator += ■	asString	`IString`
operator - ■	**frames**	`unsigned long`
operator -= ■	hours	`unsigned long`
operator < ■	hundredths	`unsigned long`
operator <= ■	minutes	`unsigned long`
operator = ■	ordinal ■	`unsigned long`
operator == ■	seconds	`unsigned long`
operator > ■	**this**	`IMMTrackMinSecFrameTime*`
operator >= ■	thousandths	`unsigned long`
operator unsigned long	**track**	`unsigned long`
	valid	`Boolean`

© Copyright IBM Corp. 1995, 1996

IMMTrackMinSecFrameTime

Chapter 143. IMMWaveAudio

Description: IBM wave audio
Part type: Nonvisual
Part file: vbmm
Header file: immwave.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IMMDevice
IMMPlayableDevice
IMMFileMedia
IMMRecordable
IMMConfigurableAudio
IMMWaveAudio

Use an IMMWaveAudio* part to provide standard audio player functions for waveform files.

Note: To use an IMMWaveAudio* part, ensure that you have installed multimedia support on your system.

The user can operate the sequencer through visual controls. You can easily provide button controls with an IMMPlayerPanel* part. You can provide volume control for all audio devices with an IMMMasterAudio* part.

Portability — IMMWaveAudio

Features not defined for Windows:

- cutCopyBufferSize
- copyFromBuffer
- copyToBuffer
- cutToBuffer
- pasteFromBuffer
- pasteToBuffer
- supportsWaveFormat
- canRedo
- canUndo
- redo

© Copyright IBM Corp. 1995, 1996

- undo
- addCuePoint
- removeCuePoint
- prerollTime
- prerollType

Related Parts — IMMWaveAudio

Multimedia player parts:

- "IMMAudioCD" on page 501
- "IMMDigitalVideo" on page 509
- "IMMSequencer" on page 533
- "IMMWaveAudio" on page 543

Multimedia control parts:

- "IAnimatedButton" on page 73
- "ICircularSlider" on page 95
- "ICustomButton" on page 137
- "IMMAmpMixer" on page 495
- "IMMMasterAudio" on page 521
- "IMMPlayerPanel" on page 529

Building Guidelines — IMMWaveAudio

Begin by placing the part in the Composition Editor and defining initial characteristics of the part. The following topics explain some common initialization tasks:

- Adding the Part
- Selecting Formats
- Defining Audio Channels
- Setting the Initial Volume

You might need to have the part respond to user input and application changes. The following topics explain some common change tasks:

- Changing the Volume
- Starting Player Actions

Adding the Part

To add an IMMWaveAudio* part to a composite part, do the following:

- Select **Add part** on the Composition Editor's **Options** menu.

- In the add part window, specify the class name in the **Part class** field and select **Add**.
- Drop the part on the free-form surface.

Selecting Formats

To specify audio, speed, and time formats rather than using system-provided formats, do the following:

- Select an audio format in the **format** field on the IMMWaveAudio* part's settings page.
- Select a speed format in the **speedFormat** field on the IMMWaveAudio* part's settings page.
- Select a time format in the **timeFormat** field on the IMMWaveAudio* part's settings page.

Defining Audio Channels

To define the number of audio output channels for the waveform player, enter a number in the **channels** field on the IMMWaveAudio* part's settings page. Enter 1 for monophonic (mono) output or 2 for stereophonic (stereo) output.

Setting the Initial Volume

To define an initial volume setting, enter a value in the **volume** field on the IMMWaveAudio* part's settings page. Values range from 0 for no volume to 100 for the maximum volume.

If you are providing an IMMMasterAudio* part for volume control and are not providing a volume dial for the waveform player, set the **volume** field value to 100.

Changing the Volume

To let the user change the waveform player volume, provide a volume dial as follows:

- Place an ICircularSlider* part on a canvas.
- Define the circular slider as a volume control with values from 0 to 100.
- Connect the ICircularSlider* part's *value* attribute to the IMMWaveAudio* part's *volume* attribute.

If you are providing volume control with an IMMMasterAudio* part as well as with the waveform player, the master audio volume setting determines the maximum volume produced. The player volume is the product of the master audio volume setting and the player volume setting. For example, if the master audio volume is set at 50 and the player volume is set at 80, the player volume is 40 percent of its potential maximum.

Starting Player Actions

To let the user start waveform player actions, provide animated buttons for the actions. For example, provide a play button as follows:

- Place an IAnimatedButton* part on a canvas.
- Define the animated button as a latchable play button.
- Connect the IAnimatedButton* part's *buttonClickEvent* feature to the IMMWaveAudio* part's *play* action.

Preferred Features — IMMWaveAudio

close	Closes a device.
copy	Copies data from the passed in start position to the passed in end position into the clipboard.
cut	Cuts the data from the passed in start position to the passed in end position into the clipboard.
filename	Returns the currently loaded filename.
open	Opens the device or file based on the passed in string.
paste	Replaces the data with data from the clipboard, from the passed in start position to the passed in end position.
pause	If the device is playing, then it pauses the device.
play	Starts playing the device from the passed in start position to the passed in end position.
position	Returns the current position in the current time format as an `unsigned long`.
record	Starts recording at the begin location till it reaches the end location.
seekToEnd	Moves the current position to the end of the data in the device.
seekToStart	Moves the current position to the start of the data in the device.
stepFrame	Steps the play one or more time units forward or backward.
stop	Stops playback of the device.
this	A Pointer to the instance.

Features — IMMWaveAudio

Action	Attribute	Event
acquire ■	acquired	anyEvent
close ■	aliasName	commandNotifyEvent
connectedDeviceId ■	asDebugInfo	cuePointEvent
copy ■	asString	deviceEvent
cueForPlayback ■	audioEnabled ■	passDeviceEvent
cueForRecording ■	bitsPerSample ■	positionChangeEvent ■
cut ■	blockAlignment ■	
deletePendingEvents ■	bytesPerSecond ■	
deleteSelection ■	channels ■	

IMMWaveAudio

Action	Attribute	Event
disableAudio ■	closeOnDestroy ■	
disableConnector ■	description	
disableNotification	deviceId	
enableConnector ■	deviceName	
isConnectionSupported ■	deviceType	
isConnectorEnabled ■	enabledForNotification ■	
loadOnThread ■	filename ■	
notifyObservers ■	format ■	
open ■	isOpen	
openOnThread ■	length	
operator = ■	mode	
paste ■	position	
pause ■	readOnly	
play ■	requiresFiles	
record ■	samplesPerSecond ■	
release ■	speedFormat ■	
resume ■	supportsAudio	
save ■	supportsDigitalTransfer	
saveAs ■	supportsDisableEject	
seek ■	supportsEject	
seekToEnd ■	supportsPlay	
seekToStart ■	supportsRecord	
startPositionTracking ■	supportsRecordInsertion	
stepFrame ■	supportsSave	
stop ■	supportsStreaming	
stopPositionTracking ■	supportsVideo	
supportsCommand ■	supportsVolumeAdjustment	
	this	
	timeFormat ■	
	volume ■	

IMMWaveAudio

Part 5. Sample Parts in vbsample.vbb

The following chapters provide information on sample business and utility parts that you can modify to create your own parts. These parts are provided without service support.

To use these parts, load the vbsample.vbb part file from the ibmcppw\ivb directory.

Chapter 144. floatSamples 553
Preferred Features — floatSamples 553
Features — floatSamples 553

Chapter 145. IAddress 555
Preferred Features — IAddress 555
Features — IAddress 555

Chapter 146. ICompany 557
Preferred Features — ICompany 557
Features — ICompany 557

Chapter 147. ICustomer 559
Preferred Features — ICustomer 559
Features — ICustomer 559

Chapter 148. IOrderedRecord 561
Preferred Features — IOrderedRecord 561
Features — IOrderedRecord 561

Chapter 149. ioSamples 563
Preferred Features — ioSamples 563
Features — ioSamples 563

Chapter 150. IRecord 565
Preferred Features — IRecord 565
Features — IRecord 565

Chapter 151. IVBBooleanPart 567
Preferred Features — IVBBooleanPart 567
Features — IVBBooleanPart 567

Chapter 152. IVBDoublePart 569

© Copyright IBM Corp. 1995, 1996

Sample Parts in vbsample.vbb

Preferred Features — IVBDoublePart 569
Features — IVBDoublePart 569

Chapter 153. IVBLogicalAndPart 571
Preferred Features — IVBLogicalAndPart 571
Features — IVBLogicalAndPart 571

Chapter 154. IVBLogicalOrPart 573
Preferred Features — IVBLogicalOrPart 573
Features — IVBLogicalOrPart 573

Chapter 155. IVBLongPart 575
Preferred Features — IVBLongPart 575
Features — IVBLongPart 575

Chapter 156. IVBShortPart 577
Preferred Features — IVBShortPart 577
Features — IVBShortPart 577

Chapter 157. IVBStringPart 579
Preferred Features — IVBStringPart 579
Features — IVBStringPart 579

Chapter 158. IVBUnsignedLongPart 581
Preferred Features — IVBUnsignedLongPart 581
Features — IVBUnsignedLongPart 581

Chapter 159. IVBUnsignedShortPart 583
Preferred Features — IVBUnsignedShortPart 583
Features — IVBUnsignedShortPart 583

Chapter 160. mathSamples 585
Preferred Features — mathSamples 585
Features — mathSamples 585

Chapter 161. staticWindowSamples 587
Preferred Features — staticWindowSamples 587
Features — staticWindowSamples 587

Chapter 162. stdioSamples 589
Preferred Features — stdioSamples 589
Features — stdioSamples 589

Sample Parts in vbsample.vbb

Chapter 163. stdlibSamples 591
Preferred Features — stdlibSamples 591
Features — stdlibSamples 591

Sample Parts in vbsample.vbb

Chapter 144. floatSamples

Description: float c library sample functions
Part type: Function group
Part file: vbsample
Header file: float.h

Derivation:
floatSamples

Preferred Features — floatSamples

_clear87	_clear87 gets the floating-point status word and then clears it.
_control87	_control87 gets the current floating-point control word and then sets it.
_fpreset	_fpreset resets the floating-point unit to the default state that the math library requires to function correctly.
_status87	_status87 gets the current floating-point status word.

Features — floatSamples

Action	Function
_clear87	`unsigned int _clear87 ();`
_control87 ■	`unsigned int _control87 (unsigned int new, unsigned int mask);`
_fpreset	`void _fpreset ();`
_status87	`unsigned int _status87 ();`

© Copyright IBM Corp. 1995, 1996

floatSamples

Chapter 145. IAddress

Description: IBM sample address part
Part type: Nonvisual
Part file: vbsample
Header file: iadd.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IAddress

Preferred Features — IAddress

city	Query the city (`IString`) attribute.
setToDefault	Perform the setToDefault action.
state	Query the state (`IString`) attribute.
street	Query the street (`IString`) attribute.
this	A Pointer to the instance.
zip	Query the zip (`IString`) attribute.

Features — IAddress

Action	Attribute	Event
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	***city***
operator != ■	**city** ■ ►	***state***
operator = ■	enabledForNotification ■	***street***
operator == ■	**state** ■ ►	***zip***
setCityToDefault	**street** ■ ►	
setStateToDefault	**this**	
setStreetToDefault	**zip** ■ ►	
setToDefault		
setZipToDefault		

© Copyright IBM Corp. 1995, 1996

IAddress

Chapter 146. ICompany

Description: IBM sample company part
Part type: Nonvisual
Part file: vbsample
Header file: icompany.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
ICompany

Preferred Features — ICompany

addCustomer	Perform the add customer action.
address	Query the address (`IAddress*`) attribute.
customerAddedEvent	Notification event for when customer added to customer list.
customerList	Query the customerList (`IVSequence <ICustomer*> *`) attribute.
name	Query the name (`IString`) attribute.
phone	Query the phone (`IString`) attribute.
this	A Pointer to the instance.

Features — ICompany

Action	Attribute	Event
addCustomer ■	**address** ■ ►	***address***
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	**customerAddedEvent**
operator != ■	**customerList** ■ ►	***customerList***
operator = ■	enabledForNotification ■	***name***
operator == ■	**name** ■ ►	***phone***
setAddressToDefault	**phone** ■ ►	
setCustomerListToDefault	**this**	
setNameToDefault		
setPhoneToDefault		

© Copyright IBM Corp. 1995, 1996

ICompany

Chapter 147. ICustomer

Description: IBM sample customer part
Part type: Nonvisual
Part file: vbsample
Header file: icust.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
ICustomer

Preferred Features — ICustomer

address	Query the address (`IAddress*`) attribute.
date	Query the date (`IDate`) attribute.
homePhone	Query the homePhone (`IString`) attribute.
name	Query the name (`IString`) attribute.
setNameToDefault	Perform the setNameToDefault action.
this	A Pointer to the instance.
time	Query the time (`ITime`) attribute.
workPhone	Query the workPhone (`IString`) attribute.

Features — ICustomer

Action	Attribute	Event
disableNotification	**address** ■ ►	***address***
notifyObservers ■	asDebugInfo	anyEvent
operator != ■	asString	***date***
operator = ■	**date** ■ ►	***homePhone***
operator == ■	enabledForNotification ■	***name***
setAddressToDefault	**homePhone** ■ ►	***time***
setHomePhoneToDefault	**name** ■ ►	***workPhone***
setNameToDefault	**this**	
setWorkPhoneToDefault	**time** ■ ►	
	workPhone ■ ►	

© Copyright IBM Corp. 1995, 1996

ICustomer

Chapter 148. IOrderedRecord

Description: IBM ordered record
Part type: Class
Part file: vbsample
Header file: iordrrec.hpp

Derivation:
IBase
IVBase
IRecord
IOrderedRecord

Preferred Features — IOrderedRecord

this A Pointer to the instance.

Features — IOrderedRecord

Action	Attribute	Type
operator != ∎	asDebugInfo	`IString`
operator < ∎	asString	`IString`
operator <= ∎	ordered	`IBoolean`
operator = ∎	size	`unsigned long`
operator == ∎	**this**	`IOrderedRecord*`
operator > ∎		
operator >= ∎		

© Copyright IBM Corp. 1995, 1996

IOrderedRecord

Chapter 149. ioSamples

Description: io c library sample functions
Part type: Function group
Part file: vbsample
Header file: io.h

Derivation:
ioSamples

Preferred Features — ioSamples

access access determines whether the specified file exists and whether you can get access to it in the given mode.

close close closes the file associated with the handle.

open open opens the file specified by pathname and prepares the file for subsequent reading or writing as defined by oflag.

read read reads count bytes from the file associated with handle into buffer.

write write writes count bytes from buffer into the file associated with handle.

Features — ioSamples

Action	Function
access ■	`int access (char* pathname, int mode);`
chmod ■	`int chmod (char* pathname, int pmode);`
close ■	`int close (int handle);`
creat ■	`int creat (char* pathname, int pmode);`
dup ■	`int dup (int handle);`
dup2 ■	`int dup2 (int handle1, int handle2);`
isatty ■	`int isatty (int handle);`
lseek ■	`long lseek (int handle, long offset, int origin);`
open ■	`int open (char* pathname, int oflag, int pmode);`
read ■	`int read (int handle, char* buffer, unsigned int count);`
umask ■	`int umask (int pmode);`
write ■	`int write (int handle, const void* buffer, unsigned int count);`
_chsize ■	`int _chsize (int handle, long size);`
_filelength ■	`long _filelength (int handle);`
_setmode ■	`int _setmode (int handle, int mode);`
_sopen ■	`int _sopen (char* pathname, int oflag, int shflag, int pmode);`
_tell ■	`long _tell (int handle);`
__eof ■	`int __eof (int handle);`

© Copyright IBM Corp. 1995, 1996

ioSamples

Chapter 150. IRecord

Description: IBM record
Part type: Class
Part file: vbsample
Header file: irecord.hpp

Derivation:
IBase
IVBase
IRecord

Preferred Features — IRecord

ordered	Returns `false`.
size	Return the size attribute.
this	A Pointer to the instance.

Features — IRecord

Action	Attribute	Type
operator = ∎	asDebugInfo	`IString`
	asString	`IString`
	ordered	`IBoolean`
	size	`unsigned long`
	this	`IRecord*`

© Copyright IBM Corp. 1995, 1996

IRecord

Chapter 151. IVBBooleanPart

Description: IBM VB part support for Boolean data
Part type: Nonvisual
Part file: vbsample
Header file: ivbbool.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBBooleanPart

Preferred Features — IVBBooleanPart

assignValueToDefault
Assign the value attribute to `false`.
assignValueToFalse
Assign the value attribute to `false`.
assignValueToTrue
Assign the value attribute to `true`.
notValue Query the notValue (`Boolean`) attribute.
this A Pointer to the instance.
value Query the value (`Boolean`) attribute.
valueAsText Query the valueAsText (`IString`) attribute.
valueFalseEvent Notification event for value is `false`.
valueTrueEvent Notification event for value is `true`.

Features — IVBBooleanPart

Action	Attribute	Event
assignValueToDefault	asDebugInfo	anyEvent
assignValueToFalse	asString	***defaultValue***
assignValueToTrue	**defaultValue** ■ ►	digitsEvent
copyValueToDefault	enabledForNotification ■	inputStringIsValid
disableNotification	**notValue** ■ ►	inputStringNotValid
logicalAndValue ■	**notValueAsText** ■ ►	lowerCaseEvent
logicalNotValue	**this**	***notValue***
logicalOrValue ■	**value** ■ ►	***notValueAsText***
notifyObservers ■	**valueAsText** ■ ►	textEqualDefaultEvent
operator != ■	**valueEqualDefault** ►	textLengthEvent
operator = ■	**valueNotEqualDefault** ►	textNotEqualDefaultEvent
operator == ■		upperCaseEvent
		value

© Copyright IBM Corp. 1995, 1996

IVBBooleanPart

Action	Attribute	Event
		valueAboveHighLimitEvent
		valueAsText
		valueBelowLowLimitEvent
		valueEqualDefault
		valueEqualDefaultEvent
		valueEqualHighLimitEvent
		valueEqualLowLimitEvent
		valueFalseEvent
		valueNegativeEvent
		valueNotEqualDefault
		valueNotEqualDefaultEvent
		valueNotZeroEvent
		valueOutsideLimitsEvent
		valuePositiveEvent
		valueTrueEvent
		valueWithinLimitsEvent
		valueZeroEvent

Chapter 152. IVBDoublePart

Description: IBM VB part support for Double data
Part type: Nonvisual
Part file: vbsample
Header file: ivbdbl.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBDoublePart

Preferred Features — IVBDoublePart

addValue Perform add operator on value.
assignValueToDefault
Assign the value attribute to `false`.
assignValueToOne
Assign value attribute to 1 (`true`).
assignValueToZero
Assign value attribute to 0 (`false`).
highLimit Query the highLimit (`double`) attribute.
lowLimit Query the lowLimit (`double`) attribute.
this A Pointer to the instance.
value Query the value (`double`) attribute.
valueAsText Query the valueAsText (`IString`) attribute.
valueOutsideLimitsEvent
Notification when (value < low limit) or (value > high limit).
valueWithinLimitsEvent
Notification when (high limit >= value >= low limit).

Features — IVBDoublePart

Action	Attribute	Event
addValue ■	asDebugInfo	anyEvent
assignValueToDefault	asString	***defaultValue***
assignValueToE	**defaultValue** ■ ►	digitsEvent
assignValueToHighLimit	enabledForNotification ■	***highLimit***
assignValueToLowLimit	**highLimit** ■ ►	inputStringIsValid
assignValueToOne	**lowLimit** ■ ►	inputStringNotValid
assignValueToPI	**this**	lowerCaseEvent
assignValueToRandom	**value** ■ ►	***lowLimit***

© Copyright IBM Corp. 1995, 1996

IVBDoublePart

Action	Attribute	Event
assignValueToZero	**valueAboveHighLimit ►**	textEqualDefaultEvent
ceilValue	**valueAs1Based ■ ►**	textLengthEvent
copyValueToDefault	**valueAsText ■ ►**	textNotEqualDefaultEvent
disableNotification	**valueBelowLowLimit ►**	upperCaseEvent
divideValue ■	**valueEqualDefault ►**	***value***
floorValue	**valueEqualHighLimit ►**	***valueAboveHighLimit***
multiplyValue ■	**valueEqualLowLimit ►**	valueAboveHighLimitEvent
notifyObservers ■	**valueNegative ►**	***valueAs1Based***
operator != ■	**valueNotEqualDefault ►**	***valueAsText***
operator = ■	**valueNotZero ►**	***valueBelowLowLimit***
operator == ■	**valueOutsideLimits ►**	valueBelowLowLimitEvent
squareValue	**valuePositive ►**	***valueEqualDefault***
subtractValue ■	**valueWithinLimits ►**	valueEqualDefaultEvent
	valueZero ►	***valueEqualHighLimit***
		valueEqualHighLimitEvent
		valueEqualLowLimit
		valueEqualLowLimitEvent
		valueNegative
		valueNegativeEvent
		valueNotEqualDefault
		valueNotEqualDefaultEvent
		valueNotZero
		valueNotZeroEvent
		valueOutsideLimits
		valueOutsideLimitsEvent
		valuePositive
		valuePositiveEvent
		valueWithinLimits
		valueWithinLimitsEvent
		valueZero
		valueZeroEvent

Chapter 153. IVBLogicalAndPart

Description: IBM VB part support for logical AND operation
Part type: Nonvisual
Part file: vbsample
Header file: ivblor.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBLogicalAndPart

Preferred Features — IVBLogicalAndPart

notValue	Query the notValue (`Boolean`) attribute.
this	A Pointer to the instance.
value	Query the value (`Boolean`) attribute.
valueAsText	Query the valueAsText (`IString`) attribute.

Features — IVBLogicalAndPart

Action	Attribute	Event
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	digitsEvent
operator != ■	**defaultInput**	***input1***
operator = ■	enabledForNotification ■	***input10***
operator == ■	**input1 ■ ►**	***input2***
setInput1	**input10 ■ ►**	***input3***
setInput10	**input2 ■ ►**	***input4***
setInput2	**input3 ■ ►**	***input5***
setInput3	**input4 ■ ►**	***input6***
setInput4	**input5 ■ ►**	***input7***
setInput5	**input6 ■ ►**	***input8***
setInput6	**input7 ■ ►**	***input9***
setInput7	**input8 ■ ►**	inputStringIsValid
setInput8	**input9 ■ ►**	inputStringNotValid
setInput9	**notInput1 ■**	lowerCaseEvent
setNotInput1	**notInput10 ■**	***notValue***
setNotInput10	**notInput2 ■**	textEqualDefaultEvent
setNotInput2	**notInput3 ■**	textLengthEvent
setNotInput3	**notInput4 ■**	textNotEqualDefaultEvent
setNotInput4	**notInput5 ■**	upperCaseEvent
setNotInput5	**notInput6 ■**	***value***
setNotInput6	**notInput7 ■**	valueAboveHighLimitEvent

© Copyright IBM Corp. 1995, 1996

IVBLogicalAndPart

Action	Attribute	Event
setNotInput7	**notInput8** ■	***valueAsText***
setNotInput8	**notInput9** ■	valueBelowLowLimitEvent
setNotInput9	**notValue** ►	valueEqualDefaultEvent
	this	valueEqualHighLimitEvent
	value ►	valueEqualLowLimitEvent
	valueAsText ►	valueNegativeEvent
		valueNotEqualDefaultEvent
		valueNotZeroEvent
		valueOutsideLimitsEvent
		valuePositiveEvent
		valueWithinLimitsEvent
		valueZeroEvent

Chapter 154. IVBLogicalOrPart

Description: IBM VB part support for logical OR operation
Part type: Nonvisual
Part file: vbsample
Header file: ivblong.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBLogicalOrPart

Preferred Features — IVBLogicalOrPart

notValue	Query the notValue (`Boolean`) attribute.
this	A Pointer to the instance.
value	Query the value (`Boolean`) attribute.
valueAsText	Query the valueAsText (`IString`) attribute.

Features — IVBLogicalOrPart

Action	Attribute	Event
disableNotification	asDebugInfo	anyEvent
notifyObservers ■	asString	digitsEvent
operator != ■	**defaultInput**	***input1***
operator = ■	enabledForNotification ■	***input10***
operator == ■	**input1 ■ ►**	***input2***
setInput1	**input10 ■ ►**	***input3***
setInput10	**input2 ■ ►**	***input4***
setInput2	**input3 ■ ►**	***input5***
setInput3	**input4 ■ ►**	***input6***
setInput4	**input5 ■ ►**	***input7***
setInput5	**input6 ■ ►**	***input8***
setInput6	**input7 ■ ►**	***input9***
setInput7	**input8 ■ ►**	inputStringIsValid
setInput8	**input9 ■ ►**	inputStringNotValid
setInput9	**notInput1 ■**	lowerCaseEvent
setNotInput1	**notInput10 ■**	***notValue***
setNotInput10	**notInput2 ■**	textEqualDefaultEvent
setNotInput2	**notInput3 ■**	textLengthEvent
setNotInput3	**notInput4 ■**	textNotEqualDefaultEvent
setNotInput4	**notInput5 ■**	upperCaseEvent
setNotInput5	**notInput6 ■**	***value***
setNotInput6	**notInput7 ■**	valueAboveHighLimitEvent

© Copyright IBM Corp. 1995, 1996

IVBLogicalOrPart

Action	Attribute	Event
setNotInput7	**notInput8** ■	***valueAsText***
setNotInput8	**notInput9** ■	valueBelowLowLimitEvent
setNotInput9	**notValue** ►	valueEqualDefaultEvent
	this	valueEqualHighLimitEvent
	value ►	valueEqualLowLimitEvent
	valueAsText ►	valueNegativeEvent
		valueNotEqualDefaultEvent
		valueNotZeroEvent
		valueOutsideLimitsEvent
		valuePositiveEvent
		valueWithinLimitsEvent
		valueZeroEvent

Chapter 155. IVBLongPart

Description: IBM VB part support for Long data
Part type: Nonvisual
Part file: vbsample
Header file: ivblong.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBLongPart

Preferred Features — IVBLongPart

addValue Perform add operator on value.
assignValueToDefault
Assign the value attribute to `false`.
assignValueToOne
Assign value attribute to 1 (`true`).
assignValueToZero
Assign value attribute to 0 (`false`).
highLimit Query the highLimit (`long`) attribute.
lowLimit Query the lowLimit (`long`) attribute.
this A Pointer to the instance.
value Query the value (`long`) attribute.
valueAsText Query the valueAsText (`IString`) attribute.
valueOutsideLimitsEvent
Notification when (value < low limit) or (value > high limit).
valueWithinLimitsEvent
Notification when (high limit >= value >= low limit).

Features — IVBLongPart

Action	Attribute	Event
addValue ■	asDebugInfo	anyEvent
andValue ■	asString	***defaultValue***
assignValueToDefault	**defaultValue** ■ ►	digitsEvent
assignValueToHighLimit	enabledForNotification ■	***highLimit***
assignValueToLowLimit	**highLimit** ■ ►	inputStringIsValid
assignValueToOne	**lowLimit** ■ ►	inputStringNotValid
assignValueToRandom	**this**	lowerCaseEvent
assignValueToZero	**value** ■ ►	***lowLimit***

© Copyright IBM Corp. 1995, 1996

IVBLongPart

Action	Attribute	Event
copyValueToDefault	**valueAboveHighLimit** ►	textEqualDefaultEvent
disableNotification	**valueAs1Based** ■ ►	textLengthEvent
divideValue ■	**valueAsText** ■ ►	textNotEqualDefaultEvent
multiplyValue ■	**valueBelowLowLimit** ►	upperCaseEvent
notifyObservers ■	**valueEqualDefault** ►	***value***
operator != ■	**valueEqualHighLimit** ►	***valueAboveHighLimit***
operator = ■	**valueEqualLowLimit** ►	valueAboveHighLimitEvent
operator == ■	**valueNegative** ►	***valueAs1Based***
orValue ■	**valueNotEqualDefault** ►	***valueAsText***
squareValue	**valueNotZero** ►	***valueBelowLowLimit***
subtractValue ■	**valueOutsideLimits** ►	valueBelowLowLimitEvent
	valuePositive ►	***valueEqualDefault***
	valueWithinLimits ►	valueEqualDefaultEvent
	valueZero ►	***valueEqualHighLimit***
		valueEqualHighLimitEvent
		valueEqualLowLimit
		valueEqualLowLimitEvent
		valueNegative
		valueNegativeEvent
		valueNotEqualDefault
		valueNotEqualDefaultEvent
		valueNotZero
		valueNotZeroEvent
		valueOutsideLimits
		valueOutsideLimitsEvent
		valuePositive
		valuePositiveEvent
		valueWithinLimits
		valueWithinLimitsEvent
		valueZero
		valueZeroEvent

Chapter 156. IVBShortPart

Description: IBM VB part support for Short data
Part type: Nonvisual
Part file: vbsample
Header file: ivbshort.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBShortPart

Preferred Features — IVBShortPart

addValue Perform add operator on value.
assignValueToDefault
Assign the value attribute to `false`.
assignValueToOne
Assign value attribute to 1 (`true`).
assignValueToZero
Assign value attribute to 0 (`false`).
highLimit Query the highLimit (`short`) attribute.
lowLimit Query the lowLimit (`short`) attribute.
this A Pointer to the instance.
value Query the value (`short`) attribute.
valueAsText Query the valueAsText (`IString`) attribute.
valueOutsideLimitsEvent
Notification when (value < low limit) or (value > high limit).
valueWithinLimitsEvent
Notification when (high limit >= value >= low limit).

Features — IVBShortPart

Action	Attribute	Event
addValue ■	asDebugInfo	anyEvent
andValue ■	asString	***defaultValue***
assignValueToDefault	**defaultValue** ■ ►	digitsEvent
assignValueToHighLimit	enabledForNotification ■	***highLimit***
assignValueToLowLimit	**highLimit** ■ ►	inputStringIsValid
assignValueToOne	**lowLimit** ■ ►	inputStringNotValid
assignValueToRandom	**this**	lowerCaseEvent
assignValueToZero	**value** ■ ►	***lowLimit***

© Copyright IBM Corp. 1995, 1996

IVBShortPart

Action	Attribute	Event
copyValueToDefault	**valueAboveHighLimit** ►	textEqualDefaultEvent
disableNotification	**valueAs1Based** ■ ►	textLengthEvent
divideValue ■	**valueAsText** ■ ►	textNotEqualDefaultEvent
multiplyValue ■	**valueBelowLowLimit** ►	upperCaseEvent
notifyObservers ■	**valueEqualDefault** ►	***value***
operator != ■	**valueEqualHighLimit** ►	***valueAboveHighLimit***
operator = ■	**valueEqualLowLimit** ►	valueAboveHighLimitEvent
operator == ■	**valueNegative** ►	***valueAs1Based***
orValue ■	**valueNotEqualDefault** ►	***valueAsText***
squareValue	**valueNotZero** ►	***valueBelowLowLimit***
subtractValue ■	**valueOutsideLimits** ►	valueBelowLowLimitEvent
	valuePositive ►	***valueEqualDefault***
	valueWithinLimits ►	valueEqualDefaultEvent
	valueZero ►	***valueEqualHighLimit***
		valueEqualHighLimitEvent
		valueEqualLowLimit
		valueEqualLowLimitEvent
		valueNegative
		valueNegativeEvent
		valueNotEqualDefault
		valueNotEqualDefaultEvent
		valueNotZero
		valueNotZeroEvent
		valueOutsideLimits
		valueOutsideLimitsEvent
		valuePositive
		valuePositiveEvent
		valueWithinLimits
		valueWithinLimitsEvent
		valueZero
		valueZeroEvent

Chapter 157. IVBStringPart

Description: IBM VB part support for String data
Part type: Nonvisual
Part file: vbsample
Header file: ivbstrng.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBStringPart

Preferred Features — IVBStringPart

appendText Append to the end of text.
changeTextToLowerCase
Change the text to lower case.
changeTextToUpperCase
Change the text to upper case.
text Query the text (`IString`) attribute.
textAsLowerCase
Query the textAsLowerCase (`IString`) attribute.
textAsUpperCase
Query the textAsUpperCase (`IString`) attribute.
textLength Query the textLength (`unsigned long`) attribute.
this A Pointer to the instance.

Features — IVBStringPart

Action	Attribute	Event
appendText ■	asDebugInfo	anyEvent
assignTextToCmdLineParm0	asString	***defaultText***
assignTextToCmdLineParm1	**defaultText** ■ ►	***digits***
assignTextToDateToday	**digits** ►	digitsEvent
assignTextToDefault	enabledForNotification ■	inputStringIsValid
assignTextToEmpty	**lowerCase** ►	inputStringNotValid
assignTextToTimeNow	**text** ■ ►	***lowerCase***
changeTextToLowerCase	**textAsLowerCase** ►	lowerCaseEvent
changeTextToUpperCase	**textAsUpperCase** ►	***text***
disableNotification	**textEqualDefault** ►	***textAsLowerCase***
notifyObservers ■	**textLength** ►	***textAsUpperCase***
operator != ■	**textNotEqualDefault** ►	***textEqualDefault***
operator = ■	**this**	textEqualDefaultEvent

© Copyright IBM Corp. 1995, 1996

IVBStringPart

Action	Attribute	Event
operator == ∎	**upperCase ►**	***textLength***
preappendText ∎		textLengthEvent
reverseText		***textNotEqualDefault***
stripBlanksOnText		textNotEqualDefaultEvent
		upperCase
		upperCaseEvent
		valueAboveHighLimitEvent
		valueBelowLowLimitEvent
		valueEqualDefaultEvent
		valueEqualHighLimitEvent
		valueEqualLowLimitEvent
		valueNegativeEvent
		valueNotEqualDefaultEvent
		valueNotZeroEvent
		valueOutsideLimitsEvent
		valuePositiveEvent
		valueWithinLimitsEvent
		valueZeroEvent

Chapter 158. IVBUnsignedLongPart

Description: IBM VB part support for Unsigned Long data
Part type: Nonvisual
Part file: vbsample
Header file: ivbulong.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBUnsignedLongPart

Preferred Features — IVBUnsignedLongPart

assignValueToDefault
Assign the value attribute to `false`.

assignValueToOne
Assign value attribute to 1 (`true`).

assignValueToZero
Assign value attribute to 0 (`false`).

highLimit Query the highLimit (`unsigned long`) attribute.

lowLimit Query the lowLimit (`unsigned long`) attribute.

this A Pointer to the instance.

value Query the value (`unsigned long`) attribute.

valueAsText Query the valueAsText (`IString`) attribute.

valueOutsideLimitsEvent
Notification when (value < low limit) or (value > high limit).

valueWithinLimitsEvent
Notification when (high limit >= value >= low limit).

Features — IVBUnsignedLongPart

Action	Attribute	Event
addValue ■	asDebugInfo	anyEvent
andValue ■	asString	***defaultValue***
assignValueToDefault	**defaultValue** ■ ►	digitsEvent
assignValueToHighLimit	enabledForNotification ■	***highLimit***
assignValueToLowLimit	**highLimit** ■ ►	inputStringIsValid
assignValueToOne	**lowLimit** ■ ►	inputStringNotValid
assignValueToRandom	**this**	lowerCaseEvent
assignValueToZero	**value** ■ ►	***lowLimit***
copyValueToDefault	**valueAboveHighLimit** ►	textEqualDefaultEvent

© Copyright IBM Corp. 1995, 1996

IVBUnsignedLongPart

Action	Attribute	Event
disableNotification	**valueAs1Based** ■ ►	textLengthEvent
divideValue ■	**valueAsText** ■ ►	textNotEqualDefaultEvent
multiplyValue ■	**valueBelowLowLimit** ►	upperCaseEvent
notifyObservers ■	**valueEqualDefault** ►	***value***
operator != ■	**valueEqualHighLimit** ►	***valueAboveHighLimit***
operator = ■	**valueEqualLowLimit** ►	valueAboveHighLimitEvent
operator == ■	**valueNotEqualDefault** ►	***valueAs1Based***
orValue ■	**valueNotZero** ►	***valueAsText***
squareValue	**valueOutsideLimits** ►	***valueBelowLowLimit***
subtractValue ■	**valueWithinLimits** ►	valueBelowLowLimitEvent
	valueZero ►	***valueEqualDefault***
		valueEqualDefaultEvent
		valueEqualHighLimit
		valueEqualHighLimitEvent
		valueEqualLowLimit
		valueEqualLowLimitEvent
		valueNegativeEvent
		valueNotEqualDefault
		valueNotEqualDefaultEvent
		valueNotZero
		valueNotZeroEvent
		valueOutsideLimits
		valueOutsideLimitsEvent
		valuePositiveEvent
		valueWithinLimits
		valueWithinLimitsEvent
		valueZero
		valueZeroEvent

Chapter 159. IVBUnsignedShortPart

Description: IBM VB part support for Unsigned Short data
Part type: Nonvisual
Part file: vbsample
Header file: ivbushrt.hpp

Derivation:
IBase
IVBase
INotifier
IStandardNotifier
IVBDataTypePart
IVBUnsignedShortPart

Preferred Features — IVBUnsignedShortPart

addValue Perform add operator on value.
assignValueToDefault
Assign the value attribute to `false`.
assignValueToOne
Assign value attribute to 1 (`true`).
assignValueToZero
Assign value attribute to 0 (`false`).
highLimit Query the highLimit (`unsigned short`) attribute.
lowLimit Query the lowLimit (`unsigned short`) attribute.
this A Pointer to the instance.
value Query the value (`unsigned short`) attribute.
valueAsText Query the valueAsText (`IString`) attribute.
valueOutsideLimitsEvent
Notification when (value < low limit) or (value > high limit).
valueWithinLimitsEvent
Notification when (high limit >= value >= low limit).

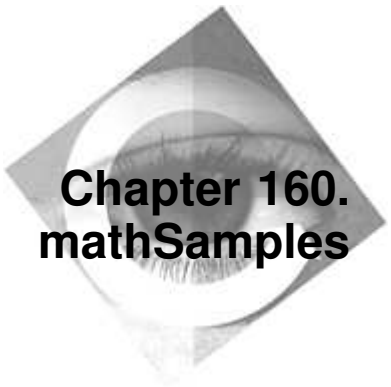
Features — IVBUnsignedShortPart

Action	Attribute	Event
addValue ■	asDebugInfo	anyEvent
andValue ■	asString	***defaultValue***
assignValueToDefault	**defaultValue** ■ ►	digitsEvent
assignValueToHighLimit	enabledForNotification ■	***highLimit***
assignValueToLowLimit	**highLimit** ■ ►	inputStringIsValid
assignValueToOne	**lowLimit** ■ ►	inputStringNotValid
assignValueToRandom	**this**	lowerCaseEvent
assignValueToZero	**value** ■ ►	***lowLimit***

© Copyright IBM Corp. 1995, 1996

IVBUnsignedShortPart

Action	Attribute	Event
copyValueToDefault	**valueAboveHighLimit** ►	textEqualDefaultEvent
disableNotification	**valueAs1Based** ■ ►	textLengthEvent
divideValue ■	**valueAsText** ■ ►	textNotEqualDefaultEvent
multiplyValue ■	**valueBelowLowLimit** ►	upperCaseEvent
notifyObservers ■	**valueEqualDefault** ►	***value***
operator != ■	**valueEqualHighLimit** ►	***valueAboveHighLimit***
operator = ■	**valueEqualLowLimit** ►	valueAboveHighLimitEvent
operator == ■	**valueNotEqualDefault** ►	***valueAs1Based***
orValue ■	**valueNotZero** ►	***valueAsText***
squareValue	**valueOutsideLimits** ►	***valueBelowLowLimit***
subtractValue ■	**valueWithinLimits** ►	valueBelowLowLimitEvent
	valueZero ►	***valueEqualDefault***
		valueEqualDefaultEvent
		valueEqualHighLimit
		valueEqualHighLimitEvent
		valueEqualLowLimit
		valueEqualLowLimitEvent
		valueNegativeEvent
		valueNotEqualDefault
		valueNotEqualDefaultEvent
		valueNotZero
		valueNotZeroEvent
		valueOutsideLimits
		valueOutsideLimitsEvent
		valuePositiveEvent
		valueWithinLimits
		valueWithinLimitsEvent
		valueZero
		valueZeroEvent



Chapter 160. mathSamples

Description: math c library sample functions
Part type: Function group
Part file: vbsample
Header file: math.h

Derivation:
mathSamples

Preferred Features — mathSamples

ceil	ceil computes the smallest integer that is greater than or equal to x.
cos	cos calculates the cosine of x.
floor	floor calculates the largest integer that is less than or equal to x.
pow	pow calculates the value of x to the power of y.
sin	sin calculates the sine of x, with x expressed in radians.
sqrt	sqrt calculates the nonnegative value of the square root of x.
tan	tan calculates the tangent of x, where x is expressed in radians.

Features — mathSamples

Action	Function
acos ■	`double acos (double x);`
asin ■	`double asin (double x);`
atan ■	`double atan (double x);`
ceil ■	`double ceil (double x);`
cos ■	`double cos (double x);`
cosh ■	`double cosh (double x);`
erf ■	`double erf (double x);`
exp ■	`double exp (double x);`
fabs ■	`double fabs (double x);`
floor ■	`double floor (double x);`
fmod ■	`double fmod (double x, double y);`
frexp ■	`double frexp (double x, int* expptr);`
gamma ■	`double gamma (double x);`
hypot ■	`double hypot (double side1, double side2);`
ldexp ■	`double ldexp (double x, int exp);`
log ■	`double log (double x);`
log10 ■	`double log10 (double x);`
modf ■	`double modf (double x, double* intptr);`
pow ■	`double pow (double x, double y);`
sin ■	`double sin (double x);`
sinh ■	`double sinh (double x);`

© Copyright IBM Corp. 1995, 1996

mathSamples

Action	Function
sqrt ■	`double sqrt (double x);`
tan ■	`double tan (double x);`
tanh ■	`double tanh (double x);`
_cabs ■	`double _cabs (struct complex z);`
_j0 ■	`double _j0 (double x);`
_matherr ■	`int _matherr (struct exception* x);`

Chapter 161. staticWindowSamples

Description: static window sample functions
Part type: Function group
Part file: vbsample
Header file: iwindow.hpp

Derivation:
staticWindowSamples

Preferred Features — staticWindowSamples

desktopWindow Static function to return the desktop window.
objectWindow Static function to return object window.

Features — staticWindowSamples

Action	Function
desktopWindow	`static IWindow* IWindow::desktopWindow ();`
objectWindow	`static IWindow* IWindow::objectWindow ();`

© Copyright IBM Corp. 1995, 1996

staticWindowSamples

Chapter 162. stdioSamples

Description: stdio c library sample functions
Part type: Function group
Part file: vbsample
Header file: stdio.h

Derivation:
stdioSamples

Preferred Features — stdioSamples

fclose	fclose closes a stream pointed to by stream.
fdopen	fdopen associates an input or output stream with the file identified by handle.
feof	feof indicates whether the end-of-file flag is set for the given stream.
rename	rename renames the file specified by oldname to the name given by newname.
rewind	rewind repositions the file pointer associated with stream to the beginning of the file.

Features — stdioSamples

Action	Function
fclose ■	`int fclose (FILE* stream);`
fdopen ■	`FILE fdopen (int handle, char* type);`
feof ■	`int feof (FILE* stream);`
ferror ■	`int ferror (FILE* stream);`
fflush ■	`int fflush (FILE* stream);`
fgetpos ■	`int fgetpos (FILE* stream, fpos_t* pos);`
fgetws ■	`wchar_t fgetws (wchar_t* wcs, int n, FILE* stream);`
fileno ■	`int fileno (FILE* stream);`
fopen ■	`FILE fopen (const char* filename, const char* mode);`
fputc ■	`int fputc (int c, FILE* stream);`
fputs ■	`int fputs (const char* string, FILE* stream);`
fputwc ■	`wint_t fputwc (wint_t wc, FILE* stream);`
fputws ■	`int fputws (const wchar_t* wcs, FILE* stream);`
fread ■	`size_t fread (void* buffer, size_t size, size_t count, FILE* stream);`
freopen ■	`FILE freopen (const char* filename, const char* mode, FILE* stream);`
fseek ■	`int fseek (FILE* stream, long int offset, int origin);`
fsetpos ■	`int fsetpos (FILE* stream, const fpos_t* pos);`
ftell ■	`long int ftell (FILE* stream);`

© Copyright IBM Corp. 1995, 1996

stdioSamples

Action	Function
fwrite ■	**size_t** fwrite (const void* **buffer**, size_t **size**, size_t **count**, FILE* **stream**);
getc ■	**int** getc (FILE* **stream**);
gets ■	**char** gets (char* **buffer**);
getwc ■	**wint_t** getwc (FILE* **stream**);
perror ■	void perror (const char* **string**);
putc ■	**int** putc (int **c**, FILE* **stream**);
putchar ■	**int** putchar (int **c**);
puts ■	**int** puts (const char* **string**);
putwc ■	**wint_t** putwc (wint_t **wc**, FILE* **stream**);
remove ■	**int** remove (const char* **filename**);
rename ■	**int** rename (const char* **oldname**, const char* **newname**);
rewind ■	void rewind (FILE* **stream**);
setbuf ■	void setbuf (FILE* **stream**, char* **buffer**);
setvbuf ■	**int** setvbuf (FILE* **stream**, char* **buf**, int **type**, size_t **size**);
tempnam ■	**char** tempnam (char* **dir**, char* **prefix**);
tmpfile	**FILE** tmpfile ();
tmpnam ■	**char** tmpnam (char* **string**);
ungetc ■	**int** ungetc (int **c**, FILE* **stream**);
ungetwc ■	**wint_t** ungetwc (wint_t **wc**, FILE* **stream**);
unlink ■	**int** unlink (const char* **pathname**);
wctob ■	**int** wctob (wint_t **wc**);
_fcloseall	**int** _fcloseall ();
_fgetchar	**int** _fgetchar ();
_flushall	**int** _flushall ();
_fputchar ■	**int** _fputchar (int **c**);
_rmtmp	**int** _rmtmp ();
_set_crt_msg_handle ■	**int** _set_crt_msg_handle (int **fh**);

Chapter 163. stdlibSamples

Description: standard c library sample functions
Part type: Function group
Part file: vbsample
Header file: stdlib.h

Derivation:
stdlibSamples

Preferred Features — stdlibSamples

abort	abort causes an abnormal program termination and returns control to the host environment.
abs	abs returns the absolute value of an integer argument n.
exit	exit returns control to the host environment from the program.
getenv	getenv searches the list of environment variables for an entry corresponding to varname.
rand	rand generates a pseudo-random integer in the range 0 to RAND_MAX (macro defined in <stdlib.
system	system passes the command string to a command processor to be run.

Features — stdlibSamples

Action	Function
abort	`void abort ();`
abs ■	`int abs (int n);`
atof ■	`double atof (const char* string);`
atoi ■	`int atoi (const char* string);`
atol ■	`long int atol (const char* string);`
calloc ■	`void calloc (size_t num, size_t size);`
csid ■	`int csid (const char* c);`
div ■	`div_t div (int numerator, int denominator);`
exit ■	`void exit (int status);`
free ■	`void free (void* ptr);`
getenv ■	`char getenv (const char* varname);`
labs ■	`long int labs (long int n);`
ldiv ■	`ldiv_t ldiv (long int numerator, long int denominator);`
malloc ■	`void malloc (size_t size);`
mblen ■	`int mblen (const char* string, size_t n);`
mbstowcs ■	`size_t mbstowcs (wchar_t* dest, const char* string, size_t len);`
mbtowc ■	`int mbtowc (wchar_t* pwc, const char* string, size_t n);`
putenv ■	`int putenv (char* envstring);`

© Copyright IBM Corp. 1995, 1996

stdlibSamples

Action	Function
rand	**int** rand ();
realloc ■	void realloc (void* **ptr**, size_t **size**);
rpmatch ■	**int** rpmatch (const char* **response**);
srand ■	void srand (unsigned int **seed**);
strtod ■	**double** strtod (const char* **nptr**, char** **endptr**);
strtol ■	**long int** strtol (const char* **nptr**, char** **endptr**, int **base**);
strtold ■	**long double** strtold (const char* **nptr**, char** **endptr**);
swab ■	void swab (char* **source**, char* **destination**, int **n**);
system ■	**int** system (char* **string**);
wcsid ■	**int** wcsid (const wchar_t **c**);
wcstombs ■	**size_t** wcstombs (char* **dest**, const wchar_t* **string**, size_t **n**);
wctomb ■	**int** wctomb (char* **string**, wchar_t **wc**);
_alloca ■	void _alloca (size_t **size**);
_atold ■	**long double** _atold (const char* **nptr**);
_crotl ■	**unsigned char** _crotl (unsigned char **value**, int **shift**);
_debug_calloc ■	void _debug_calloc (size_t **num**, size_t **size**, const char* **file**, size_t **line**);
_debug_free ■	void _debug_free (void* **ptr**, const char* **file**, size_t **line**);
_debug_heapmin ■	**int** _debug_heapmin (const char* **file**, size_t **line**);
_debug_malloc ■	void _debug_malloc (size_t **size**, const char* **file**, size_t **line**);
_debug_realloc ■	void _debug_realloc (void* **ptr**, size_t **size**, const char* **file**, size_t **line**);
_debug_tcalloc ■	void _debug_tcalloc (size_t **num**, size_t **size**, const char* **file**, size_t **line**);
_debug_tfree ■	void _debug_tfree (void* **ptr**, const char* **file**, size_t **line**);
_debug_theapmin ■	**int** _debug_theapmin (const char* **file**, size_t **line**);
_debug_tmalloc ■	void _debug_tmalloc (size_t **size**, const char* **file**, size_t **line**);
_debug_trealloc ■	void _debug_trealloc (void* **ptr**, size_t **size**, const char* **file**, size_t **line**);
_dump_allocated_delta ■	void _dump_allocated_delta (int **nbytes**);
_ecvt ■	**char** _ecvt (double **value**, int **ndigits**, int* **decptr**, int* **signptr**);
_endthread	void _endthread ();
_exit ■	void _exit (int **status**);
_fcvt ■	**char** _fcvt (double **value**, int **ndec**, int* **decptr**, int* **signptr**);
_freemod ■	**int** _freemod (unsigned long **module_handle**);
_fullpath ■	**char** _fullpath (char* **pathbuf**, char* **partialpath**, size_t **n**);
_gcvt ■	**char** _gcvt (double **value**, int **ndec**, char* **buffer**);
_heapmin	**int** _heapmin ();
_heap_check	void _heap_check ();
_itoa ■	**char** _itoa (int **value**, char* **string**, int **radix**);
_loadmod ■	**int** _loadmod (char* **module_name**, unsigned long* **module_handle**);
_lrotl ■	**unsigned long** _lrotl (unsigned long **value**, int **shift**);
_ltoa ■	**char** _ltoa (long **value**, char* **string**, int **radix**);
_makepath ■	void _makepath (char* **path**, char* **drive**, char* **dir**, char* **fname**, char* **ext**);
_msize ■	**size_t** _msize (void* **ptr**);
_onexit ■	**onexit_t** _onexit (onexit_t **func**);
_rotl ■	**unsigned int** _rotl (unsigned int **value**, int **shift**);

Action	Function
_searchenv ■	`void _searchenv (char* name, char* env_var, char* path);`
_splitpath ■	`void _splitpath (char* path, char* drive, char* dir, char* fname, char* ext);`
_tcalloc ■	`void _tcalloc (size_t number, size_t size);`
_tdump_allocated_delta ■	`void _tdump_allocated_delta (int nbytes);`
_tfree ■	`void _tfree (void* ptr);`
_threadstore	`void _threadstore ();`
_tmalloc ■	`void _tmalloc (size_t size);`
_trealloc ■	`void _trealloc (void* ptr, size_t size);`
_ultoa ■	`char _ultoa (unsigned long value, char* string, int radix);`
__parmdwords	`unsigned char __parmdwords ();`

stdlibSamples

Part 6. Part Interface Features

The following chapters describe actions, attributes, and events of all parts covered by this book.

Chapter 164. Actions 653
abort — stdlibSamples — vbsample 653
abs — stdlibSamples — vbsample 653
access — ioSamples — vbsample 653
acos — mathSamples — vbsample 653
acquire — IMMDevice — vbmm 653
add — IComboBox 653
add — IListBox 654
add — IMultiLineEdit 654
add — IPointArray 654
add — IRBag — vbcc 654
add — IRDeque — vbcc 654
add — IREqualitySequence — vbcc 654
add — IRHeap — vbcc 654
add — IRQueue — vbcc 654
add — IRSequence 654
add — IRSet — vbcc 655
add — IRSortedBag — vbcc 655
add — IRSortedSet — vbcc 655
add — IRStack — vbcc 655
add — ITextSpinButton 655
addAllFrom — IACollection 655
addAllFrom — IADeque — vbcc 655
addAllFrom — IAQueue — vbcc 655
addAllFrom — IAStack — vbcc 655
addAscending — IComboBox 656
addAscending — IListBox 656
addAsFirst — IComboBox 656
addAsFirst — IListBox 656
addAsFirst — IRDeque — vbcc 656
addAsFirst — IREqualitySequence — vbcc 656
addAsFirst — IRSequence 656
addAsFirst — ITextSpinButton 656
addAsFirst — IToolBar 657
addAsLast — IComboBox 657
addAsLast — IListBox 657
addAsLast — IMultiLineEdit 657

© Copyright IBM Corp. 1995, 1996

addAsLast — IRDeque — vbcc . . . 657
addAsLast — IREqualitySequence — vbcc . . . 657
addAsLast — IRQueue — vbcc . . . 657
addAsLast — IRSequence . . . 657
addAsLast — IRStack — vbcc . . . 658
addAsLast — ITextSpinButton . . . 658
addAsLast — IToolBar . . . 658
addAsNext — IToolBar . . . 658
addAsPrevious — IToolBar . . . 658
addAtOffset — IMultiLineEdit . . . 658
addAtPosition — IREqualitySequence — vbcc . . . 658
addAtPosition — IRSequence . . . 659
addBorder — IBaseSpinButton . . . 659
addColumn — IContainerControl . . . 659
addCustomer — ICompany — vbsample . . . 659
addDescending — IComboBox . . . 659
addDescending — IListBox . . . 659
addDetent — ISlider . . . 659
addDifference — IRBag — vbcc . . . 659
addDifference — IRSet — vbcc . . . 660
addDifference — IRSortedBag — vbcc . . . 660
addDifference — IRSortedSet — vbcc . . . 660
addDrive — IVBFileDialog . . . 660
addEntryAsFirst — IMMAudioCDContents — vbmm . . . 660
addExtension — IFrameWindow . . . 660
addFileType — IVBFileDialog . . . 660
addFirstPage — INotebook . . . 661
addIntersection — IRBag — vbcc . . . 661
addIntersection — IRSet — vbcc . . . 661
addIntersection — IRSortedBag — vbcc . . . 661
addIntersection — IRSortedSet — vbcc . . . 661
addKey — IAcceleratorTable . . . 661
addLastPage — INotebook . . . 661
addLibraries — IHelpWindow . . . 662
addLine — IMultiLineEdit . . . 662
addLineAsLast — IMultiLineEdit . . . 662
addOrReplaceElementWithKey — IProfile . . . 662
addOrReplaceKey — IAcceleratorTable . . . 662
addPageAfter — INotebook . . . 662
addPageBefore — INotebook . . . 663
addToCell — IMultiCellCanvas . . . 663
addToWindowList — IFrameWindow . . . 663
addUnion — IRBag — vbcc . . . 663

addUnion — IRSet — vbcc 663
addUnion — IRSortedBag — vbcc 663
addUnion — IRSortedSet — vbcc 663
addValue — IVBDoublePart — vbsample 664
addValue — IVBLongPart — vbsample 664
addValue — IVBShortPart — vbsample 664
addValue — IVBUnsignedLongPart — vbsample 664
addValue — IVBUnsignedShortPart — vbsample 664
allElementsDo — IRBag — vbcc 664
allElementsDo — IRDeque — vbcc 664
allElementsDo — IREqualitySequence — vbcc 664
allElementsDo — IRHeap — vbcc 665
allElementsDo — IRQueue — vbcc 665
allElementsDo — IRSequence 665
allElementsDo — IRSet — vbcc 665
allElementsDo — IRSortedBag — vbcc 665
allElementsDo — IRSortedSet — vbcc 665
allElementsDo — IRStack — vbcc 666
andValue — IVBLongPart — vbsample 666
andValue — IVBShortPart — vbsample 666
andValue — IVBUnsignedLongPart — vbsample 666
andValue — IVBUnsignedShortPart — vbsample 666
appendText — IVBStringPart — vbsample 666
applyBidiSettings — IWindow 666
arrangeIconView — IContainerControl 666
asGUIStyle — I3StateCheckBox 667
asin — mathSamples — vbsample 667
assignTextToCmdLineParm0 — IVBStringPart — vbsample 667
assignTextToCmdLineParm1 — IVBStringPart — vbsample 667
assignTextToDateToday — IVBStringPart — vbsample 667
assignTextToDefault — IVBStringPart — vbsample 667
assignTextToEmpty — IVBStringPart — vbsample 667
assignTextToTimeNow — IVBStringPart — vbsample 667
assignValueToDefault — IVBBooleanPart — vbsample 667
assignValueToDefault — IVBDoublePart — vbsample 668
assignValueToDefault — IVBLongPart — vbsample 668
assignValueToDefault — IVBShortPart — vbsample 668
assignValueToDefault — IVBUnsignedLongPart — vbsample 668
assignValueToDefault — IVBUnsignedShortPart — vbsample 668
assignValueToE — IVBDoublePart — vbsample 668
assignValueToFalse — IVBBooleanPart — vbsample 668
assignValueToHighLimit — IVBDoublePart — vbsample 668
assignValueToHighLimit — IVBLongPart — vbsample 668

assignValueToHighLimit — IVBShortPart — vbsample 668
assignValueToHighLimit — IVBUnsignedLongPart — vbsample 669
assignValueToHighLimit — IVBUnsignedShortPart — vbsample 669
assignValueToLowLimit — IVBDoublePart — vbsample 669
assignValueToLowLimit — IVBLongPart — vbsample 669
assignValueToLowLimit — IVBShortPart — vbsample 669
assignValueToLowLimit — IVBUnsignedLongPart — vbsample 669
assignValueToLowLimit — IVBUnsignedShortPart — vbsample 669
assignValueToOne — IVBDoublePart — vbsample 669
assignValueToOne — IVBLongPart — vbsample 669
assignValueToOne — IVBShortPart — vbsample 669
assignValueToOne — IVBUnsignedLongPart — vbsample 670
assignValueToOne — IVBUnsignedShortPart — vbsample 670
assignValueToPI — IVBDoublePart — vbsample 670
assignValueToRandom — IVBDoublePart — vbsample 670
assignValueToRandom — IVBLongPart — vbsample 670
assignValueToRandom — IVBShortPart — vbsample 670
assignValueToRandom — IVBUnsignedLongPart — vbsample 670
assignValueToRandom — IVBUnsignedShortPart — vbsample 670
assignValueToTrue — IVBBooleanPart — vbsample 670
assignValueToZero — IVBDoublePart — vbsample 670
assignValueToZero — IVBLongPart — vbsample 671
assignValueToZero — IVBShortPart — vbsample 671
assignValueToZero — IVBUnsignedLongPart — vbsample 671
assignValueToZero — IVBUnsignedShortPart — vbsample 671
atan — mathSamples — vbsample 671
atof — stdlibSamples — vbsample 671
atoi — stdlibSamples — vbsample 671
atol — stdlibSamples — vbsample 671
b2c — IString 671
b2d — IString 672
b2x — IString 672
beginEdit — ICnrEditHandler 672
beginFlashing — IFrameWindow 672
beginUsingFont — IFont 672
c2b — IString 672
c2d — IString 672
c2x — IString 672
calloc — stdlibSamples — vbsample 672
capturePointer — IWindow 673
ceil — mathSamples — vbsample 673
ceilValue — IVBDoublePart — vbsample 673
center — IString 673

centerAt — IRectangle . . . 673
centeredAt — IRectangle . . . 673
change — I0String . . . 673
change — IString . . . 674
changeTextToLowerCase — IVBStringPart — vbsample . . . 674
changeTextToUpperCase — IVBStringPart — vbsample . . . 674
charType — IString . . . 674
charWidth — IFont . . . 674
chmod — ioSamples — vbsample . . . 674
clear — IEntryField . . . 674
clear — IMultiLineEdit . . . 674
clearDefaultTransparentColor — IToolBarButton . . . 674
click — IButton . . . 675
clientRectFor — IFrameWindow . . . 675
close — IClipboard . . . 675
close — IDynamicLinkLibrary . . . 675
close — IFrameWindow . . . 675
close — IMMDevice — vbmm . . . 675
close — ioSamples — vbsample . . . 675
closeDoor — IMMRemovableMedia — vbmm . . . 675
closeEdit — IContainerControl . . . 676
collapse — IContainerControl . . . 676
collapse — IToolBar . . . 676
collapseTree — IContainerControl . . . 676
columnAt — IContainerControl . . . 676
columnUnderPoint — IContainerControl . . . 676
columnWidth — IMultiCellCanvas . . . 676
compare — IADeque — vbcc . . . 676
compare — IAEqualitySequence — vbcc . . . 677
compare — IAQueue — vbcc . . . 677
compare — IASequence . . . 677
compare — IASortedBag — vbcc . . . 677
compare — IASortedSet — vbcc . . . 677
compare — IAStack — vbcc . . . 677
connectedDeviceId — IMMDevice — vbmm . . . 677
containerFromHandle — IContainerControl . . . 678
contains — IRBag — vbcc . . . 678
contains — IRectangle . . . 678
contains — IREqualitySequence — vbcc . . . 678
contains — IRSet — vbcc . . . 678
contains — IRSortedBag — vbcc . . . 678
contains — IRSortedSet — vbcc . . . 678
containsAllFrom — IAEqualityCollection — vbcc . . . 678

containsApplication — IProfile . . . 679
containsKeyLike — IAcceleratorTable . . . 679
containsKeyName — IProfile . . . 679
containsObject — IContainerControl . . . 679
convertToGUIStyle — IMenuItem . . . 679
convertToGUIStyle — IWindow . . . 679
convertToWorkspace — IContainerControl . . . 679
copy — IACollection . . . 680
copy — IADeque — vbcc . . . 680
copy — IAQueue — vbcc . . . 680
copy — IAStack — vbcc . . . 680
copy — IEntryField . . . 680
copy — IMMRecordable — vbmm . . . 680
copy — IMultiLineEdit . . . 680
copy — IString . . . 680
copyObjectTo — IContainerControl . . . 681
copyValueToDefault — IVBBooleanPart — vbsample . . . 681
copyValueToDefault — IVBDoublePart — vbsample . . . 681
copyValueToDefault — IVBLongPart — vbsample . . . 681
copyValueToDefault — IVBShortPart — vbsample . . . 681
copyValueToDefault — IVBUnsignedLongPart — vbsample . . . 681
copyValueToDefault — IVBUnsignedShortPart — vbsample . . . 681
cos — mathSamples — vbsample . . . 681
cosh — mathSamples — vbsample . . . 682
creat — ioSamples — vbsample . . . 682
csid — stdlibSamples — vbsample . . . 682
cueForPlayback — IMMPlayableDevice — vbmm . . . 682
cueForRecording — IMMRecordable — vbmm . . . 682
cut — IEntryField . . . 682
cut — IMMRecordable — vbmm . . . 682
cut — IMultiLineEdit . . . 682
d2b — IString . . . 683
d2c — IString . . . 683
d2x — IString . . . 683
data — IClipboard . . . 683
dataAsDate — IContainerColumn . . . 683
dataAsIcon — IContainerColumn . . . 683
dataAsNumber — IContainerColumn . . . 683
dataAsString — IContainerColumn . . . 683
dataAsTime — IContainerColumn . . . 683
dayName — IDate . . . 684
daysInMonth — IDate . . . 684
daysInYear — IDate . . . 684

deleteElementWithApplication — IProfile 684
deleteElementWithKey — IProfile 684
deletePendingEvents — IMMDevice — vbmm 684
deleteSelection — IMMRecordable — vbmm 684
dequeue — IRQueue — vbcc 684
descendentsOf — IContainerControl 685
deselect — IBaseComboBox 685
deselect — IBaseListBox 685
deselect — ICollectionViewComboBox 685
deselect — ICollectionViewListBox 685
deselect — ISettingButton 685
deselect — IVBContainerControl 685
deselectAll — IBaseComboBox 685
deselectAll — IBaseListBox 686
desktopWindow — staticWindowSamples — vbsample 686
detailsObjectRectangle — IContainerControl 686
detentPosition — ISlider 686
differenceWith — IRBag — vbcc 686
differenceWith — IRSet — vbcc 686
differenceWith — IRSortedBag — vbcc 686
differenceWith — IRSortedSet — vbcc 686
disable — IHandler 687
disable — IInfoArea 687
disable — IMenuItem 687
disable — IWindow 687
disableAnimateWhenLatched — IAnimatedButton 687
disableAudio — IMMDevice — vbmm 687
disableAutoLatch — ICustomButton 687
disableAutoPlay — IMMAudioCD — vbmm 687
disableAutoScroll — IEntryField 687
disableAutoSelect — I3StateCheckBox 688
disableAutoSelect — ICheckBox 688
disableAutoSelect — IRadioButton 688
disableAutoTab — IEntryField 688
disableCaching — IContainerControl 688
disableCommand — IEntryField 688
disableConnector — IMMDevice — vbmm 688
disableContinuousPlay — IMMAudioCD — vbmm 688
disableCursorSelect — IRadioButton 688
disableDataUpdate — IBaseSpinButton 689
disableDataUpdate — IContainerColumn 689
disableDataUpdate — IContainerControl 689
disableDataUpdate — IContainerObject 689

disableDataUpdate — IEntryField 689
disableDataUpdate — IMultiLineEdit 689
disableDefault — IPushButton 689
disableDragDelete — IToolBarButton 689
disableDragDrop — IToolBar 689
disableDragLines — IMultiCellCanvas 690
disableDrawBackground — IContainerControl 690
disableDrawItem — IBaseListBox 690
disableDrawItem — IContainerControl 690
disableDrawItem — IProgressIndicator 690
disableDrop — IContainerControl 690
disableDrop — IContainerObject 690
disableExtendedSelect — IBaseListBox 690
disableFastSpin — IBaseSpinButton 690
disableFillBackground — IStaticText 691
disableFlyOverHelp — IVBFlyText 691
disableGridLines — IMultiCellCanvas 691
disableGroup — IControl 691
disableHalftone — IStaticText 691
disableHeadingUpdate — IContainerColumn 691
disableHeadphones — IMMMasterAudio — vbmm 691
disableHelp — IPushButton 691
disableInsertMode — IEntryField 691
disableLatching — ICustomButton 692
disableMargin — IEntryField 692
disableMisfitFiltering — IToolBar 692
disableMonitoring — IMMAmpMixer — vbmm 692
disableMouseClickFocus — IButton 692
disableMultipleSelect — IBaseListBox 692
disableNoAdjustPosition — IBaseListBox 692
disableNotification — IRBag — vbcc 692
disableNotification — IRDeque — vbcc 692
disableNotification — IREqualitySequence — vbcc 693
disableNotification — IRHeap — vbcc 693
disableNotification — IRQueue — vbcc 693
disableNotification — IRSequence 693
disableNotification — IRSet — vbcc 693
disableNotification — IRSortedBag — vbcc 693
disableNotification — IRSortedSet — vbcc 693
disableNotification — IRStack — vbcc 693
disableNotification — IStandardNotifier 693
disableNotification — IWindow 693
disableRibbonStrip — IProgressIndicator 694

disableSizeToGraphic — IGraphicPushButton 694
disableSnapToTick — IProgressIndicator 694
disableSpeakers — IMMMasterAudio — vbmm 694
disableStrikeout — IStaticText 694
disableSystemCommand — IPushButton 694
disableTabStop — IControl 694
disableTitleUpdate — IContainerControl 694
disableTrace — ITrace 694
disableUnderscore — IStaticText 695
disableUpdate — IWindow 695
disableWordWrap — IMultiLineEdit 695
disableWriteLineNumber — ITrace 695
disableWritePrefix — ITrace 695
discard — IEntryField 695
discard — IMultiLineEdit 695
dismiss — IFrameWindow 695
dispatchRemainingHandlers — IWindow 695
distanceFrom — IPair 696
div — stdlibSamples — vbsample 696
divideValue — IVBDoublePart — vbsample 696
divideValue — IVBLongPart — vbsample 696
divideValue — IVBShortPart — vbsample 696
divideValue — IVBUnsignedLongPart — vbsample 696
divideValue — IVBUnsignedShortPart — vbsample 696
dotProduct — IPair 696
dup — ioSamples — vbsample 696
dup2 — ioSamples — vbsample 697
editColumnTitle — IContainerControl 697
editContainerTitle — IContainerControl 697
editObject — IContainerControl 697
elementAt — ITextSpinButton 697
elementAtPosition — IRDeque — vbcc 697
elementAtPosition — IREqualitySequence — vbcc 697
elementAtPosition — IRQueue — vbcc 697
elementAtPosition — IRSequence 698
elementAtPosition — IRSortedBag — vbcc 698
elementAtPosition — IRSortedSet — vbcc 698
elementAtPosition — IRStack — vbcc 698
elementWithKey — IProfile 698
empty — IClipboard 698
enable — IInfoArea 698
enable — IMenuItem 698
enable — IWindow 699

enableConnector — IMMDevice — vbmm 699
enableDataUpdate — IContainerControl 699
enableDrop — IContainerControl 699
enableHeadphones — IMMMasterAudio — vbmm 699
enableSpeakers — IMMMasterAudio — vbmm 699
enableTrace — ITrace 699
enableUpdate — IWindow 699
enableWriteLineNumber — ITrace 700
enableWritePrefix — ITrace 700
endEdit — ICnrEditHandler 700
endFlashing — IFrameWindow 700
endUsingFont — IFont 700
enqueue — IRQueue — vbcc 700
erf — mathSamples — vbsample 700
exit — stdlibSamples — vbsample 700
exp — mathSamples — vbsample 700
expand — IContainerControl 701
expand — IToolBar 701
expandBy — IRectangle 701
expandedBy — IRectangle 701
expandTree — IContainerControl 701
exportSelectedTextToFile — IMultiLineEdit 701
exportToFile — IMultiLineEdit 701
fabs — mathSamples — vbsample 701
fclose — stdioSamples — vbsample 702
fdopen — stdioSamples — vbsample 702
feof — stdioSamples — vbsample 702
ferror — stdioSamples — vbsample 702
fflush — stdioSamples — vbsample 702
fgetpos — stdioSamples — vbsample 702
fgetws — stdioSamples — vbsample 702
fileno — stdioSamples — vbsample 702
filter — IContainerControl 703
floor — mathSamples — vbsample 703
floorValue — IVBDoublePart — vbsample 703
flyHelpText — IFlyOverHelpHandler 703
fmod — mathSamples — vbsample 703
fopen — stdioSamples — vbsample 703
format — IClipboard 703
formatAsHandle — IClipboard 703
fputc — stdioSamples — vbsample 704
fputs — stdioSamples — vbsample 704
fputwc — stdioSamples — vbsample 704

fputws — stdioSamples — vbsample 704
frameRectFor — IFrameWindow 704
fread — stdioSamples — vbsample 704
free — stdlibSamples — vbsample 704
freopen — stdioSamples — vbsample 705
frexp — mathSamples — vbsample 705
fseek — stdioSamples — vbsample 705
fsetpos — stdioSamples — vbsample 705
ftell — stdioSamples — vbsample 705
fwrite — stdioSamples — vbsample 705
gamma — mathSamples — vbsample 705
getc — stdioSamples — vbsample 705
getenv — stdlibSamples — vbsample 706
gets — stdioSamples — vbsample 706
getwc — stdioSamples — vbsample 706
goToEntry — IMMAudioCD — vbmm 706
handleCursoredChange — IContainerObject 706
handleException — IWindow 706
handleFor — IFrameWindow 706
handleInuseChange — IContainerObject 707
handleOpen — IContainerObject 707
handleSelectedChange — IContainerObject 707
handleTreeCollapse — IContainerObject 707
handleTreeExpand — IContainerObject 707
hasData — IClipboard 707
helpWindow — IHelpWindow 707
hide — IContainerColumn 708
hide — IContainerObject 708
hide — IWindow 708
hideDetailsViewTitles — IContainerControl 708
hideList — IBaseComboBox 708
hideObject — IContainerControl 708
hidePanelIds — IHelpWindow 708
hideSeparators — IContainerColumn 708
hideSourceEmphasis — IWindow 708
hideSplitBar — IContainerControl 709
hideTitle — IContainerControl 709
hideTitleSeparator — IContainerControl 709
hideTreeLine — IContainerControl 709
highlight — IButton 709
hypot — mathSamples — vbsample 709
iconRectangle — IContainerControl 709
immediateDescendentsOf — IContainerControl 709

importFromFile — IMultiLineEdit 710
includes — IRange 710
includes — IString 710
indexOf — IString 710
indexOfAnyBut — IString 710
indexOfAnyOf — IString 710
indexOfPhrase — IString 710
indexOfWord — IString 711
insert — I0String 711
insert — IPointArray 711
insert — IString 711
integerWithKey — IProfile 711
intersectionWith — IRBag — vbcc 711
intersectionWith — IRSet — vbcc 711
intersectionWith — IRSortedBag — vbcc 711
intersectionWith — IRSortedSet — vbcc 712
intersects — IRectangle 712
isAbbreviationFor — IString 712
isAnExtension — IFrameWindow 712
isatty — ioSamples — vbsample 712
isCollapsed — IContainerControl 712
isColumnExpandable — IMultiCellCanvas 712
isColumnRight — IContainerControl 712
isConnectionSupported — IMMDevice — vbmm 713
isConnectorEnabled — IMMDevice — vbmm 713
isCursored — IContainerControl 713
isDropOnAble — IContainerControl 713
isEntryPoint32Bit — IDynamicLinkLibrary 713
isExpanded — IContainerControl 713
isInUse — IContainerControl 713
isLayoutDistorted — IWindow 713
isLeapYear — IDate 714
isLike — IAcceleratorKey 714
isLike — IString 714
isMoveValid — IContainerControl 714
isRowExpandable — IMultiCellCanvas 714
isSelected — IBaseComboBox 714
isSelected — IBaseListBox 714
isSelected — IContainerControl 714
isSource — IContainerControl 715
isTarget — IContainerControl 715
isValid — IDate 715
isWriteable — IContainerControl 715

itemHandle — IBaseComboBox . . . 715
itemHandle — IBaseListBox . . . 715
itemText — IBaseComboBox . . . 715
itemText — IBaseListBox . . . 715
justifyData — IContainerColumn . . . 715
justifyHeading — IContainerColumn . . . 716
labs — stdlibSamples — vbsample . . . 716
lastIndexOf — IString . . . 716
lastIndexOfAnyBut — IString . . . 716
lastIndexOfAnyOf — IString . . . 716
ldexp — mathSamples — vbsample . . . 716
ldiv — stdlibSamples — vbsample . . . 716
leftJustify — IString . . . 717
lengthOfWord — IString . . . 717
lineFrom — IString . . . 717
loadAccelTable — IResourceLibrary . . . 717
loadBitmap — IResourceLibrary . . . 717
loadDialog — IResourceLibrary . . . 717
loadHelpTable — IResourceLibrary . . . 717
loadIcon — IResourceLibrary . . . 717
loadMenu — IResourceLibrary . . . 718
loadMessage — IResourceLibrary . . . 718
loadOnThread — IMMFileMedia — vbmm . . . 718
loadPointer — IResourceLibrary . . . 718
loadString — IResourceLibrary . . . 718
locateOrAdd — IRBag — vbcc . . . 718
locateOrAdd — IREqualitySequence — vbcc . . . 718
locateOrAdd — IRSet — vbcc . . . 718
locateOrAdd — IRSortedBag — vbcc . . . 719
locateOrAdd — IRSortedSet — vbcc . . . 719
locateText — IBaseComboBox . . . 719
locateText — IBaseListBox . . . 719
log — mathSamples — vbsample . . . 719
log10 — mathSamples — vbsample . . . 719
logicalAndValue — IVBBooleanPart — vbsample . . . 719
logicalNotValue — IVBBooleanPart — vbsample . . . 720
logicalOrValue — IVBBooleanPart — vbsample . . . 720
longHelpText — IFlyOverHelpHandler . . . 720
lowerCase — IString . . . 720
lseek — ioSamples — vbsample . . . 720
malloc — stdlibSamples — vbsample . . . 720
matchForMnemonic — IWindow . . . 720
maximize — IFrameWindow . . . 720

maximum — IPair . . . 721
mblen — stdlibSamples — vbsample . . . 721
mbstowcs — stdlibSamples — vbsample . . . 721
mbtowc — stdlibSamples — vbsample . . . 721
menuSelected — IVBCheckMenuHandler . . . 721
minimize — IFrameWindow . . . 721
minimum — IPair . . . 721
minTextWidth — IFont . . . 721
modf — mathSamples — vbsample . . . 722
monthName — IDate . . . 722
moveAfter — IToolBar . . . 722
moveBefore — IToolBar . . . 722
moveBy — IRectangle . . . 722
movedBy — IRectangle . . . 722
movedTo — IRectangle . . . 722
moveIconTo — IContainerControl . . . 722
moveObjectTo — IContainerControl . . . 723
moveSizeToClient — IFrameWindow . . . 723
moveToFirst — IToolBar . . . 723
moveToLast — IToolBar . . . 723
multiLineEdit — ICnrEditHandler . . . 723
multiplyValue — IVBDoublePart — vbsample . . . 723
multiplyValue — IVBLongPart — vbsample . . . 723
multiplyValue — IVBShortPart — vbsample . . . 724
multiplyValue — IVBUnsignedLongPart — vbsample . . . 724
multiplyValue — IVBUnsignedShortPart — vbsample . . . 724
nextPage — INotebook . . . 724
nlsCompare — IContainerControl . . . 724
notebookSize — INotebook . . . 724
notifyObservers — IStandardNotifier . . . 724
notifyObservers — IWindow . . . 724
notifyOwner — IFrameWindow . . . 725
numberOfOccurrences — IRBag — vbcc . . . 725
numberOfOccurrences — IREqualitySequence — vbcc . . . 725
numberOfOccurrences — IRSortedBag — vbcc . . . 725
numberOfTicks — IProgressIndicator . . . 725
objectAt — IContainerControl . . . 725
objectUnderPoint — IContainerControl . . . 725
objectWindow — staticWindowSamples — vbsample . . . 725
occurrencesOf — IString . . . 726
open — IClipboard . . . 726
open — IDynamicLinkLibrary . . . 726
open — IMMDevice — vbmm . . . 726

open — ioSamples — vbsample . . . 726
openDoor — IMMRemovableMedia — vbmm . . . 726
openOnThread — IMMDevice — vbmm . . . 726
operator != — IABag — vbcc . . . 726
operator != — IAcceleratorKey . . . 727
operator != — IAddress — vbsample . . . 727
operator != — IAEqualitySequence — vbcc . . . 727
operator != — IASet — vbcc . . . 727
operator != — IASortedBag — vbcc . . . 727
operator != — IASortedSet — vbcc . . . 727
operator != — IColor . . . 727
operator != — ICompany — vbsample . . . 727
operator != — ICustomer — vbsample . . . 727
operator != — IDate . . . 728
operator != — IMMTime — vbmm . . . 728
operator != — IOrderedRecord — vbsample . . . 728
operator != — IPair . . . 728
operator != — IPointArray . . . 728
operator != — IRectangle . . . 728
operator != — ITime . . . 728
operator != — IVBBooleanPart — vbsample . . . 728
operator != — IVBDoublePart — vbsample . . . 728
operator != — IVBLogicalAndPart — vbsample . . . 728
operator != — IVBLogicalOrPart — vbsample . . . 728
operator != — IVBLongPart — vbsample . . . 729
operator != — IVBShortPart — vbsample . . . 729
operator != — IVBStringPart — vbsample . . . 729
operator != — IVBUnsignedLongPart — vbsample . . . 729
operator != — IVBUnsignedShortPart — vbsample . . . 729
operator %= — IPair . . . 729
operator & — IRectangle . . . 729
operator & — IString . . . 729
operator &= — IRectangle . . . 729
operator &= — IString . . . 729
operator * — IMngPointer — vbcc . . . 730
operator *= — IPair . . . 730
operator + — IDate . . . 730
operator + — IMMTime — vbmm . . . 730
operator + — IString . . . 730
operator + — ITime . . . 730
operator += — IDate . . . 730
operator += — IMMTime — vbmm . . . 730
operator += — IPair . . . 730

operator += — IString . . . 731
operator += — ITime . . . 731
operator - — IDate . . . 731
operator - — IMMTime — vbmm . . . 731
operator - — IPair . . . 731
operator - — ITime . . . 731
operator -= — IDate . . . 731
operator -= — IMMTime — vbmm . . . 731
operator -= — IPair . . . 731
operator -= — ITime . . . 732
operator /= — IPair . . . 732
operator < — IDate . . . 732
operator < — IMMTime — vbmm . . . 732
operator < — IOrderedRecord — vbsample . . . 732
operator < — IPair . . . 732
operator < — ITime . . . 732
operator <= — IDate . . . 732
operator <= — IMMTime — vbmm . . . 733
operator <= — IOrderedRecord — vbsample . . . 733
operator <= — IPair . . . 733
operator <= — ITime . . . 733
operator = — IAcceleratorKey . . . 733
operator = — IAcceleratorTable . . . 733
operator = — IColor . . . 733
operator = — IFont . . . 733
operator = — IMMAudioBuffer — vbmm . . . 734
operator = — IMMAudioCDContents — vbmm . . . 734
operator = — IMMTime — vbmm . . . 734
operator = — IMngPointer — vbcc . . . 734
operator = — IPointArray . . . 734
operator = — IProfile . . . 734
operator = — IRecord — vbsample . . . 734
operator = — IResourceLibrary . . . 734
operator = — IStandardNotifier . . . 734
operator = — IString . . . 735
operator = — IVBagOnBase — vbcc . . . 735
operator = — IVDequeOnBase — vbcc . . . 735
operator = — IVEqualitySequenceOnBase — vbcc . . . 735
operator = — IVHeapOnBase — vbcc . . . 735
operator = — IVQueueOnBase — vbcc . . . 735
operator = — IVSequenceOnBase . . . 735
operator = — IVSetOnBase — vbcc . . . 735
operator = — IVSortedBagOnBase — vbcc . . . 736

operator = — IVSortedSetOnBase — vbcc 736
operator = — IVStackOnBase — vbcc 736
operator == — IABag — vbcc 736
operator == — IAcceleratorKey 736
operator == — IAddress — vbsample 736
operator == — IAEqualitySequence — vbcc 736
operator == — IASet — vbcc 736
operator == — IASortedBag — vbcc 736
operator == — IASortedSet — vbcc 737
operator == — IColor 737
operator == — ICompany — vbsample 737
operator == — IContainerControl 737
operator == — IContainerObject 737
operator == — ICustomer — vbsample 737
operator == — IDate 737
operator == — IMMTime — vbmm 737
operator == — IOrderedRecord — vbsample 737
operator == — IPair 738
operator == — IPointArray 738
operator == — IRectangle 738
operator == — ITime 738
operator == — IVBBooleanPart — vbsample 738
operator == — IVBDoublePart — vbsample 738
operator == — IVBLogicalAndPart — vbsample 738
operator == — IVBLogicalOrPart — vbsample 738
operator == — IVBLongPart — vbsample 738
operator == — IVBShortPart — vbsample 738
operator == — IVBStringPart — vbsample 738
operator == — IVBUnsignedLongPart — vbsample 738
operator == — IVBUnsignedShortPart — vbsample 739
operator > — IDate 739
operator > — IMMTime — vbmm 739
operator > — IOrderedRecord — vbsample 739
operator > — IPair 739
operator > — ITime 739
operator >= — IDate 739
operator >= — IMMTime — vbmm 739
operator >= — IOrderedRecord — vbsample 740
operator >= — IPair 740
operator >= — ITime 740
operator char * — IString 740
operator const char * — ICLibErrorInfo 740
operator const char * — IGUIErrorInfo 740

operator const char * — IMMErrorInfo — vbmm 740
operator const char * — ISystemErrorInfo 740
operator delete — IContainerObject 740
operator new — IContainerObject 741
operator signed char * — IString 741
operator unsigned char * — IString 741
operator unsigned long — IMMTime — vbmm 741
operator unsigned long — IResourceId 741
operator [] — IPointArray 741
operator [] — IString 741
operator ^ — IString 741
operator ^= — IString 741
operator | — IRectangle 742
operator | — IString 742
operator |= — IRectangle 742
operator |= — IString 742
operator ~ — IString 742
operator= — IResourceId 742
orValue — IVBLongPart — vbsample 742
orValue — IVBShortPart — vbsample 742
orValue — IVBUnsignedLongPart — vbsample 742
orValue — IVBUnsignedShortPart — vbsample 743
overlayWith — I0String 743
overlayWith — IString 743
pageSettings — INotebook 743
pagesToEnd — INotebook 743
pagesToMajorTab — INotebook 743
pagesToMinorTab — INotebook 743
parentObject — IContainerControl 744
paste — IEntryField 744
paste — IMMRecordable — vbmm 744
paste — IMultiLineEdit 744
pause — IMMPlayableDevice — vbmm 744
perror — stdioSamples — vbsample 744
play — IMMPlayableDevice — vbmm 744
playAt — IMMDigitalVideo — vbmm 745
playFast — IMMDigitalVideo — vbmm 745
playScan — IMMDigitalVideo — vbmm 745
playSlow — IMMDigitalVideo — vbmm 745
pop — IRStack — vbcc 745
positionBehindSibling — IWindow 745
positionBehindSiblings — IWindow 746
positionOnSiblings — IWindow 746

postEvent — IWindow . . . 746
pow — mathSamples — vbsample . . . 746
preappendText — IVBStringPart — vbsample . . . 746
previousPage — INotebook . . . 746
procAddress — IDynamicLinkLibrary . . . 746
push — IRStack — vbcc . . . 746
putc — stdioSamples — vbsample . . . 747
putchar — stdioSamples — vbsample . . . 747
putenv — stdlibSamples — vbsample . . . 747
puts — stdioSamples — vbsample . . . 747
putwc — stdioSamples — vbsample . . . 747
rand — stdlibSamples — vbsample . . . 747
read — ioSamples — vbsample . . . 747
realloc — stdlibSamples — vbsample . . . 747
reallocateString — ICnrEditHandler . . . 748
record — IMMDigitalVideo — vbmm . . . 748
record — IMMRecordable — vbmm . . . 748
refresh — IContainerObject . . . 748
refresh — IWindow . . . 748
refreshAllContainers — IContainerControl . . . 748
refreshTabs — INotebook . . . 748
registerFormat — IClipboard . . . 748
release — IMMDevice — vbmm . . . 749
releasePointer — IWindow . . . 749
releasePresSpace — IWindow . . . 749
remove — I0String . . . 749
remove — IAccelerator . . . 749
remove — IComboBox . . . 749
remove — IListBox . . . 749
remove — IPointArray . . . 749
remove — IRBag — vbcc . . . 750
remove — IREqualitySequence — vbcc . . . 750
remove — IRSet — vbcc . . . 750
remove — IRSortedBag — vbcc . . . 750
remove — IRSortedSet — vbcc . . . 750
remove — IString . . . 750
remove — IToolBar . . . 750
remove — stdioSamples — vbsample . . . 750
removeAll — IEntryField . . . 751
removeAll — IListBox . . . 751
removeAll — IMultiLineEdit . . . 751
removeAll — IRBag — vbcc . . . 751
removeAll — IRDeque — vbcc . . . 751

removeAll — IREqualitySequence — vbcc 751
removeAll — IRHeap — vbcc 751
removeAll — IRQueue — vbcc 751
removeAll — IRSequence 751
removeAll — IRSet — vbcc 751
removeAll — IRSortedBag — vbcc 752
removeAll — IRSortedSet — vbcc 752
removeAll — IRStack — vbcc 752
removeAll — ITextSpinButton 752
removeAllItems — IVBContainerControl 752
removeAllKeys — IAcceleratorTable 752
removeAllOccurrences — IRBag — vbcc 752
removeAllOccurrences — IREqualitySequence — vbcc 752
removeAllOccurrences — IRSet — vbcc 752
removeAllOccurrences — IRSortedBag — vbcc 753
removeAllOccurrences — IRSortedSet — vbcc 753
removeAllPages — INotebook 753
removeAtPosition — IREqualitySequence — vbcc 753
removeAtPosition — IRSequence 753
removeAtPosition — IRSortedBag — vbcc 753
removeAtPosition — IRSortedSet — vbcc 753
removeBorder — IBaseSpinButton 753
removeBorder — IPushButton 754
removeButton2Items — IVBContainerControl 754
removeColumn — IContainerControl 754
removeDetent — ISlider 754
removeExtension — IFrameWindow 754
removeFirst — IRDeque — vbcc 754
removeFirst — IREqualitySequence — vbcc 754
removeFirst — IRQueue — vbcc 754
removeFirst — IRSequence 754
removeFirst — IRSortedBag — vbcc 755
removeFirst — IRSortedSet — vbcc 755
removeFromCell — IMultiCellCanvas 755
removeFromWindowList — IFrameWindow 755
removeHelpText — IFlyOverHelpHandler 755
removeHelpText — IVBInfoArea 755
removeInUse — IContainerControl 755
removeInUse — IContainerObject 755
removeKeyLike — IAcceleratorTable 755
removeLast — IRDeque — vbcc 756
removeLast — IREqualitySequence — vbcc 756
removeLast — IRSequence 756

removeLast — IRSortedBag — vbcc 756
removeLast — IRSortedSet — vbcc 756
removeLast — IRStack — vbcc 756
removeLine — IMultiLineEdit 756
removePage — INotebook 756
removeSelectedItems — IVBContainerControl 756
removeTabSection — INotebook 757
removeWords — IString 757
rename — stdioSamples — vbsample 757
reset — IAccelerator 757
resetActiveColor — IWindow 757
resetActiveTextBackgroundColor — ITitle 757
resetActiveTextForegroundColor — ITitle 757
resetBackgroundColor — IWindow 757
resetBorderColor — IWindow 757
resetChangedFlag — IMultiLineEdit 758
resetDisabledBackgroundColor — IWindow 758
resetDisabledForegroundColor — IWindow 758
resetFillColor — IStaticText 758
resetFont — IWindow 758
resetForegroundColor — IWindow 758
resetHiliteBackgroundColor — IWindow 758
resetHiliteForegroundColor — IWindow 758
resetInactiveColor — IWindow 758
resetInactiveTextBackgroundColor — ITitle 759
resetInactiveTextForegroundColor — ITitle 759
resetLatchedBackgroundColor — ICustomButton 759
resetLatchedForegroundColor — ICustomButton 759
resetMajorTabBackgroundColor — INotebook 759
resetMajorTabForegroundColor — INotebook 759
resetMinimumSize — IWindow 759
resetMinorTabBackgroundColor — INotebook 759
resetMinorTabForegroundColor — INotebook 760
resetPageBackgroundColor — INotebook 760
resetShadowColor — IWindow 760
resetSplitBarEdgeColor — ISplitCanvas 760
resetSplitBarMiddleColor — ISplitCanvas 760
resetTransparentColor — IToolBarButton 760
resize — IPointArray 760
restore — IFrameWindow 760
resume — IMMPlayableDevice — vbmm 761
reverse — IPointArray 761
reverse — IString 761

reverseText — IVBStringPart — vbsample 761
rewind — stdioSamples — vbsample 761
rightJustify — IString 761
rowHeight — IMultiCellCanvas 761
rpmatch — stdlibSamples — vbsample 761
save — IMMRecordable — vbmm 762
saveAs — IMMRecordable — vbmm 762
saveHeadphonesSetting — IMMMasterAudio — vbmm 762
saveSpeakersSetting — IMMMasterAudio — vbmm 762
saveVolume — IMMMasterAudio — vbmm 762
scaleBy — IPair 762
scaleBy — IRectangle 762
scaledBy — IPair 762
scaledBy — IRectangle 763
scroll — IContainerControl 763
scrollDetailsHorizontally — IContainerControl 763
scrollHorizontally — IContainerControl 763
scrollToObject — IContainerControl 763
scrollVertically — IContainerControl 763
scrollViewHorizontallyTo — IViewPort 763
scrollViewVerticallyTo — IViewPort 763
seek — IMMPlayableDevice — vbmm 764
seekToEnd — IMMPlayableDevice — vbmm 764
seekToStart — IMMPlayableDevice — vbmm 764
select — IVBContainerControl 764
selectAll — IBaseListBox 764
selectedCnrElements — IVBContainerControl 764
selectedElements — ICollectionViewListBox 764
selectedElements — IVBContainerControl 764
selectHalftone — I3StateCheckBox 765
selectRange — IEntryField 765
selectRange — IMultiLineEdit 765
sendEvent — IWindow 765
set — IAccelerator 765
setActiveWindow — IHelpWindow 765
setAddressToDefault — ICompany — vbsample 765
setAddressToDefault — ICustomer — vbsample 765
setAllEmphasis — IFont 766
setAssociatedWindow — IHelpWindow 766
setBitmaps — IAnimatedButton 766
setBorderSize — IFrameWindow 766
setbuf — stdioSamples — vbsample 766
setChangedFlag — IEntryField 766

setChangedFlag — IMultiLineEdit . . . 766
setCharHeight — IFont . . . 766
setCharSize — IFont . . . 767
setCharWidth — IFont . . . 767
setCityToDefault — IAddress — vbsample . . . 767
setClosed — IContainerObject . . . 767
setColumnWidth — IMultiCellCanvas . . . 767
setCursor — IContainerControl . . . 767
setCustomerListToDefault — ICompany — vbsample . . . 767
setData — IClipboard . . . 767
setDataOffset — IContainerColumn . . . 768
setDecrementBitmaps — ICircularSlider . . . 768
setDeleteColumnsOnClose — IContainerControl . . . 768
setDeleteObjectsOnClose — IContainerControl . . . 768
setDestroyOnClose — IFrameWindow . . . 768
setDialogTemplate — IVBFileDialog . . . 768
setDialogTemplate — IVBFontDialog . . . 768
setDirection — IFont . . . 769
setDisplayPS — IVBFontDialog . . . 769
setEditColumn — IContainerControl . . . 769
setEditMLE — IContainerControl . . . 769
setEditObject — IContainerControl . . . 769
setEditRegion — IMultiLineEdit . . . 769
setExtendedSelection — IContainerControl . . . 769
setExtensionSize — IFrameWindow . . . 769
setFamily — IVBFontDialog . . . 770
setFocus — IWindow . . . 770
setFontAngle — IFont . . . 770
setFontFromFontModally — IVBFontDialog . . . 770
setFontFromWindowModally — IVBFontDialog . . . 770
setFontShear — IFont . . . 770
setHandle — IClipboard . . . 770
setHelpKey — IAcceleratorKey . . . 771
setHelpTable — IHelpWindow . . . 771
setHelpText — IFlyOverHelpHandler . . . 771
setHelpText — IVBInfoArea . . . 771
setHomePhoneToDefault — ICustomer — vbsample . . . 771
setIncrementBitmaps — ICircularSlider . . . 771
setInitialDrive — IVBFileDialog . . . 771
setInitialFileType — IVBFileDialog . . . 771
setInput1 — IVBLogicalAndPart — vbsample . . . 772
setInput1 — IVBLogicalOrPart — vbsample . . . 772
setInput10 — IVBLogicalAndPart — vbsample . . . 772

setInput10 — IVBLogicalOrPart — vbsample 772
setInput2 — IVBLogicalAndPart — vbsample 772
setInput2 — IVBLogicalOrPart — vbsample 772
setInput3 — IVBLogicalAndPart — vbsample 772
setInput3 — IVBLogicalOrPart — vbsample 772
setInput4 — IVBLogicalAndPart — vbsample 772
setInput4 — IVBLogicalOrPart — vbsample 772
setInput5 — IVBLogicalAndPart — vbsample 773
setInput5 — IVBLogicalOrPart — vbsample 773
setInput6 — IVBLogicalAndPart — vbsample 773
setInput6 — IVBLogicalOrPart — vbsample 773
setInput7 — IVBLogicalAndPart — vbsample 773
setInput7 — IVBLogicalOrPart — vbsample 773
setInput8 — IVBLogicalAndPart — vbsample 773
setInput8 — IVBLogicalOrPart — vbsample 773
setInput9 — IVBLogicalAndPart — vbsample 773
setInput9 — IVBLogicalOrPart — vbsample 773
setInUse — IContainerControl 774
setItemHandle — IBaseComboBox 774
setItemHandle — IBaseListBox 774
setItemHeight — IBaseListBox 774
setItemText — IBaseComboBox 774
setItemText — IBaseListBox 774
setKey — IAcceleratorKey 774
setLayoutDistorted — IWindow 774
setMajorTabSize — INotebook 775
setMaster — IBaseSpinButton 775
setMinorTabSize — INotebook 775
setMixedTargetEmphasis — IContainerControl 775
setMLEHandler — ICnrEditHandler 775
setMultipleSelection — IContainerControl 775
setNameToDefault — ICompany — vbsample 775
setNameToDefault — ICustomer — vbsample 775
setNormalTargetEmphasis — IContainerControl 775
setNotInput1 — IVBLogicalAndPart — vbsample 776
setNotInput1 — IVBLogicalOrPart — vbsample 776
setNotInput10 — IVBLogicalAndPart — vbsample 776
setNotInput10 — IVBLogicalOrPart — vbsample 776
setNotInput2 — IVBLogicalAndPart — vbsample 776
setNotInput2 — IVBLogicalOrPart — vbsample 776
setNotInput3 — IVBLogicalAndPart — vbsample 776
setNotInput3 — IVBLogicalOrPart — vbsample 776
setNotInput4 — IVBLogicalAndPart — vbsample 776

setNotInput4 — IVBLogicalOrPart — vbsample 776
setNotInput5 — IVBLogicalAndPart — vbsample 777
setNotInput5 — IVBLogicalOrPart — vbsample 777
setNotInput6 — IVBLogicalAndPart — vbsample 777
setNotInput6 — IVBLogicalOrPart — vbsample 777
setNotInput7 — IVBLogicalAndPart — vbsample 777
setNotInput7 — IVBLogicalOrPart — vbsample 777
setNotInput8 — IVBLogicalAndPart — vbsample 777
setNotInput8 — IVBLogicalOrPart — vbsample 777
setNotInput9 — IVBLogicalAndPart — vbsample 777
setNotInput9 — IVBLogicalOrPart — vbsample 777
setOKButtonText — IVBFileDialog 778
setOpenDialog — IVBFileDialog 778
setOrderedTargetEmphasis — IContainerControl 778
setPageButtonSize — INotebook 778
setPhoneToDefault — ICompany — vbsample 778
setPosition — IVBFileDialog 778
setPosition — IVBFontDialog 778
setPreviewText — IVBFontDialog 778
setPrinterPS — IVBFontDialog 778
setProgram — IMMAudioCD — vbmm 779
setRefreshOff — IContainerControl 779
setRefreshOff — IContainerObject 779
setRowHeight — IMultiCellCanvas 779
setSaveAsDialog — IVBFileDialog 779
setScrollBar — IScrollBar 779
setSelected — IContainerControl 779
setSeparator — IMenuItem 779
setShaftBreadth — IProgressIndicator 780
setSingleSelection — IContainerControl 780
setSizeList — IVBFontDialog 780
setSplitBarThickness — ISplitCanvas 780
setSplitWindowPercentage — ISplitCanvas 780
setStateToDefault — IAddress — vbsample 780
setStatusText — INotebook 780
setStreetToDefault — IAddress — vbsample 780
setTab — IMultiLineEdit 780
setTabBitmap — INotebook 781
setTabText — INotebook 781
setTickLength — IProgressIndicator 781
setTicks — IProgressIndicator 781
setTickText — IProgressIndicator 781
setTitle — IHelpWindow 781

setTitle — IMessageBox . . . 781
setTitle — IVBFileDialog . . . 781
setTitle — IVBFontDialog . . . 782
setTitleAlignment — IContainerControl . . . 782
setTitleText — ITitle . . . 782
setToDefault — IAddress — vbsample . . . 782
setTrackTitle — IMMAudioCD — vbmm . . . 782
setTreeExpandIconSize — IContainerControl . . . 782
setTreeItemIcons — IContainerControl . . . 782
setTreeViewIndent — IContainerControl . . . 782
setUserData — INotebook . . . 783
setUsingHelp — IHelpWindow . . . 783
setvbuf — stdioSamples — vbsample . . . 783
setWindow — IMMDigitalVideo — vbmm . . . 783
setWindow — INotebook . . . 783
setWindowFont — IFont . . . 783
setWorkPhoneToDefault — ICustomer — vbsample . . . 783
setZipToDefault — IAddress — vbsample . . . 783
show — IHelpWindow . . . 784
show — IMessageBox . . . 784
show — IVBFileDialog . . . 784
show — IVBFontDialog . . . 784
show — IWindow . . . 784
showContentsHelp — IHelpWindow . . . 784
showCustomized — IMessageBox . . . 784
showDetailsView — IContainerControl . . . 784
showErrorInfo — IMessageBox . . . 785
showException — IMessageBox . . . 785
showFlowedNameView — IContainerControl . . . 785
showFlowedTextView — IContainerControl . . . 785
showFromFont — IVBFontDialog . . . 785
showGeneralHelp — IHelpWindow . . . 785
showHelpPanel — IHelpWindow . . . 785
showIconView — IContainerControl . . . 785
showIndexHelp — IHelpWindow . . . 785
showKeysHelp — IHelpWindow . . . 786
showList — IBaseComboBox . . . 786
showMiniIcons — IContainerControl . . . 786
showModally — IFrameWindow . . . 786
showModally — IVBFileDialog . . . 786
showModally — IVBFontDialog . . . 786
showNameView — IContainerControl . . . 786
showObject — IContainerControl . . . 786

showPanelIds — IHelpWindow . . . 787
showSeparators — IContainerColumn . . . 787
showSourceEmphasis — IWindow . . . 787
showSplitBar — IContainerControl . . . 787
showTextView — IContainerControl . . . 787
showTreeIconView — IContainerControl . . . 787
showTreeLine — IContainerControl . . . 787
showTreeNameView — IContainerControl . . . 787
showTreeTextView — IContainerControl . . . 788
showUsingHelp — IHelpWindow . . . 788
shrinkBy — IRectangle . . . 788
shrunkBy — IRectangle . . . 788
sin — mathSamples — vbsample . . . 788
sinh — mathSamples — vbsample . . . 788
sizeBy — IRectangle . . . 788
sizedBy — IRectangle . . . 788
sizedTo — IRectangle . . . 789
sort — IREqualitySequence — vbcc . . . 789
sort — IRSequence . . . 789
sortByIconText — IContainerControl . . . 789
space — IString . . . 789
spinDown — IBaseSpinButton . . . 789
spinTo — INumericSpinButton . . . 789
spinTo — ITextSpinButton . . . 789
spinUp — IBaseSpinButton . . . 790
splitBarThickness — ISplitCanvas . . . 790
splitWindowPercentage — ISplitCanvas . . . 790
sqrt — mathSamples — vbsample . . . 790
squareValue — IVBDoublePart — vbsample . . . 790
squareValue — IVBLongPart — vbsample . . . 790
squareValue — IVBShortPart — vbsample . . . 790
squareValue — IVBUnsignedLongPart — vbsample . . . 790
squareValue — IVBUnsignedShortPart — vbsample . . . 790
srand — stdlibSamples — vbsample . . . 791
start — IFrameWindow . . . 791
start — IHandler . . . 791
start — IInfoArea . . . 791
startAnimation — IAnimatedButton . . . 791
startPositionTracking — IMMPlayableDevice — vbmm . . . 791
startScanningBackward — IMMAudioCD — vbmm . . . 791
startScanningForward — IMMAudioCD — vbmm . . . 791
stepFrame — IMMPlayableDevice — vbmm . . . 792
stop — IHandler . . . 792

stop — IInfoArea . . . 792
stop — IMMPlayableDevice — vbmm . . . 792
stopAnimation — IAnimatedButton . . . 792
stopPositionTracking — IMMPlayableDevice — vbmm . . . 792
strip — IString . . . 792
stripBlanks — IString . . . 792
stripBlanksOnText — IVBStringPart — vbsample . . . 793
stripLeading — IString . . . 793
stripLeadingBlanks — IString . . . 793
stripTrailing — IString . . . 793
stripTrailingBlanks — IString . . . 793
strtod — stdlibSamples — vbsample . . . 793
strtol — stdlibSamples — vbsample . . . 793
strtold — stdlibSamples — vbsample . . . 793
subString — IString . . . 793
subtractValue — IVBDoublePart — vbsample . . . 793
subtractValue — IVBLongPart — vbsample . . . 794
subtractValue — IVBShortPart — vbsample . . . 794
subtractValue — IVBUnsignedLongPart — vbsample . . . 794
subtractValue — IVBUnsignedShortPart — vbsample . . . 794
supportsCommand — IMMDevice — vbmm . . . 794
swab — stdlibSamples — vbsample . . . 794
system — stdlibSamples — vbsample . . . 794
tan — mathSamples — vbsample . . . 794
tanh — mathSamples — vbsample . . . 794
tempnam — stdioSamples — vbsample . . . 795
textLines — IFont . . . 795
textRectangle — IContainerControl . . . 795
textWidth — IFont . . . 795
throwCLibError — ICLibErrorInfo . . . 795
throwError — IErrorInfo . . . 795
throwGUIError — IGUIErrorInfo . . . 795
throwMMError — IMMErrorInfo — vbmm . . . 796
throwSystemError — ISystemErrorInfo . . . 796
tickLength — IProgressIndicator . . . 796
tickPosition — IProgressIndicator . . . 796
tickSpacing — IProgressIndicator . . . 796
tickText — IProgressIndicator . . . 796
tmpfile — stdioSamples — vbsample . . . 796
tmpnam — stdioSamples — vbsample . . . 796
trackBackward — IMMAudioCD — vbmm . . . 797
trackForward — IMMAudioCD — vbmm . . . 797
trackTitle — IMMAudioCD — vbmm . . . 797

translate — IString . . . 797
transpose — IPair . . . 797
tryToLoadBitmap — IResourceLibrary . . . 797
tryToLoadIcon — IResourceLibrary . . . 797
tryToLoadMessage — IResourceLibrary . . . 797
tryToLoadString — IResourceLibrary . . . 798
turnToPage — INotebook . . . 798
umask — ioSamples — vbsample . . . 798
undo — IMultiLineEdit . . . 798
undo — ISubmenu . . . 798
ungetc — stdioSamples — vbsample . . . 798
ungetwc — stdioSamples — vbsample . . . 798
unhighlight — IButton . . . 798
unionWith — IRBag — vbcc . . . 799
unionWith — IRSet — vbcc . . . 799
unionWith — IRSortedBag — vbcc . . . 799
unionWith — IRSortedSet — vbcc . . . 799
unlatch — ICustomButton . . . 799
unlink — stdioSamples — vbsample . . . 799
update — IAcceleratorTable . . . 799
update — IFrameWindow . . . 799
upperCase — IString . . . 800
useBitmapOnly — IFont . . . 800
useDefaultWindow — IMMDigitalVideo — vbmm . . . 800
useExtensionMinimumSize — IFrameWindow . . . 800
useNonPropOnly — IFont . . . 800
useVectorOnly — IFont . . . 800
wcsid — stdlibSamples — vbsample . . . 800
wcstombs — stdlibSamples — vbsample . . . 801
wctob — stdioSamples — vbsample . . . 801
wctomb — stdlibSamples — vbsample . . . 801
window — INotebook . . . 801
windowInCell — IMultiCellCanvas . . . 801
word — IString . . . 801
wordIndexOfPhrase — IString . . . 801
words — IString . . . 801
write — ioSamples — vbsample . . . 802
write — ITrace . . . 802
writeToQueue — ITrace . . . 802
writeToStandardError — ITrace . . . 802
writeToStandardOutput — ITrace . . . 802
x2b — IString . . . 802
x2c — IString . . . 802

x2d — IString . . . 802
_alloca — stdlibSamples — vbsample . . . 802
_atold — stdlibSamples — vbsample . . . 803
_cabs — mathSamples — vbsample . . . 803
_chsize — ioSamples — vbsample . . . 803
_clear87 — floatSamples — vbsample . . . 803
_control87 — floatSamples — vbsample . . . 803
_crotl — stdlibSamples — vbsample . . . 803
_debug_calloc — stdlibSamples — vbsample . . . 803
_debug_free — stdlibSamples — vbsample . . . 803
_debug_heapmin — stdlibSamples — vbsample . . . 804
_debug_malloc — stdlibSamples — vbsample . . . 804
_debug_realloc — stdlibSamples — vbsample . . . 804
_debug_tcalloc — stdlibSamples — vbsample . . . 804
_debug_tfree — stdlibSamples — vbsample . . . 804
_debug_theapmin — stdlibSamples — vbsample . . . 804
_debug_tmalloc — stdlibSamples — vbsample . . . 804
_debug_trealloc — stdlibSamples — vbsample . . . 804
_dump_allocated_delta — stdlibSamples — vbsample . . . 804
_ecvt — stdlibSamples — vbsample . . . 805
_endthread — stdlibSamples — vbsample . . . 805
_exit — stdlibSamples — vbsample . . . 805
_fcloseall — stdioSamples — vbsample . . . 805
_fcvt — stdlibSamples — vbsample . . . 805
_fgetchar — stdioSamples — vbsample . . . 805
_filelength — ioSamples — vbsample . . . 805
_flushall — stdioSamples — vbsample . . . 805
_fpreset — floatSamples — vbsample . . . 806
_fputchar — stdioSamples — vbsample . . . 806
_freemod — stdlibSamples — vbsample . . . 806
_fullpath — stdlibSamples — vbsample . . . 806
_gcvt — stdlibSamples — vbsample . . . 806
_heapmin — stdlibSamples — vbsample . . . 806
_heap_check — stdlibSamples — vbsample . . . 806
_itoa — stdlibSamples — vbsample . . . 807
_j0 — mathSamples — vbsample . . . 807
_loadmod — stdlibSamples — vbsample . . . 807
_lrotl — stdlibSamples — vbsample . . . 807
_ltoa — stdlibSamples — vbsample . . . 807
_makepath — stdlibSamples — vbsample . . . 807
_matherr — mathSamples — vbsample . . . 807
_msize — stdlibSamples — vbsample . . . 807
_onexit — stdlibSamples — vbsample . . . 808

_rmtmp — stdioSamples — vbsample . . . 808
_rotl — stdlibSamples — vbsample . . . 808
_searchenv — stdlibSamples — vbsample . . . 808
_setmode — ioSamples — vbsample . . . 808
_set_crt_msg_handle — stdioSamples — vbsample . . . 808
_sopen — ioSamples — vbsample . . . 808
_splitpath — stdlibSamples — vbsample . . . 809
_status87 — floatSamples — vbsample . . . 809
_tcalloc — stdlibSamples — vbsample . . . 809
_tdump_allocated_delta — stdlibSamples — vbsample . . . 809
_tell — ioSamples — vbsample . . . 809
_tfree — stdlibSamples — vbsample . . . 809
_threadstore — stdlibSamples — vbsample . . . 809
_tmalloc — stdlibSamples — vbsample . . . 809
_trealloc — stdlibSamples — vbsample . . . 810
_ultoa — stdlibSamples — vbsample . . . 810
__eof — ioSamples — vbsample . . . 810
__parmdwords — stdlibSamples — vbsample . . . 810

Chapter 165. Attributes . . . 811
acquired (Boolean) — IMMDevice — vbmm . . . 811
actionType (ICommand::ActionType) — IAcceleratorKey . . . 811
activeColor (IColor) — IWindow . . . 811
activeTextBackgroundColor (IColor) — ITitle . . . 811
activeTextForegroundColor (IColor) — ITitle . . . 812
address (IAddress*) — ICompany — vbsample . . . 812
address (IAddress*) — ICustomer — vbsample . . . 812
aliasName (IString) — IMMDevice — vbmm . . . 812
alignment (IBaseSpinButton::Alignment) — IBaseSpinButton . . . 812
alignment (IEntryField::Alignment) — IEntryField . . . 812
alignment (IProgressIndicator::Alignment) — IProgressIndicator . . . 812
alignment (ISetCanvas::Alignment) — ISetCanvas . . . 813
alignment (IStaticText::Alignment) — IStaticText . . . 813
allowsDragDelete (Boolean) — IToolBarButton . . . 813
allowsDragDrop (Boolean) — IToolBar . . . 813
allowsMouseClickFocus (Boolean) — IButton . . . 813
alphabetic (Boolean) — IString . . . 813
alphanumeric (Boolean) — IString . . . 813
animatedWhenLatched (Boolean) — IAnimatedButton . . . 814
animationRate (unsigned long) — IAnimatedButton . . . 814
animationStarted (Boolean) — IAnimatedButton . . . 814
anyElement (Element const) — IRBag — vbcc . . . 814
anyElement (Element const) — IRDeque — vbcc . . . 814

anyElement (Element const) — IREqualitySequence — vbcc 814
anyElement (Element const) — IRHeap — vbcc 814
anyElement (Element const) — IRQueue — vbcc 814
anyElement (Element const) — IRSequence 815
anyElement (Element const) — IRSet — vbcc 815
anyElement (Element const) — IRSortedBag — vbcc 815
anyElement (Element const) — IRSortedSet — vbcc 815
anyElement (Element const) — IRStack — vbcc 815
area (IRectangle::Coord) — IRectangle 815
areDetailsViewTitlesVisible (Boolean) — IContainerControl 815
areHeadphonesEnabled (Boolean) — IMMMasterAudio — vbmm 815
areSpeakersEnabled (Boolean) — IMMMasterAudio — vbmm 816
armPixelOffset (unsigned long) — IProgressIndicator 816
armRange (IRange) — ICircularSlider 816
armRange (unsigned long) — IProgressIndicator 816
armSize (ISize) — ISlider 816
armTickOffset (unsigned long) — IProgressIndicator 817
asACCEL (ACCEL) — IAcceleratorKey 817
asDebugInfo (IString) — IBase 817
asDebugInfo (IString) — IInfoArea 817
asDouble (double) — IString 817
asInt (long) — IString 817
asMMTime (unsigned long) — IMMTime — vbmm 817
asPOINTL (struct _POINTL) — IPoint 817
asRECTL (struct _RECTL) — IRectangle 818
asRGBLong (long) — IColor 818
asSeconds (long) — ITime 818
asSIZEL (SIZEL) — ISize 818
asString (IString) — IBase 818
asString (IString) — IInfoArea 818
asUnsigned (unsigned long) — IString 818
audioEnabled (Boolean) — IMMDevice — vbmm 819
autoDeleteObject (Boolean) — IWindow 819
autoDestroyWindow (Boolean) — IWindow 819
autoLatchEnabled (Boolean) — ICustomButton 819
autoPlayEnabled (Boolean) — IMMAudioCD — vbmm 819
autoScroll (Boolean) — IEntryField 820
autoSelect (Boolean) — I3StateCheckBox 820
autoSelect (Boolean) — ICheckBox 820
autoSelect (Boolean) — IRadioButton 820
autoTab (Boolean) — IEntryField 820
available (Boolean) — ICLibErrorInfo 820
available (Boolean) — IGUIErrorInfo 820

available (Boolean) — IMMErrorInfo — vbmm . . . 821
available (Boolean) — ISystemErrorInfo . . . 821
avgCharWidth (unsigned long) — IFont . . . 821
backgroundColor (IColor) — IWindow . . . 821
balance (unsigned long) — IMMAmpMixer — vbmm . . . 821
bass (unsigned long) — IMMAmpMixer — vbmm . . . 821
bidiSupported (Boolean) — IWindow . . . 821
binaryDigits (Boolean) — IString . . . 822
binding (INotebook::Binding) — INotebook . . . 822
bitmap (IBitmapHandle) — IAnimatedButton . . . 822
bitmap (IBitmapHandle) — IBitmapControl . . . 822
bitmap (IBitmapHandle) — IClipboard . . . 822
bitmap (Boolean) — IFont . . . 822
bitmap (IBitmapHandle) — IGraphicPushButton . . . 822
bitmap (IBitmapHandle) — IMenuItem . . . 823
bitmap (IBitmapHandle) — IToolBarButton . . . 823
bitmapCount (unsigned long) — IAnimatedButton . . . 823
bitmapOnly (Boolean) — IFont . . . 823
bitmapSize (ISize) — IToolBarButton . . . 823
bitmapVisible (Boolean) — IToolBarButton . . . 823
bitsPerSample (unsigned long) — IMMConfigurableAudio — vbmm . . . 823
blockAlignment (unsigned long) — IMMConfigurableAudio — vbmm . . . 824
blueMix (unsigned char) — IColor . . . 824
bold (Boolean) — IFont . . . 824
border (Boolean) — IBaseSpinButton . . . 824
border (Boolean) — IPushButton . . . 824
borderColor (IColor) — IWindow . . . 824
borderHeight (unsigned long) — IFrameWindow . . . 825
borderSize (ISize) — IFrameWindow . . . 825
borderWidth (unsigned long) — IFrameWindow . . . 825
bottom (IRectangle::Coord) — IRectangle . . . 825
bottomCenter (IPoint) — IRectangle . . . 825
bottomLeft (IPoint) — IRectangle . . . 825
bottomRight (IPoint) — IRectangle . . . 825
bounded (IBoolean) — IRBag — vbcc . . . 826
bounded (IBoolean) — IRDeque — vbcc . . . 826
bounded (IBoolean) — IREqualitySequence — vbcc . . . 826
bounded (IBoolean) — IRHeap — vbcc . . . 826
bounded (IBoolean) — IRQueue — vbcc . . . 826
bounded (IBoolean) — IRSequence . . . 826
bounded (IBoolean) — IRSet — vbcc . . . 826
bounded (IBoolean) — IRSortedBag — vbcc . . . 826
bounded (IBoolean) — IRSortedSet — vbcc . . . 826

bounded (IBoolean) — IRStack — vbcc 827
button2CnrItems (IVSequence<CnrElement*>*) — IVBContainerControl . 827
button2CollectionPosition (unsigned long) — IVBContainerControl 827
button2CollectionPositions (IVSequence<unsigned long>*) — IVBContainerControl 827
button2Item (Element) — IVBContainerControl 827
button2Items (IVSequence<Element>*) — IVBContainerControl 827
button2Point (IPoint) — IVBContainerControl 828
buttonPressedId (long) — IVBFileDialog 828
buttonPressedId (unsigned long) — IVBFontDialog 828
buttonsPosition (ISlider::ButtonsPosition) — ISlider 828
buttonView (IToolBarButton::View) — IToolBar 828
bytesPerSecond (unsigned long) — IMMConfigurableAudio — vbmm . . . 828
cachingEnabled (Boolean) — IContainerControl 829
center (IPoint) — IRectangle 829
centerXCenterY (IPoint) — IRectangle 829
centerXMaxY (IPoint) — IRectangle 829
centerXMinY (IPoint) — IRectangle 829
changed (Boolean) — IEntryField 829
changed (Boolean) — IMultiLineEdit 829
channels (unsigned long) — IMMConfigurableAudio — vbmm 830
character (IString) — IAcceleratorKey 830
characterSize (ISize) — IWindow 830
charType (IEntryField::CharType) — IEntryField 830
checked (Boolean) — IMenuItem 830
city (IString) — IAddress — vbsample 830
client (IWindow*) — IFrameWindow 831
clientHandle (IWindowHandle) — IFrameWindow 831
clipboardHasTextFormat (Boolean) — ITextControl 831
closeOnDestroy (Boolean) — IMMDevice — vbmm 831
columnCount (unsigned long) — IContainerControl 831
command (Boolean) — IEntryField 831
commandId (ICommand::CommandId) — IAcceleratorKey 832
commandType (IMenuItem::CommandType) — IMenuItem 832
communicationWindow (IWindowHandle) — IHelpWindow 832
consistent (IBoolean) — IRBag — vbcc 832
consistent (IBoolean) — IRDeque — vbcc 832
consistent (IBoolean) — IREqualitySequence — vbcc 832
consistent (IBoolean) — IRHeap — vbcc 832
consistent (IBoolean) — IRQueue — vbcc 832
consistent (IBoolean) — IRSequence 832
consistent (IBoolean) — IRSet — vbcc 832
consistent (IBoolean) — IRSortedBag — vbcc 832

consistent (IBoolean) — IRSortedSet — vbcc 833
consistent (IBoolean) — IRStack — vbcc 833
contents (IMMAudioCDContents) — IMMAudioCD — vbmm 833
contents (IString) — IString 833
contentsWindow (IWindowHandle) — IHelpWindow 833
continuousPlayEnabled (Boolean) — IMMAudioCD — vbmm 833
control (Boolean) — IString 833
coord1 (IPair::Coord) — IPair 833
coord2 (IPair::Coord) — IPair 834
count (unsigned long) — IBaseComboBox 834
count (unsigned long) — IBaseListBox 834
coverPageWindow (IWindowHandle) — IHelpWindow 834
currentBitmapIndex (unsigned long) — IAnimatedButton 834
currentEditColumn (IContainerColumn*) — IContainerControl 834
currentEditMLE (IMultiLineEdit*) — IContainerControl 834
currentEditObject (IContainerObject*) — IContainerControl 834
currentGraphicType (IGraphicPushButton::GraphicType) —
IGraphicPushButton 835
cursoredCollectionPosition (unsigned long) — IVBContainerControl 835
cursoredItem (Element) — IVBContainerControl 835
cursoredObject (IContainerObject*) — IContainerControl 835
cursorLinePosition (unsigned long) — IMultiLineEdit 835
cursorPosition (unsigned long) — IEntryField 835
cursorPosition (unsigned long) — IMultiLineEdit 836
cursorSelect (Boolean) — IRadioButton 836
customerList (IVSequence<ICustomer*>*) — ICompany — vbsample . . . 836
dataIString (Boolean) — ICnrEditHandler 836
date (Boolean) — IContainerColumn 836
date (IDate) — ICustomer — vbsample 837
dayOfMonth (int) — IDate 837
dayOfWeek (IDate::DayOfWeek) — IDate 837
dayOfYear (int) — IDate 837
deckCount (unsigned long) — ISetCanvas 837
deckOrientation (ISetCanvas::DeckOrientation) — ISetCanvas 837
default (Boolean) — IPushButton 838
defaultApplicationName (IString) — IProfile 838
defaultCell (ISize) — IMultiCellCanvas 838
defaultGroupPad (unsigned long) — IToolBar 838
defaultInput (Boolean) — IVBLogicalAndPart — vbsample 838
defaultInput (Boolean) — IVBLogicalOrPart — vbsample 838
defaultMargin (ISize) — IToolBar 838
defaultMisfitWidth (unsigned long) — IToolBar 839
defaultOrdering (IWindow::SiblingOrder) — IFrameWindow 839

defaultPad (long) — IToolBar 839
defaultPushButton (IWindowHandle) — IWindow 839
defaultText (IString) — IFlyOverHelpHandler 839
defaultText (IString) — IVBStringPart — vbsample 840
defaultTransparentColor (IColor) — IToolBarButton 840
defaultTransparentColorSet (Boolean) — IToolBarButton 840
defaultValue (Boolean) — IVBBooleanPart — vbsample 840
defaultValue (double) — IVBDoublePart — vbsample 840
defaultValue (long) — IVBLongPart — vbsample 841
defaultValue (short) — IVBShortPart — vbsample 841
defaultValue (unsigned long) — IVBUnsignedLongPart — vbsample 841
defaultValue (unsigned short) — IVBUnsignedShortPart — vbsample . . . 841
delayTime (unsigned long) — IFlyOverHelpHandler 841
description (IString) — IMMDevice — vbmm 841
destinationRectangle (IRectangle) — IMMDigitalVideo — vbmm 842
detailsTitleRectangle (IRectangle) — IContainerControl 842
detailsView (Boolean) — IContainerControl 842
detailsViewPortOnWindow (IRectangle) — IContainerControl 842
detailsViewPortOnWorkspace (IRectangle) — IContainerControl 842
detailsViewSplit (IContainerColumn*) — IContainerControl 842
deviceId (unsigned long) — IMMDevice — vbmm 843
deviceName (IString) — IMMDevice — vbmm 843
deviceType (unsigned long) — IMMDevice — vbmm 843
deviceType (unsigned long) — IMMPlayerPanel — vbmm 843
digits (Boolean) — IString 843
digits (Boolean) — IVBStringPart — vbsample 843
disabled (Boolean) — IMenuItem 843
disabledBackgroundColor (IColor) — IWindow 844
disabledForegroundColor (IColor) — IWindow 844
disabledText (IString) — IInfoArea 844
discId (IString) — IMMAudioCD — vbmm 844
discId (IString) — IMMAudioCDContents — vbmm 844
discTitle (IString) — IMMAudioCD — vbmm 844
displaySize (ISize) — ITextControl 844
displayWidth (unsigned long) — IContainerColumn 845
dragLines (Boolean) — IMultiCellCanvas 845
drawBackgroundEnabled (Boolean) — IContainerControl 845
drawItem (Boolean) — IBaseListBox 845
drawItem (Boolean) — IMenuItem 845
drawItemEnabled (Boolean) — IContainerControl 845
drawItemEnabled (Boolean) — IProgressIndicator 846
dropOnAble (Boolean) — IContainerObject 846
editRegionHeight (unsigned long) — IMultiLineEdit 846

editRegionWidth (unsigned long) — IMultiLineEdit 846
empty (Boolean) — IBaseListBox 846
empty (Boolean) — IEntryField 846
empty (Boolean) — INotebook 846
empty (IBoolean) — IRBag — vbcc 847
empty (IBoolean) — IRDeque — vbcc 847
empty (IBoolean) — IREqualitySequence — vbcc 847
empty (IBoolean) — IRHeap — vbcc 847
empty (IBoolean) — IRQueue — vbcc 847
empty (IBoolean) — IRSequence 847
empty (IBoolean) — IRSet — vbcc 847
empty (IBoolean) — IRSortedBag — vbcc 847
empty (IBoolean) — IRSortedSet — vbcc 847
empty (IBoolean) — IRStack — vbcc 848
enabled (Boolean) — IHandler 848
enabled (Boolean) — IInfoArea 848
enabled (Boolean) — IWindow 848
enabledForNotification (Boolean) — IRBag — vbcc 848
enabledForNotification (Boolean) — IRDeque — vbcc 848
enabledForNotification (Boolean) — IREqualitySequence — vbcc 849
enabledForNotification (Boolean) — IRHeap — vbcc 849
enabledForNotification (Boolean) — IRQueue — vbcc 849
enabledForNotification (Boolean) — IRSequence 849
enabledForNotification (Boolean) — IRSet — vbcc 849
enabledForNotification (Boolean) — IRSortedBag — vbcc 849
enabledForNotification (Boolean) — IRSortedSet — vbcc 850
enabledForNotification (Boolean) — IRStack — vbcc 850
enabledForNotification (Boolean) — IStandardNotifier 850
enabledForNotification (Boolean) — IWindow 850
errorId (unsigned long) — ICLibErrorInfo 850
errorId (unsigned long) — IGUIErrorInfo 850
errorId (unsigned long) — IMMErrorInfo — vbmm 850
errorId (unsigned long) — ISystemErrorInfo 851
expanded (Boolean) — IToolBar 851
extendedSelect (Boolean) — IBaseListBox 851
extendedSelection (Boolean) — IContainerControl 851
externalLeading (unsigned long) — IFont 851
fastForwardButton (IAnimatedButton*) — IMMPlayerPanel — vbmm . . . 851
fastSpinEnabled (Boolean) — IBaseSpinButton 851
fileDialog (IFileDialog*) — IVBFileDialog 851
fileDialogSettings (IFileDialog::Settings*) — IVBFileDialog 852
filename (IString) — IMMFileMedia — vbmm 852
fileName (IString) — IResourceLibrary 852

fileName (IString) — IVBFileDialog 852
fileNormalSpeed (IMMSpeed) — IMMDigitalVideo — vbmm 852
fillBackground (Boolean) — IStaticText 853
fillColor (IColor) — IStaticText 853
firstElement (Element const) — IRDeque — vbcc 853
firstElement (Element const) — IREqualitySequence — vbcc 853
firstElement (Element const) — IRQueue — vbcc 853
firstElement (Element const) — IRSequence 853
firstElement (Element const) — IRSortedBag — vbcc 853
firstElement (Element const) — IRSortedSet — vbcc 853
firstElement (Element const) — IRStack — vbcc 854
firstPage (IPageHandle) — INotebook 854
fixed (Boolean) — IFont 854
flashing (Boolean) — IFrameWindow 854
floatingFrame (IToolBarFrameWindow*) — IToolBar 854
floatingPosition (IPoint) — IToolBar 854
floatingTitle (IString) — IToolBar 855
flowed (Boolean) — IContainerControl 855
flowedNameView (Boolean) — IContainerControl 855
flowedTextView (Boolean) — IContainerControl 855
flyOverHelp (Boolean) — IVBFlyText 855
flyOverHelpHandler (IFlyOverHelpHandler*) — IVBFlyText 855
flyTextControl (IFlyText*) — IFlyOverHelpHandler 855
flyTextStringTableOffset (long) — IFlyOverHelpHandler 856
focus (Boolean) — IWindow 856
font (IFont) — IVBFontDialog 856
font (IFont) — IWindow 856
fontDialog (IFontDialog*) — IVBFontDialog 856
fontDialogSettings (IFontDialog::Settings*) — IVBFontDialog 856
foregroundColor (IColor) — IWindow 857
format (IMMAudioBuffer::Format) — IMMConfigurableAudio — vbmm . 857
format (IMMSpeed::Format) — IMMSpeed — vbmm 857
formatCount (unsigned long) — IClipboard 857
framed (Boolean) — IMenuItem 857
frames (unsigned long) — IMMHourMinSecFrameTime — vbmm 857
frames (unsigned long) — IMMMinSecFrameTime — vbmm 857
frames (unsigned long) — IMMTrackMinSecFrameTime — vbmm 858
framesPerSecond (unsigned long) — IMMHourMinSecFrameTime — vbmm 858
frameWindow (Boolean) — IWindow 858
full (IBoolean) — IRBag — vbcc 858
full (IBoolean) — IRDeque — vbcc 858
full (IBoolean) — IREqualitySequence — vbcc 858
full (IBoolean) — IRHeap — vbcc 858

full (IBoolean) — IRQueue — vbcc . . . 858
full (IBoolean) — IRSequence . . . 859
full (IBoolean) — IRSet — vbcc . . . 859
full (IBoolean) — IRSortedBag — vbcc . . . 859
full (IBoolean) — IRSortedSet — vbcc . . . 859
full (IBoolean) — IRStack — vbcc . . . 859
gain (unsigned long) — IMMAmpMixer — vbmm . . . 859
graphicContext (IGraphicContext*) — IDrawingCanvas . . . 860
graphicList (IGList*) — IDrawingCanvas . . . 860
graphics (Boolean) — IString . . . 860
graphicWindow (IIconControl) — IGraphicPushButton . . . 860
greenMix (unsigned char) — IColor . . . 860
gridLines (Boolean) — IMultiCellCanvas . . . 860
group (Boolean) — IControl . . . 860
group (Boolean) — IWindow . . . 860
groupPad (unsigned long) — ISetCanvas . . . 861
halftone (Boolean) — I3StateCheckBox . . . 861
halftone (Boolean) — IStaticText . . . 861
handle (IAccelTblHandle) — IAccelerator . . . 861
handle (IWindowHandle) — IMMDigitalVideo — vbmm . . . 861
handle (IModuleHandle) — IResourceLibrary . . . 861
handle (IWindowHandle) — IWindow . . . 861
hasBitmap (Boolean) — IClipboard . . . 861
hasSelectedText (Boolean) — IEntryField . . . 862
hasSelectedText (Boolean) — IMultiLineEdit . . . 862
hasText (Boolean) — IClipboard . . . 862
hasTransparentColor (Boolean) — IToolBarButton . . . 862
headingIcon (IPointerHandle) — IContainerColumn . . . 862
headingIconHandle (Boolean) — IContainerColumn . . . 862
headingString (Boolean) — IContainerColumn . . . 862
headingText (IString) — IContainerColumn . . . 863
headingWriteable (Boolean) — IContainerColumn . . . 863
height (IRectangle::Coord) — IRectangle . . . 863
height (Coord) — ISize . . . 863
help (Boolean) — IPushButton . . . 863
helpId (unsigned long) — IContainerColumn . . . 863
helpId (unsigned long) — IContainerObject . . . 863
helpId (unsigned long) — IWindow . . . 864
hexDigits (Boolean) — IString . . . 864
highlighted (Boolean) — IButton . . . 864
highlighted (Boolean) — IMenuItem . . . 864
highLimit (double) — IVBDoublePart — vbsample . . . 864
highLimit (long) — IVBLongPart — vbsample . . . 864

highLimit (short) — IVBShortPart — vbsample 865
highLimit (unsigned long) — IVBUnsignedLongPart — vbsample 865
highLimit (unsigned short) — IVBUnsignedShortPart — vbsample 865
hiliteBackgroundColor (IColor) — IWindow 865
hiliteForegroundColor (IColor) — IWindow 866
homePhone (IString) — ICustomer — vbsample 866
homePosition (IProgressIndicator::HomePosition) — IProgressIndicator . . . 866
horizontal (Boolean) — IScrollBar 866
horizontalDataAlignment (IContainerColumn::HorizontalAlignment) — IContainerColumn 866
horizontalHeadingAlignment (IContainerColumn::HorizontalAlignment) — IContainerColumn 866
horizontalScroll (Boolean) — IBaseComboBox 866
horizontalScroll (Boolean) — IBaseListBox 867
horizontalScrollBar (IScrollBar*) — IViewPort 867
horizontalSeparator (Boolean) — IContainerColumn 867
hours (unsigned long) — IMMTime — vbmm 867
hours (unsigned) — ITime 867
hundredths (unsigned long) — IMMTime — vbmm 867
icon (IPointerHandle) — IContainerObject 867
icon (IPointerHandle) — IFrameWindow 868
icon (IPointerHandle) — IGraphicPushButton 868
icon (IPointerHandle) — IIconControl 868
iconHandle (Boolean) — IContainerColumn 868
iconSize (ISize) — IContainerControl 868
iconText (IString) — IContainerObject 868
iconView (Boolean) — IContainerControl 868
id (unsigned long) — IMenuItem 868
id (unsigned long) — IResourceId 869
id (unsigned long) — IWindow 869
inactiveColor (IColor) — IWindow 869
inactiveText (IString) — IInfoArea 869
inactiveTextBackgroundColor (IColor) — ITitle 869
inactiveTextForegroundColor (IColor) — ITitle 870
includesDBCS (Boolean) — IString 870
includesMBCS (Boolean) — IString 870
includesSBCS (Boolean) — IString 870
index (long) — IColor 870
index (unsigned long) — IMenuItem 870
indexWindow (IWindowHandle) — IHelpWindow 870
initialize () — IVBContainerControl 870
input1 (Boolean) — IVBLogicalAndPart — vbsample 871
input1 (Boolean) — IVBLogicalOrPart — vbsample 871

input10 (Boolean) — IVBLogicalAndPart — vbsample 871
input10 (Boolean) — IVBLogicalOrPart — vbsample 871
input2 (Boolean) — IVBLogicalAndPart — vbsample 871
input2 (Boolean) — IVBLogicalOrPart — vbsample 872
input3 (Boolean) — IVBLogicalAndPart — vbsample 872
input3 (Boolean) — IVBLogicalOrPart — vbsample 872
input4 (Boolean) — IVBLogicalAndPart — vbsample 872
input4 (Boolean) — IVBLogicalOrPart — vbsample 872
input5 (Boolean) — IVBLogicalAndPart — vbsample 873
input5 (Boolean) — IVBLogicalOrPart — vbsample 873
input6 (Boolean) — IVBLogicalAndPart — vbsample 873
input6 (Boolean) — IVBLogicalOrPart — vbsample 873
input7 (Boolean) — IVBLogicalAndPart — vbsample 873
input7 (Boolean) — IVBLogicalOrPart — vbsample 874
input8 (Boolean) — IVBLogicalAndPart — vbsample 874
input8 (Boolean) — IVBLogicalOrPart — vbsample 874
input9 (Boolean) — IVBLogicalAndPart — vbsample 874
input9 (Boolean) — IVBLogicalOrPart — vbsample 874
insertMode (Boolean) — IEntryField 875
internalLeading (unsigned long) — IFont 875
inUse (Boolean) — IContainerObject 875
isASCII (Boolean) — IString 875
isBitmap (Boolean) — IMenuItem 875
isDBCS (Boolean) — IString 875
isDigits (Boolean) — ITextControl 875
isDigits (Boolean) — ITextSpinButton 876
isLowerCase (Boolean) — IString 876
isMBCS (Boolean) — IString 876
isOpen (Boolean) — IClipboard 876
isOpen (Boolean) — IMMDevice — vbmm 876
isSBCS (Boolean) — IString 876
isSet (Boolean) — IAccelerator 876
isText (Boolean) — IMenuItem 876
isUpperCase (Boolean) — IString 876
italic (Boolean) — IFont 877
itemProvider (IDMItemProvider*) — IWindow 877
items (Collection*) — ICollectionViewComboBox 877
items (Collection*) — ICollectionViewListBox 877
items (Collection*) — IVBContainerControl 877
julianDate (unsigned long) — IDate 877
keyCount (unsigned long) — IAcceleratorTable 877
keyModifier (IKey::KeyModifier) — IAcceleratorKey 878
lastElement (Element const) — IRDeque — vbcc 878

lastElement (Element const) — IREqualitySequence — vbcc 878
lastElement (Element const) — IRQueue — vbcc 878
lastElement (Element const) — IRSequence 878
lastElement (Element const) — IRSortedBag — vbcc 878
lastElement (Element const) — IRSortedSet — vbcc 878
lastElement (Element const) — IRStack — vbcc 878
lastPage (IPageHandle) — INotebook 878
latched (Boolean) — ICustomButton 879
latchedBackgroundColor (IColor) — ICustomButton 879
latchedBackgroundColorHalftone (Boolean) — ICustomButton 879
latchedBitmap (IBitmapHandle) — IToolBarButton 879
latchedForegroundColor (IColor) — ICustomButton 879
latchingEnabled (Boolean) — ICustomButton 880
layoutAdjustment (IRectangle) — IWindow 880
layoutType (IMenuItem::LayoutType) — IMenuItem 880
left (IRectangle::Coord) — IRectangle 880
leftCenter (IPoint) — IRectangle 880
leftIndex (unsigned long) — IEntryField 880
length (unsigned long) — IMMPlayableDevice — vbmm 880
length (unsigned) — IString 881
limit (unsigned long) — IBaseSpinButton 881
limit (unsigned long) — IEntryField 881
limit (unsigned long) — IMultiLineEdit 881
limit (unsigned long) — IStaticText 881
lineCount (unsigned long) — IInfoArea 881
lineSpacing (long) — IContainerControl 882
listShowing (Boolean) — IBaseComboBox 882
location (IToolBar::Location) — IToolBar 882
longStringTableOffset (long) — IFlyOverHelpHandler 882
longTextControl (ITextControl*) — IFlyOverHelpHandler 882
longTextControl (ITextControl*) — IVBFlyText 882
lowerBound (Coord) — IRange 883
lowerCase (Boolean) — IVBStringPart — vbsample 883
lowLimit (double) — IVBDoublePart — vbsample 883
lowLimit (long) — IVBLongPart — vbsample 883
lowLimit (short) — IVBShortPart — vbsample 883
lowLimit (unsigned long) — IVBUnsignedLongPart — vbsample 884
lowLimit (unsigned short) — IVBUnsignedShortPart — vbsample 884
majorTabBackgroundColor (IColor) — INotebook 884
majorTabForegroundColor (IColor) — INotebook 884
margin (Boolean) — IEntryField 884
margin (ISize) — ISetCanvas 885
marginSize (ISize) — IGraphicPushButton 885

master (Boolean) — IBaseSpinButton 885
maxAscender (unsigned long) — IFont 885
maxCharHeight (unsigned long) — IFont 885
maxDescender (unsigned long) — IFont 885
maximized (Boolean) — IFrameWindow 885
maximizeRect (IRectangle) — IFrameWindow 885
maximumSpeed (IMMSpeed) — IMMDigitalVideo — vbmm 886
maximumWindows (unsigned long) — IMMDigitalVideo — vbmm 886
maxNumberOfElements (INumber) — IRBag — vbcc 886
maxNumberOfElements (INumber) — IRDeque — vbcc 886
maxNumberOfElements (INumber) — IREqualitySequence — vbcc 886
maxNumberOfElements (INumber) — IRHeap — vbcc 886
maxNumberOfElements (INumber) — IRQueue — vbcc 886
maxNumberOfElements (INumber) — IRSequence 887
maxNumberOfElements (INumber) — IRSet — vbcc 887
maxNumberOfElements (INumber) — IRSortedBag — vbcc 887
maxNumberOfElements (INumber) — IRSortedSet — vbcc 887
maxNumberOfElements (INumber) — IRStack — vbcc 887
maxSize (ISize) — IFont 887
maxUppercaseSize (ISize) — IFont 887
maxX (IRectangle::Coord) — IRectangle 887
maxXCenterY (IPoint) — IRectangle 888
maxXMaxY (IPoint) — IRectangle 888
maxXMinY (IPoint) — IRectangle 888
maxY (IRectangle::Coord) — IRectangle 888
mediaPresent (Boolean) — IMMRemovableMedia — vbmm 888
menu (IMenuBar) — IFrameWindow 888
menu (IMenu) — IMenuCascade 888
menu (IPopupMenu) — IWindow 888
minimized (Boolean) — IFrameWindow 888
minimizeRect (IRectangle) — IFrameWindow 889
minimumCharacters (unsigned long) — IBaseListBox 889
minimumRows (unsigned long) — IBaseComboBox 889
minimumRows (unsigned long) — IBaseListBox 889
minimumSize (ISize) — IWindow 889
minimumSpeed (IMMSpeed) — IMMDigitalVideo — vbmm 889
minorTabBackgroundColor (IColor) — INotebook 890
minorTabForegroundColor (IColor) — INotebook 890
minScrollIncrement (unsigned long) — IScrollBar 890
minutes (unsigned long) — IMMTime — vbmm 890
minutes (unsigned) — ITime 890
minX (IRectangle::Coord) — IRectangle 890
minXCenterY (IPoint) — IRectangle 890

minXMaxY (IPoint) — IRectangle 891
minXMinY (IPoint) — IRectangle 891
minY (IRectangle::Coord) — IRectangle 891
misfitFilteringEnabled (Boolean) — IToolBar 891
missingText (IString) — IInfoArea 891
mixedTargetEmphasis (Boolean) — IContainerControl 891
mleHandler (IHandler) — ICnrEditHandler 891
modal (Boolean) — IFrameWindow 892
mode (IMMDevice::Mode) — IMMDevice — vbmm 892
modeless (Boolean) — IVBFileDialog 892
modeless (Boolean) — IVBFontDialog 892
monitoringEnabled (Boolean) — IMMAmpMixer — vbmm 892
monthOfYear (IDate::Month) — IDate 892
mousePointer (IPointerHandle) — IFrameWindow 892
multipleSelect (Boolean) — IBaseListBox 893
multipleSelection (Boolean) — IContainerControl 893
name (IString) — ICompany — vbsample 893
name (IString) — ICustomer — vbsample 893
name (IString) — IFont 893
name (IString) — IProfile 893
nameView (Boolean) — IContainerControl 893
nativeRect (IRectangle) — IWindow 893
nextShellRect (IRectangle) — IFrameWindow 894
noAdjustPosition (Boolean) — IBaseListBox 894
noDismiss (Boolean) — IMenuItem 894
nonPropOnly (Boolean) — IFont 894
normalSpeed (IMMSpeed) — IMMDigitalVideo — vbmm 894
normalTargetEmphasis (Boolean) — IContainerControl 894
notInput1 (Boolean) — IVBLogicalAndPart — vbsample 895
notInput1 (Boolean) — IVBLogicalOrPart — vbsample 895
notInput10 (Boolean) — IVBLogicalAndPart — vbsample 895
notInput10 (Boolean) — IVBLogicalOrPart — vbsample 895
notInput2 (Boolean) — IVBLogicalAndPart — vbsample 895
notInput2 (Boolean) — IVBLogicalOrPart — vbsample 895
notInput3 (Boolean) — IVBLogicalAndPart — vbsample 895
notInput3 (Boolean) — IVBLogicalOrPart — vbsample 896
notInput4 (Boolean) — IVBLogicalAndPart — vbsample 896
notInput4 (Boolean) — IVBLogicalOrPart — vbsample 896
notInput5 (Boolean) — IVBLogicalAndPart — vbsample 896
notInput5 (Boolean) — IVBLogicalOrPart — vbsample 896
notInput6 (Boolean) — IVBLogicalAndPart — vbsample 896
notInput6 (Boolean) — IVBLogicalOrPart — vbsample 896
notInput7 (Boolean) — IVBLogicalAndPart — vbsample 897

notInput7 (Boolean) — IVBLogicalOrPart — vbsample 897
notInput8 (Boolean) — IVBLogicalAndPart — vbsample 897
notInput8 (Boolean) — IVBLogicalOrPart — vbsample 897
notInput9 (Boolean) — IVBLogicalAndPart — vbsample 897
notInput9 (Boolean) — IVBLogicalOrPart — vbsample 897
notValue (Boolean) — IVBBooleanPart — vbsample 898
notValue (Boolean) — IVBLogicalAndPart — vbsample 898
notValue (Boolean) — IVBLogicalOrPart — vbsample 898
notValueAsText (IString) — IVBBooleanPart — vbsample 898
now (ITime) — ITime 898
number (Boolean) — IContainerColumn 898
numberOfApplications (unsigned long) — IProfile 898
numberOfButton2Items (unsigned long) — IVBContainerControl 899
numberOfColumnChanges (unsigned long) — IContainerControl 899
numberOfDifferentElements (INumber) — IRBag — vbcc 899
numberOfDifferentElements (INumber) — IRSortedBag — vbcc 899
numberOfElements (INumber) — IRBag — vbcc 899
numberOfElements (INumber) — IRDeque — vbcc 899
numberOfElements (INumber) — IREqualitySequence — vbcc 899
numberOfElements (INumber) — IRHeap — vbcc 899
numberOfElements (INumber) — IRQueue — vbcc 899
numberOfElements (INumber) — IRSequence 900
numberOfElements (INumber) — IRSet — vbcc 900
numberOfElements (INumber) — IRSortedBag — vbcc 900
numberOfElements (INumber) — IRSortedSet — vbcc 900
numberOfElements (INumber) — IRStack — vbcc 900
numberOfEntries (unsigned long) — IMMAudioCDContents — vbmm . . . 900
numberOfKeys (unsigned long) — IProfile 900
numberOfLines (unsigned long) — IMultiLineEdit 900
numberOfObjectChanges (unsigned long) — IContainerControl 900
numberOfSelections (unsigned long) — IBaseComboBox 901
numberOfSelections (unsigned long) — IBaseListBox 901
numberOfSelections (unsigned long) — IVBContainerControl 901
numberOfTracks (unsigned long) — IMMAudioCD — vbmm 901
numberOfTracks (unsigned long) — IMMAudioCDContents — vbmm . . . 901
numWords (unsigned) — IString 901
objectCopy (IContainerObject*) — IContainerObject 901
objectCount (unsigned long) — IContainerControl 901
objectList (ICnrObjectSet) — IContainerControl 902
objectText (IString) — ITitle 902
open (Boolean) — IContainerObject 902
open (Boolean) — IResourceLibrary 902
ordered (IBoolean) — IRecord — vbsample 902

orderedTargetEmphasis (Boolean) — IContainerControl 902
ordinal (unsigned long) — IMMTime — vbmm 902
orientation (INotebook::Orientation) — INotebook 903
orientation (ISplitCanvas::Orientation) — ISplitCanvas 903
origDefaultButtonHandle (IWindowHandle) — ICanvas 903
outline (Boolean) — IFont 903
outlineType (IOutlineBox::OutlineType) — IOutlineBox 903
owner (IWindow*) — IAccelerator 903
owner (IWindowHandle) — IClipboard 903
owner (IWindow*) — IWindow 904
packType (ISetCanvas::PackType) — ISetCanvas 904
pad (ISize) — ISetCanvas 904
pageBackgroundColor (IColor) — INotebook 904
pageScrollIncrement (unsigned long) — IScrollBar 904
pageSize (ISize) — INotebook 904
parent (IWindow*) — IWindow 905
parentSize (ISize) — IWindow 905
pauseButton (IAnimatedButton*) — IMMPlayerPanel — vbmm 905
phone (IString) — ICompany — vbsample 905
pitch (unsigned long) — IMMAmpMixer — vbmm 905
playableDevice (IMMPlayableDevice*) — IMMPlayerPanel — vbmm . . . 905
playButton (IAnimatedButton*) — IMMPlayerPanel — vbmm 905
playingForward (Boolean) — IMMDigitalVideo — vbmm 906
pointerCaptured (Boolean) — IWindow 906
pointSize (unsigned long) — IFont 906
pointSize (unsigned long) — IVBFontDialog 906
position (unsigned long) — IMMPlayableDevice — vbmm 906
position (IPoint) — IWindow 906
pressedOK (Boolean) — IVBFileDialog 906
pressedOK (Boolean) — IVBFontDialog 907
presSpace (IPresSpaceHandle) — IWindow 907
primaryFormat (IString) — IClipboard 907
primaryScale (IProgressIndicator::Scale) — IProgressIndicator 907
printable (Boolean) — IString 907
profile (IProfile) — IMMAudioCD — vbmm 907
punctuation (Boolean) — IString 907
radius (unsigned long) — ICircularSlider 907
range (IRange) — INumericSpinButton 908
readOnly (Boolean) — IMMFileMedia — vbmm 908
rect (IRectangle) — IWindow 908
redMix (unsigned char) — IColor 908
refreshOn (Boolean) — IContainerControl 908
refreshOn (Boolean) — IContainerObject 908

relativeWindowRect (IRectangle) — IFlyText 909
requiresFiles (Boolean) — IMMDevice — vbmm 909
resourceLibrary (IResourceLibrary) — IFlyOverHelpHandler 909
resourceLibrary (IResourceLibrary) — IInfoArea 909
resourceLibrary (IResourceLibrary) — IResourceId 909
restoreRect (IRectangle) — IFrameWindow 909
result (unsigned long) — IFrameWindow 910
returnValue (long) — IVBFileDialog 910
returnValue (long) — IVBFontDialog 910
reversed (IPointArray) — IPointArray 910
rewindButton (IAnimatedButton*) — IMMPlayerPanel — vbmm 910
ribbonStripEnabled (Boolean) — IProgressIndicator 910
right (IRectangle::Coord) — IRectangle 910
rightCenter (IPoint) — IRectangle 910
rotationIncrement (unsigned long) — ICircularSlider 911
samplesPerSecond (unsigned long) — IMMConfigurableAudio — vbmm . 911
scrollableRange (IRange) — IScrollBar 911
scrollBoxPosition (unsigned long) — IScrollBar 911
scrollBoxRange (IRange) — IScrollBar 911
searchListWindow (IWindowHandle) — IHelpWindow 911
seconds (unsigned long) — IMMTime — vbmm 911
seconds (unsigned) — ITime 912
selectable (Boolean) — IMenuItem 912
selected (Boolean) — ISettingButton 912
selectedCnrElement (CnrElement*) — IVBContainerControl 912
selectedCnrItem (CnrElement*) — IVBContainerControl 912
selectedCnrItems (IVSequence<CnrElement*>*) — IVBContainerControl . 912
selectedCollectionPosition (unsigned long) — ICollectionViewComboBox . 913
selectedCollectionPosition (unsigned long) — ICollectionViewListBox . . . 913
selectedCollectionPosition (unsigned long) — IVBContainerControl 913
selectedCollectionPositions (IVSequence<unsigned long>*) — IVBContainerControl 913
selectedElement (Element) — ICollectionViewComboBox 914
selectedElement (Element) — ICollectionViewListBox 914
selectedElement (Element) — IVBContainerControl 914
selectedFileCount (unsigned long) — IVBFileDialog 914
selectedIndex (unsigned long) — IRadioButton 914
selectedItem (Element) — IVBContainerControl 914
selectedItems (IVSequence<Element>*) — IVBContainerControl 915
selectedRange (IRange) — IEntryField 915
selectedRange (IRange) — IMultiLineEdit 915
selectedText (IString) — IEntryField 915
selectedText (IString) — IMultiLineEdit 915

selectedTextLength (unsigned long) — IEntryField 915
selectedTextLength (unsigned long) — IMultiLineEdit 915
selection (long) — IBaseComboBox 916
selection (long) — IBaseListBox 916
selection (long) — ICollectionViewComboBox 916
selection (long) — ICollectionViewListBox 916
selectionText (IString) — IBaseListBox 916
separator (Boolean) — IMenuItem 916
servant (Boolean) — IBaseSpinButton 916
shadowColor (IColor) — IWindow 917
shaftPosition (IPoint) — IProgressIndicator 917
shaftSize (ISize) — IProgressIndicator 917
showing (Boolean) — IWindow 917
showingMiniIcons (Boolean) — IContainerControl 917
singleSelection (Boolean) — IContainerControl 917
size (unsigned long) — IPointArray 917
size (unsigned long) — IRecord — vbsample 917
size (ISize) — IRectangle 918
size (unsigned) — IString 918
size (ISize) — IWindow 918
sizeToGraphic (Boolean) — IGraphicPushButton 918
snapToTickEnabled (Boolean) — IProgressIndicator 918
sourceRectangle (IRectangle) — IMMDigitalVideo — vbmm 918
speed (IMMSpeed) — IMMDigitalVideo — vbmm 918
speed (unsigned long) — IMMSpeed — vbmm 919
speedFormat (IMMSpeed::Format) — IMMDevice — vbmm 919
spinFieldValid (Boolean) — INumericSpinButton 919
spinFieldValid (Boolean) — ITextSpinButton 919
splitBarEdgeColor (IColor) — ISplitCanvas 919
splitBarMiddleColor (IColor) — ISplitCanvas 919
splitBarOffset (unsigned long) — IContainerControl 919
standardBitmapSize (ISize) — IToolBarButton 920
standardFormat (Boolean) — IToolBarButton 920
standardTextLines (unsigned long) — IToolBarButton 920
standardTextWidth (unsigned long) — IToolBarButton 920
state (IString) — IAddress — vbsample 920
statusTextAlignment (INotebook::TextAlignment) — INotebook 920
stepBackwardButton (IAnimatedButton*) — IMMPlayerPanel — vbmm . . 921
stepForwardButton (IAnimatedButton*) — IMMPlayerPanel — vbmm . . . 921
stopButton (IAnimatedButton*) — IMMPlayerPanel — vbmm 921
street (IString) — IAddress — vbsample 921
strikeout (Boolean) — IFont 921
strikeout (Boolean) — IStaticText 921

string (Boolean) — IContainerColumn 921
stringGenerator (IStringGenerator<Element>) — ICollectionViewComboBox 922
stringGenerator (IStringGenerator<Element>) — ICollectionViewListBox . 922
stringTableOffset (long) — IInfoArea 922
submenu (Boolean) — IMenuItem 922
submenuHandle (IWindowHandle) — IMenuItem 922
supportsAudio (Boolean) — IMMDevice — vbmm 922
supportsDigitalTransfer (Boolean) — IMMDevice — vbmm 922
supportsDisableEject (Boolean) — IMMDevice — vbmm 923
supportsEject (Boolean) — IMMDevice — vbmm 923
supportsOverlayGraphics (Boolean) — IMMDigitalVideo — vbmm 923
supportsPlay (Boolean) — IMMDevice — vbmm 923
supportsRecord (Boolean) — IMMDevice — vbmm 923
supportsRecordInsertion (Boolean) — IMMDevice — vbmm 923
supportsReverse (Boolean) — IMMDigitalVideo — vbmm 923
supportsSave (Boolean) — IMMDevice — vbmm 923
supportsSizing (Boolean) — IMMDigitalVideo — vbmm 924
supportsStreaming (Boolean) — IMMDevice — vbmm 924
supportsStretchToFit (Boolean) — IMMDigitalVideo — vbmm 924
supportsVideo (Boolean) — IMMDevice — vbmm 924
supportsVolumeAdjustment (Boolean) — IMMDevice — vbmm 924
systemCommand (ICommand::CommandId) — IAcceleratorKey 924
systemCommand (Boolean) — IPushButton 925
systemProfile (IProfile::IProfile) — IProfile 925
systemScrollBarWidth (unsigned long) — IScrollBar 925
systemScrollBoxLength (unsigned long) — IScrollBar 925
systemScrollButtonLength (unsigned long) — IScrollBar 925
tabShape (INotebook::TabShape) — INotebook 925
tabStop (Boolean) — IControl 926
tabStop (Boolean) — IWindow 926
tabTextAlignment (INotebook::TextAlignment) — INotebook 926
text (const char*) — ICLibErrorInfo 926
text (IString) — IClipboard 926
text (const char*) — IGUIErrorInfo 926
text (IVBMnemonicString) — IMenuItem 926
text (const char*) — IMMErrorInfo — vbmm 926
text (IVBMnemonicString) — ISetCanvas 927
text (const char*) — ISystemErrorInfo 927
text (IVBMnemonicString) — ITextControl 927
text (IString) — ITextSpinButton 927
text (IString) — IVBStringPart — vbsample 927
textAsLowerCase (IString) — IVBStringPart — vbsample 927
textAsUpperCase (IString) — IVBStringPart — vbsample 928

textEqualDefault (Boolean) — IVBStringPart — vbsample 928
textLength (unsigned long) — ITextControl 928
textLength (unsigned) — ITextSpinButton 928
textLength (unsigned long) — IVBStringPart — vbsample 928
textNotEqualDefault (Boolean) — IVBStringPart — vbsample 928
textView (Boolean) — IContainerControl 928
textVisible (Boolean) — IToolBarButton 928
thousandths (unsigned long) — IMMTime — vbmm 929
tickSpacing (unsigned long) — ICircularSlider 929
time (Boolean) — IContainerColumn 929
time (ITime) — ICustomer — vbsample 929
timeFormat (IMMTime::Format) — IMMDevice — vbmm 929
title (IString) — IContainerControl 930
titleRectangle (IRectangle) — IContainerControl 930
titleSeparatorVisible (Boolean) — IContainerControl 930
titleVisible (Boolean) — IContainerControl 930
titleWriteable (Boolean) — IContainerControl 930
today (IDate) — IDate 930
toolBarContainer (IToolBarContainer*) — IToolBar 930
toolBarList (IToolBarList*) — IFrameWindow 931
top (unsigned long) — IBaseComboBox 931
top (unsigned long) — IBaseListBox 931
top (unsigned long) — IMultiLineEdit 931
top (IRectangle::Coord) — IRectangle 931
top (Element const) — IRStack — vbcc 931
topCenter (IPoint) — IRectangle 931
topLeft (IPoint) — IRectangle 932
topPage (IPageHandle) — INotebook 932
topRight (IPoint) — IRectangle 932
totalPages (unsigned long) — INotebook 932
traceDestination (ITrace::Destination) — ITrace 932
traceEnabled (Boolean) — ITrace 932
track (unsigned long) — IMMTrackMinSecFrameTime — vbmm 932
transparentColor (IColor) — IToolBarButton 933
treble (unsigned long) — IMMAmpMixer — vbmm 933
treeIconView (Boolean) — IContainerControl 933
treeNameView (Boolean) — IContainerControl 933
treeTextView (Boolean) — IContainerControl 933
treeView (Boolean) — IContainerControl 933
type (IBaseComboBox::ControlType) — IBaseComboBox 933
underscore (Boolean) — IFont 934
underscore (Boolean) — IStaticText 934
undoable (Boolean) — IMultiLineEdit 934

upc (IString) — IMMAudioCD — vbmm 934
upperBound (Coord) — IRange 934
upperCase (Boolean) — IVBStringPart — vbsample 934
userData (unsigned long) — ICustomButton 935
userProfile (IProfile::IProfile) — IProfile 935
usesDialogBackground (Boolean) — IFrameWindow 935
valid (Boolean) — IMMAudioCDContents — vbmm 935
valid (Boolean) — IMMTime — vbmm 935
valid (Boolean) — IWindow 935
validDBCS (Boolean) — IString 935
validMBCS (Boolean) — IString 935
value (long) — ICircularSlider 936
value (IColor::Color) — IColor 936
value (long) — INumericSpinButton 936
value (Boolean) — IVBBooleanPart — vbsample 936
value (double) — IVBDoublePart — vbsample 936
value (Boolean) — IVBLogicalAndPart — vbsample 936
value (Boolean) — IVBLogicalOrPart — vbsample 937
value (long) — IVBLongPart — vbsample 937
value (short) — IVBShortPart — vbsample 937
value (unsigned long) — IVBUnsignedLongPart — vbsample 937
value (unsigned short) — IVBUnsignedShortPart — vbsample 937
valueAboveHighLimit (Boolean) — IVBDoublePart — vbsample 937
valueAboveHighLimit (Boolean) — IVBLongPart — vbsample 938
valueAboveHighLimit (Boolean) — IVBShortPart — vbsample 938
valueAboveHighLimit (Boolean) — IVBUnsignedLongPart — vbsample . . 938
valueAboveHighLimit (Boolean) — IVBUnsignedShortPart — vbsample . . 938
valueAs1Based (double) — IVBDoublePart — vbsample 938
valueAs1Based (long) — IVBLongPart — vbsample 939
valueAs1Based (short) — IVBShortPart — vbsample 939
valueAs1Based (unsigned long) — IVBUnsignedLongPart — vbsample . . 939
valueAs1Based (unsigned short) — IVBUnsignedShortPart — vbsample . . 939
valueAsDouble (double) — ITextControl 939
valueAsDouble (double) — ITextSpinButton 940
valueAsInt (int) — ITextControl 940
valueAsInt (int) — ITextSpinButton 940
valueAsText (IString) — IVBBooleanPart — vbsample 940
valueAsText (IString) — IVBDoublePart — vbsample 940
valueAsText (IString) — IVBLogicalAndPart — vbsample 941
valueAsText (IString) — IVBLogicalOrPart — vbsample 941
valueAsText (IString) — IVBLongPart — vbsample 941
valueAsText (IString) — IVBShortPart — vbsample 941
valueAsText (IString) — IVBUnsignedLongPart — vbsample 941

valueAsText (IString) — IVBUnsignedShortPart — vbsample 942
valueAsUnsigned (unsigned int) — ITextControl 942
valueAsUnsigned (unsigned int) — ITextSpinButton 942
valueBelowLowLimit (Boolean) — IVBDoublePart — vbsample 942
valueBelowLowLimit (Boolean) — IVBLongPart — vbsample 942
valueBelowLowLimit (Boolean) — IVBShortPart — vbsample 942
valueBelowLowLimit (Boolean) — IVBUnsignedLongPart — vbsample . . 943
valueBelowLowLimit (Boolean) — IVBUnsignedShortPart — vbsample . . 943
valueEqualDefault (Boolean) — IVBBooleanPart — vbsample 943
valueEqualDefault (Boolean) — IVBDoublePart — vbsample 943
valueEqualDefault (Boolean) — IVBLongPart — vbsample 943
valueEqualDefault (Boolean) — IVBShortPart — vbsample 943
valueEqualDefault (Boolean) — IVBUnsignedLongPart — vbsample 943
valueEqualDefault (Boolean) — IVBUnsignedShortPart — vbsample 944
valueEqualHighLimit (Boolean) — IVBDoublePart — vbsample 944
valueEqualHighLimit (Boolean) — IVBLongPart — vbsample 944
valueEqualHighLimit (Boolean) — IVBShortPart — vbsample 944
valueEqualHighLimit (Boolean) — IVBUnsignedLongPart — vbsample . . 944
valueEqualHighLimit (Boolean) — IVBUnsignedShortPart — vbsample . . 944
valueEqualLowLimit (Boolean) — IVBDoublePart — vbsample 944
valueEqualLowLimit (Boolean) — IVBLongPart — vbsample 945
valueEqualLowLimit (Boolean) — IVBShortPart — vbsample 945
valueEqualLowLimit (Boolean) — IVBUnsignedLongPart — vbsample . . 945
valueEqualLowLimit (Boolean) — IVBUnsignedShortPart — vbsample . . 945
valueNegative (Boolean) — IVBDoublePart — vbsample 945
valueNegative (Boolean) — IVBLongPart — vbsample 945
valueNegative (Boolean) — IVBShortPart — vbsample 945
valueNotEqualDefault (Boolean) — IVBBooleanPart — vbsample 946
valueNotEqualDefault (Boolean) — IVBDoublePart — vbsample 946
valueNotEqualDefault (Boolean) — IVBLongPart — vbsample 946
valueNotEqualDefault (Boolean) — IVBShortPart — vbsample 946
valueNotEqualDefault (Boolean) — IVBUnsignedLongPart — vbsample . . 946
valueNotEqualDefault (Boolean) — IVBUnsignedShortPart — vbsample . . 946
valueNotZero (Boolean) — IVBDoublePart — vbsample 946
valueNotZero (Boolean) — IVBLongPart — vbsample 947
valueNotZero (Boolean) — IVBShortPart — vbsample 947
valueNotZero (Boolean) — IVBUnsignedLongPart — vbsample 947
valueNotZero (Boolean) — IVBUnsignedShortPart — vbsample 947
valueOutsideLimits (Boolean) — IVBDoublePart — vbsample 947
valueOutsideLimits (Boolean) — IVBLongPart — vbsample 947
valueOutsideLimits (Boolean) — IVBShortPart — vbsample 947
valueOutsideLimits (Boolean) — IVBUnsignedLongPart — vbsample . . . 948
valueOutsideLimits (Boolean) — IVBUnsignedShortPart — vbsample . . . 948

valuePositive (Boolean) — IVBDoublePart — vbsample 948
valuePositive (Boolean) — IVBLongPart — vbsample 948
valuePositive (Boolean) — IVBShortPart — vbsample 948
valueWithinLimits (Boolean) — IVBDoublePart — vbsample 948
valueWithinLimits (Boolean) — IVBLongPart — vbsample 948
valueWithinLimits (Boolean) — IVBShortPart — vbsample 949
valueWithinLimits (Boolean) — IVBUnsignedLongPart — vbsample 949
valueWithinLimits (Boolean) — IVBUnsignedShortPart — vbsample 949
valueZero (Boolean) — IVBDoublePart — vbsample 949
valueZero (Boolean) — IVBLongPart — vbsample 949
valueZero (Boolean) — IVBShortPart — vbsample 949
valueZero (Boolean) — IVBUnsignedLongPart — vbsample 949
valueZero (Boolean) — IVBUnsignedShortPart — vbsample 950
vectorOnly (Boolean) — IFont 950
vertical (Boolean) — IProgressIndicator 950
vertical (Boolean) — IScrollBar 950
verticalDataAlignment (IContainerColumn::VerticalAlignment) —
IContainerColumn 950
verticalHeadingAlignment (IContainerColumn::VerticalAlignment) —
IContainerColumn 950
verticalScrollBar (IScrollBar*) — IViewPort 950
verticalSeparator (Boolean) — IContainerColumn 950
videoFileHeight (unsigned long) — IMMDigitalVideo — vbmm 951
videoFileName (IString) — IMMDigitalVideo — vbmm 951
videoFileWidth (unsigned long) — IMMDigitalVideo — vbmm 951
videoHeight (unsigned long) — IMMDigitalVideo — vbmm 951
videoWidth (unsigned long) — IMMDigitalVideo — vbmm 951
view (IToolBarButton::View) — IToolBarButton 951
viewedPagesWindow (IWindowHandle) — IHelpWindow 951
viewNumber (unsigned long) — ITitle 952
viewPortOnWindow (IRectangle) — IContainerControl 952
viewPortOnWorkspace (IRectangle) — IContainerControl 952
viewText (IString) — ITitle 952
viewWindow (IWindowHandle) — IViewPort 952
viewWindowDrawRectangle (IRectangle) — IViewPort 952
viewWindowSize (ISize) — IViewPort 953
virtualKey (IKey::VirtualKey) — IAcceleratorKey 953
visible (Boolean) — IContainerColumn 953
visible (Boolean) — IContainerObject 953
visible (Boolean) — IHelpWindow 953
visible (Boolean) — IWindow 954
visibleCount (unsigned long) — IScrollBar 954
visibleLines (unsigned long) — IMultiLineEdit 954

visibleRectangle (IRectangle) — IWindow . . . 954
volume (unsigned long) — IMMDevice — vbmm . . . 954
volume (unsigned long) — IMMMasterAudio — vbmm . . . 955
whiteSpace (Boolean) — IString . . . 955
width (IRectangle::Coord) — IRectangle . . . 955
width (Coord) — ISize . . . 955
willDeleteColumnsOnClose (Boolean) — IContainerControl . . . 955
willDeleteObjectsOnClose (Boolean) — IContainerControl . . . 955
willDestroyOnClose (Boolean) — IFrameWindow . . . 955
wordWrap (Boolean) — IMultiLineEdit . . . 956
workPhone (IString) — ICustomer — vbsample . . . 956
writeable (Boolean) — IBaseSpinButton . . . 956
writeable (Boolean) — IContainerColumn . . . 956
writeable (Boolean) — IContainerObject . . . 956
writeable (Boolean) — IEntryField . . . 957
writeable (Boolean) — IMultiLineEdit . . . 957
writeLineNumberEnabled (Boolean) — ITrace . . . 957
writePrefixEnabled (Boolean) — ITrace . . . 957
x (Coord) — IPoint . . . 957
y (Coord) — IPoint . . . 957
year (int) — IDate . . . 957
zip (IString) — IAddress — vbsample . . . 958

Chapter 166. Events . . . 959
activateEvent — IFrameWindow . . . 959
addedEvent — IRBag — vbcc . . . 959
addedEvent — IRDeque — vbcc . . . 959
addedEvent — IREqualitySequence — vbcc . . . 959
addedEvent — IRHeap — vbcc . . . 959
addedEvent — IRQueue — vbcc . . . 959
addedEvent — IRSequence . . . 959
addedEvent — IRSet — vbcc . . . 959
addedEvent — IRSortedBag — vbcc . . . 960
addedEvent — IRSortedSet — vbcc . . . 960
addedEvent — IRStack — vbcc . . . 960
addEvent — IContainerControl . . . 960
anyEvent — INotifier . . . 960
anyEvent — IRBag — vbcc . . . 960
anyEvent — IRDeque — vbcc . . . 960
anyEvent — IREqualitySequence — vbcc . . . 960
anyEvent — IRHeap — vbcc . . . 960
anyEvent — IRQueue — vbcc . . . 960
anyEvent — IRSequence . . . 961

anyEvent — IRSet — vbcc 961
anyEvent — IRSortedBag — vbcc 961
anyEvent — IRSortedSet — vbcc 961
anyEvent — IRStack — vbcc 961
armTrackEvent — ISlider 961
buttonClickEvent — IButton 961
characterTypeEvent — IEntryField 961
closeEvent — IFrameWindow 961
commandEvent — IMenuItem 962
commandEvent — IWindow 962
commandNotifyEvent — IMMDevice — vbmm 962
createdEvent — IVBFileDialog 962
createdEvent — IVBFontDialog 962
cuePointEvent — IMMDevice — vbmm 962
customerAddedEvent — ICompany — vbsample 962
deactivateEvent — IFrameWindow 962
detailsViewTitlesEvent — IContainerControl 963
deviceEvent — IMMDevice — vbmm 963
digitsEvent — IVBDataTypePart — vbsample 963
dismissedEvent — IVBFileDialog 963
dismissedEvent — IVBFontDialog 963
enterEvent — IBaseComboBox 963
enterEvent — IBaseListBox 963
enterEvent — IContainerControl 963
extendedSelectChangedEvent — ICollectionViewListBox 964
gotFocusEvent — IWindow 964
inputDisabledEvent — IWindow 964
inputEnabledEvent — IWindow 964
inputStringIsValid — IVBDataTypePart — vbsample 964
inputStringNotValid — IVBDataTypePart — vbsample 964
itemChangedEvent — ICollectionViewComboBox 964
itemChangedEvent — ICollectionViewListBox 964
itemChangedEvent — IVBContainerControl 965
lostFocusEvent — IWindow 965
lowerCaseEvent — IVBDataTypePart — vbsample 965
mediaLoadedEvent — IMMRemovableMedia — vbmm 965
modifiedEvent — IRBag — vbcc 965
modifiedEvent — IRDeque — vbcc 965
modifiedEvent — IREqualitySequence — vbcc 965
modifiedEvent — IRHeap — vbcc 965
modifiedEvent — IRQueue — vbcc 965
modifiedEvent — IRSequence 966
modifiedEvent — IRSet — vbcc 966

modifiedEvent — IRSortedBag — vbcc 966
modifiedEvent — IRSortedSet — vbcc 966
modifiedEvent — IRStack — vbcc 966
passDeviceEvent — IMMDevice — vbmm 966
positionChangeEvent — IMMDevice — vbmm 966
positionTimerEvent — IMMAudioCD — vbmm 966
pressedOkEvent — IVBFileDialog 966
pressedOkEvent — IVBFontDialog 967
removedEvent — IRBag — vbcc 967
removedEvent — IRDeque — vbcc 967
removedEvent — IREqualitySequence — vbcc 967
removedEvent — IRHeap — vbcc 967
removedEvent — IRQueue — vbcc 967
removedEvent — IRSequence 967
removedEvent — IRSet — vbcc 967
removedEvent — IRSortedBag — vbcc 967
removedEvent — IRSortedSet — vbcc 967
removedEvent — IRStack — vbcc 968
removeEvent — IContainerControl 968
replacedEvent — IRBag — vbcc 968
replacedEvent — IRDeque — vbcc 968
replacedEvent — IREqualitySequence — vbcc 968
replacedEvent — IRHeap — vbcc 968
replacedEvent — IRQueue — vbcc 968
replacedEvent — IRSequence 968
replacedEvent — IRSet — vbcc 968
replacedEvent — IRSortedBag — vbcc 968
replacedEvent — IRSortedSet — vbcc 969
replacedEvent — IRStack — vbcc 969
scaleEvent — IProgressIndicator 969
selectEvent — IContainerControl 969
systemCommandEvent — IWindow 969
textEqualDefaultEvent — IVBDataTypePart — vbsample 969
textLengthEvent — IVBDataTypePart — vbsample 969
textNotEqualDefaultEvent — IVBDataTypePart — vbsample 969
trackStartedEvent — IMMAudioCD — vbmm 969
upperCaseEvent — IVBDataTypePart — vbsample 970
valueAboveHighLimitEvent — IVBDataTypePart — vbsample 970
valueBelowLowLimitEvent — IVBDataTypePart — vbsample 970
valueEqualDefaultEvent — IVBDataTypePart — vbsample 970
valueEqualHighLimitEvent — IVBDataTypePart — vbsample 970
valueEqualLowLimitEvent — IVBDataTypePart — vbsample 970
valueFalseEvent — IVBBooleanPart — vbsample 970

valueNegativeEvent — IVBDataTypePart — vbsample 970
valueNotEqualDefaultEvent — IVBDataTypePart — vbsample 970
valueNotZeroEvent — IVBDataTypePart — vbsample 970
valueOutsideLimitsEvent — IVBDataTypePart — vbsample 971
valuePositiveEvent — IVBDataTypePart — vbsample 971
valueTrueEvent — IVBBooleanPart — vbsample 971
valueWithinLimitsEvent — IVBDataTypePart — vbsample 971
valueZeroEvent — IVBDataTypePart — vbsample 971
visibilityDisabledEvent — IWindow 971
visibilityEnabledEvent — IWindow 971

Chapter 164. Actions

abort — stdlibSamples — vbsample

Description: abort causes an abnormal program termination and returns control to the host environment.

Member:
```
void abort ();
```

abs — stdlibSamples — vbsample

Description: abs returns the absolute value of an integer argument n.

Member:
```
int abs (int n);
```

access — ioSamples — vbsample

Description: access determines whether the specified file exists and whether you can get access to it in the given mode.

Member:
```
int access (char* pathname, int mode);
```

acos — mathSamples — vbsample

Description: acos calculates the arccosine of x, expressed in radians, in the range 0 to pi.

Member:
```
double acos (double x);
```

acquire — IMMDevice — vbmm

Description: Requests assignment of (access to) the device.

Member:
```
virtual IMMDevice& acquire
(ShareMode acquire=IMMDevice::shareable,
Boolean queuedForResources=false,
CallType call=IMMDevice::wait);
```

add — IComboBox

Description: Inserts text into the list box portion of the combination box by either.

Member:
```
virtual unsigned long add (unsigned long index,
const char* text);
```

© Copyright IBM Corp. 1995, 1996

add

add — IListBox

Description: Inserts one of the following, depending on which form of the virtual member function you use.

Member:
```
virtual unsigned long add (unsigned long index,
const char* text);
```

add — IMultiLineEdit

Description: Inserts the specified text into the MLE at the current cursor position and positions the cursor at the end of the inserted text.

Member:
```
virtual IMultiLineEdit& add (const char* text,
unsigned long textSize=0,
EOLFormat type=IMultiLineEdit::cfText);
```

add — IPointArray

Description: Adds a point to the end of the array.

Member:
```
IPointArray& add (const IPoint& point);
```

add — IRBag — vbcc

Description: Returns `True` if the element was added.

Member:
```
IBoolean add (Element const& element);
```

add — IRDeque — vbcc

Description: Returns `True` if the element was added.

Member:
```
IBoolean add (Element const& element);
```

add — IREqualitySequence — vbcc

Description: Returns `True` if the element was added.

Member:
```
IBoolean add (Element const& element);
```

add — IRHeap — vbcc

Description: Returns `True` if the element was added.

Member:
```
IBoolean add (Element const& element);
```

add — IRQueue — vbcc

Description: Returns `True` if the element was added.

Member:
```
IBoolean add (Element const& element);
```

add — IRSequence

Description: Returns `True` if the element was added.

Member:
```
IBoolean add (Element const& element);
```

add — IRSet — vbcc

Description: Returns `True` if the element was added.
Member:
```
IBoolean add (Element const& element);
```

add — IRSortedBag — vbcc

Description: Returns `True` if the element was added.
Member:
```
IBoolean add (Element const& element);
```

add — IRSortedSet — vbcc

Description: Returns `True` if the element was added.
Member:
```
IBoolean add (Element const& element);
```

add — IRStack — vbcc

Description: Returns `True` if the element was added.
Member:
```
IBoolean add (Element const& element);
```

add — ITextSpinButton

Description: Adds a new item at the cursor or index position.
Member:
```
virtual ITextSpinButton& add (const char*const* stringArray,
unsigned long index, unsigned long count);
```

addAllFrom — IACollection

Description: Adds (copies) all elements of the given collection to the collection.
Member:
```
virtual void addAllFrom
(IACollection<Element>const& collection);
```

addAllFrom — IADeque — vbcc

Description: Adds (copies) all elements of the given collection to the collection.
Member:
```
virtual void addAllFrom (IADeque<Element>const& collection);
```

addAllFrom — IAQueue — vbcc

Description: Adds (copies) all elements of the given collection to the collection.
Member:
```
virtual void addAllFrom (IAQueue<Element>const& collection);
```

addAllFrom — IAStack — vbcc

Description: Adds (copies) all elements of the given collection to the collection.
Member:
```
virtual void addAllFrom (IAStack<Element>const& collection);
```

addAscending — IComboBox

Description: Inserts the specified item in ascending sort order and returns the index of the item inserted.

Member:
```
virtual unsigned long addAscending (const char* text);
```

addAscending — IListBox

Description: Inserts the line of text in ascending sort order and returns the index of the item inserted.

Member:
```
virtual unsigned long addAscending (const char* text);
```

addAsFirst — IComboBox

Description: Inserts text as the first item in the list box portion of the combination box by either.

Member:
```
virtual unsigned long addAsFirst (const char* text);
```

addAsFirst — IListBox

Description: Inserts one of the following, depending on which form of the virtual function you use.

Member:
```
virtual unsigned long addAsFirst (const char* text);
```

addAsFirst — IRDeque — vbcc

Description: Adds the element to the collection as the first element in sequential order.

Member:
```
void addAsFirst (Element const& element);
```

addAsFirst — IREqualitySequence — vbcc

Description: Adds the element to the collection as the first element in sequential order.

Member:
```
void addAsFirst (Element const& element);
```

addAsFirst — IRSequence

Description: Adds the element to the collection as the first element in sequential order.

Member:
```
void addAsFirst (Element const& element);
```

addAsFirst — ITextSpinButton

Description: Adds the item as the first item.

Member:
```
virtual ITextSpinButton& addAsFirst (const char* string);
```

addAsFirst — IToolBar

Description: Adds the window as the first item in the tool bar.

Member:
```
virtual IToolBar& addAsFirst (IWindow* window,
Boolean startNewGroup=false);
```

addAsLast — IComboBox

Description: Inserts a line of text at the end of the list box portion of the combination box.

Member:
```
virtual unsigned long addAsLast (const char* text);
```

addAsLast — IListBox

Description: Inserts one of the following, depending on which form of the virtual function you use.

Member:
```
virtual unsigned long addAsLast (const char* text);
```

addAsLast — IMultiLineEdit

Description: Inserts the specified text into the MLE at the end of the current text, but does not change the cursor position.

Member:
```
virtual IMultiLineEdit& addAsLast (const char* text,
unsigned long textSize=0,
EOLFormat type=IMultiLineEdit::cfText);
```

addAsLast — IRDeque — vbcc

Description: Adds the element to the collection as the last element in sequential order.

Member:
```
void addAsLast (Element const& element);
```

addAsLast — IREqualitySequence — vbcc

Description: Adds the element to the collection as the last element in sequential order.

Member:
```
void addAsLast (Element const& element);
```

addAsLast — IRQueue — vbcc

Description: Adds the element to the collection as the last element in sequential order.

Member:
```
void addAsLast (Element const& element);
```

addAsLast — IRSequence

Description: Adds the element to the collection as the last element in sequential order.

Member:
```
void addAsLast (Element const& element);
```

addAsLast — IRStack — vbcc

Description: Adds the element to the collection as the last element in sequential order.

Member:
```
void addAsLast (Element const& element);
```

addAsLast — ITextSpinButton

Description: Adds the item as the last item.

Member:
```
virtual ITextSpinButton& addAsLast (const char* string);
```

addAsLast — IToolBar

Description: Adds the window as the last item in the tool bar.

Member:
```
virtual IToolBar& addAsLast (IWindow* window,
Boolean startNewGroup=false);
```

addAsNext — IToolBar

Description: Adds the window as the next item in the tool bar immediately after the reference window.

Member:
```
virtual IToolBar& addAsNext (IWindow* window,
IWindow* referenceWindow, Boolean startNewGroup=false);
```

addAsPrevious — IToolBar

Description: Adds the window as the previous item in the tool bar immediately before the reference window.

Member:
```
virtual IToolBar& addAsPrevious (IWindow* window,
IWindow* referenceWindow, Boolean startNewGroup=false);
```

addAtOffset — IMultiLineEdit

Description: Inserts the specified text into the MLE at the point specified.

Member:
```
virtual IMultiLineEdit& addAtOffset (const char* text,
unsigned long charnumber, unsigned long textSize=0,
EOLFormat type=IMultiLineEdit::cfText);
```

addAtPosition — IREqualitySequence — vbcc

Description: Adds the element at the given position to the collection.

Member:
```
void addAtPosition (IPosition position,
Element const& element);
```

addAtPosition — IRSequence

Description: Adds the element at the given position to the collection.

Member:
```
void addAtPosition (IPosition position,
Element const& element);
```

addBorder — IBaseSpinButton

Description: Adds a border to the spin button.

Member:
```
virtual IBaseSpinButton& addBorder (Boolean add=true);
```

addColumn — IContainerControl

Description: Adds a column to the container.

Member:
```
virtual IContainerControl& addColumn
(const IContainerColumn* column,
const IContainerColumn* afterColumn=0);
```

addCustomer — ICompany — vbsample

Description: Perform the add customer action.

Member:
```
virtual ICompany& addCustomer (const IString& name);
```

addDescending — IComboBox

Description: Inserts the line of text in descending sort order and returns the index of the item inserted.

Member:
```
virtual unsigned long addDescending (const char* text);
```

addDescending — IListBox

Description: Inserts the line of text in descending sort order and returns the index of the item inserted.

Member:
```
virtual unsigned long addDescending (const char* text);
```

addDetent — ISlider

Description: Adds a detent to the slider at the specified pixel offset from the home position.

Member:
```
virtual unsigned long addDetent (unsigned long offset);
```

addDifference — IRBag — vbcc

Description: Creates the difference between the two given collections, and adds this difference to the collection.

Member:
```
void addDifference (IABag<Element>const& bag,
IABag<Element>const& bag2);
```

addDifference — IRSet — vbcc

Description: Creates the difference between the two given collections, and adds this difference to the collection.

Member:
```
void addDifference (IASet<Element>const& collection,
IASet<Element>const& collection2);
```

addDifference — IRSortedBag — vbcc

Description: Creates the difference between the two given collections, and adds this difference to the collection.

Member:
```
void addDifference (IASortedBag<Element>const& bag,
IASortedBag<Element>const& bag2);
```

addDifference — IRSortedSet — vbcc

Description: Creates the difference between the two given collections, and adds this difference to the collection.

Member:
```
void addDifference (IASortedSet<Element>const& collection,
IASortedSet<Element>const& collection2);
```

addDrive — IVBFileDialog

Description: Adds a drive or network identifier to the Drive list in the initial dialog.

Member:
```
virtual IVBFileDialog& addDrive (const char* drive);
```

addEntryAsFirst — IMMAudioCDContents — vbmm

Description: Adds the track number to the beginning of the playback list.

Member:
```
IMMAudioCDContents& addEntryAsFirst
(unsigned long trackNumber);
```

addExtension — IFrameWindow

Description: Adds a window as a frame extension.

Member:
```
virtual IFrameWindow& addExtension (IWindow* newExtension,
Location location,
SeparatorType separator=IFrameWindow::thinLine);
```

addFileType — IVBFileDialog

Description: Adds a specified type to the drop-down box of extended-attribute types.

Member:
```
virtual IVBFileDialog& addFileType (const char* fileType);
```

addFirstPage — INotebook

Description: Adds a page as the first page in the notebook using the specified page settings.

Member:
```
virtual IPageHandle addFirstPage
(const PageSettings& pageInfo, IWindow* pageWindow=0);
```

addIntersection — IRBag — vbcc

Description: Creates the intersection of the two given collections, and adds this intersection to the collection.

Member:
```
void addIntersection (IABag<Element>const& bag,
IABag<Element>const& bag2);
```

addIntersection — IRSet — vbcc

Description: Creates the intersection of the two given collections, and adds this intersection to the collection.

Member:
```
void addIntersection (IASet<Element>const& collection,
IASet<Element>const& collection2);
```

addIntersection — IRSortedBag — vbcc

Description: Creates the intersection of the two given collections, and adds this intersection to the collection.

Member:
```
void addIntersection (IASortedBag<Element>const& bag,
IASortedBag<Element>const& bag2);
```

addIntersection — IRSortedSet — vbcc

Description: Creates the intersection of the two given collections, and adds this intersection to the collection.

Member:
```
void addIntersection (IASortedSet<Element>const& collection,
IASortedSet<Element>const& collection2);
```

addKey — IAcceleratorTable

Description: Adds the specified accelerator key to the table.

Member:
```
Boolean addKey (const IAcceleratorKey& newKey);
```

addLastPage — INotebook

Description: Adds a page as the last page in the notebook using the specified page settings.

Member:
```
virtual IPageHandle addLastPage (const PageSettings& pageInfo,
IWindow* pageWindow=0);
```

addLibraries — IHelpWindow

Description: Adds a library or list of libraries to those already used by the Information Presentation Facility (IPF).

Member:
```
virtual IHelpWindow& addLibraries
(const char* helpLibraryNames);
```

addLine — IMultiLineEdit

Description: Inserts the specified NULL-terminated text at the specified line number.

Member:
```
virtual IMultiLineEdit& addLine (const char* text,
unsigned long lineNumber,
EOLFormat type=IMultiLineEdit::cfText);
```

addLineAsLast — IMultiLineEdit

Description: Inserts the specified NULL-terminated text as the last line in the MLE.

Member:
```
virtual IMultiLineEdit& addLineAsLast (const char* text,
EOLFormat type=IMultiLineEdit::cfText);
```

addOrReplaceElementWithKey — IProfile

Description: Writes the specified numeric, text, or binary data.

Member:
```
virtual IProfile& addOrReplaceElementWithKey (const char* key,
long data, const char* applName=0);
```

addOrReplaceKey — IAcceleratorTable

Description: Adds the specified accelerator key to the table, or replaces an existing entry.

Member:
```
Boolean addOrReplaceKey (const IAcceleratorKey& newKey);
```

addPageAfter — INotebook

Description: Adds a page to the notebook following the specified page using the specified page settings.

Member:
```
virtual IPageHandle addPageAfter
(const PageSettings& pageInfo,
const IPageHandle& referencePage, IWindow* pageWindow=0);
```

addPageBefore — INotebook

Description: Adds a page to the notebook before the specified page using the specified page settings.

Member:
```
virtual IPageHandle addPageBefore
(const PageSettings& pageInfo,
const IPageHandle& referencePage, IWindow* pageWindow=0);
```

addToCell — IMultiCellCanvas

Description: Specifies the starting cell into which a window is placed.

Member:
```
virtual IMultiCellCanvas& addToCell (IWindow* childWindow,
unsigned long startingColumn, unsigned long startingRow,
unsigned long numberOfColumns=1,
unsigned long numberOfRows=1);
```

addToWindowList — IFrameWindow

Description: Adds this frame window's title as a window list entry.

Member:
```
virtual IFrameWindow& addToWindowList ();
```

addUnion — IRBag — vbcc

Description: Creates the union of the two given collections, and adds this union to the collection.

Member:
```
void addUnion (IABag<Element>const& bag,
IABag<Element>const& bag2);
```

addUnion — IRSet — vbcc

Description: Creates the union of the two given collections, and adds this union to the collection.

Member:
```
void addUnion (IASet<Element>const& collection,
IASet<Element>const& collection2);
```

addUnion — IRSortedBag — vbcc

Description: Creates the union of the two given collections, and adds this union to the collection.

Member:
```
void addUnion (IASortedBag<Element>const& bag,
IASortedBag<Element>const& bag2);
```

addUnion — IRSortedSet — vbcc

Description: Creates the union of the two given collections, and adds this union to the collection.

Member:
```
void addUnion (IASortedSet<Element>const& collection,
IASortedSet<Element>const& collection2);
```

addValue — IVBDoublePart — vbsample

Description: Perform add operator on value.

Member:
```
virtual IVBDoublePart& addValue (double addValue=1);
```

addValue — IVBLongPart — vbsample

Description: Perform add operator on value.

Member:
```
virtual IVBLongPart& addValue (long addValue=1);
```

addValue — IVBShortPart — vbsample

Description: Perform add operator on value.

Member:
```
virtual IVBShortPart& addValue (short addValue=1);
```

addValue — IVBUnsignedLongPart — vbsample

Description: Perform add operator on value.

Member:
```
virtual IVBUnsignedLongPart& addValue
(unsigned long addValue=1);
```

addValue — IVBUnsignedShortPart — vbsample

Description: Perform add operator on value.

Member:
```
virtual IVBUnsignedShortPart& addValue
(unsigned short addValue=1);
```

allElementsDo — IRBag — vbcc

Description: Calls the given function for all elements in the collection until the given function returns False.

Member:
```
IBoolean allElementsDo (IIterator<Element>& collection);
```

allElementsDo — IRDeque — vbcc

Description: Calls the given function for all elements in the collection until the given function returns False.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element const&function,void*) allFuction,
void* additionalArgument=0) const;
```

allElementsDo — IREqualitySequence — vbcc

Description: Calls the given function for all elements in the collection until the given function returns False.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element&function,void*) allFuction,
void* additionalArgument=0);
```

allElementsDo — IRHeap — vbcc

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element&function,void*) allFuction,
void* additionalArgument=0);
```

allElementsDo — IRQueue — vbcc

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element const&function,void*) allFuction,
void* additionalArgument=0) const;
```

allElementsDo — IRSequence

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo (IIterator<Element>& collection);
```

allElementsDo — IRSet — vbcc

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo (IIterator<Element>& collection);
```

allElementsDo — IRSortedBag — vbcc

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element const&function,void*) allFuction,
void* additionalArgument=0) const;
```

allElementsDo — IRSortedSet — vbcc

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element&function,void*) allFuction,
void* additionalArgument=0);
```

allElementsDo — IRStack — vbcc

Description: Calls the given function for all elements in the collection until the given function returns `False`.

Member:
```
IBoolean allElementsDo
(IBoolean(*function)(Element const&function,void*) allFuction,
void* additionalArgument=0) const;
```

andValue — IVBLongPart — vbsample

Description: Perform and operator on value.

Member:
```
virtual IVBLongPart& andValue (long andValue=1);
```

andValue — IVBShortPart — vbsample

Description: Perform and operator on value.

Member:
```
virtual IVBShortPart& andValue (short andValue=1);
```

andValue — IVBUnsignedLongPart — vbsample

Description: Perform and operator on value.

Member:
```
virtual IVBUnsignedLongPart& andValue
(unsigned long andValue=1);
```

andValue — IVBUnsignedShortPart — vbsample

Description: Perform and operator on value.

Member:
```
virtual IVBUnsignedShortPart& andValue
(unsigned short andValue=1);
```

appendText — IVBStringPart — vbsample

Description: Append to the end of text.

Member:
```
virtual IVBStringPart& appendText (const IString& appendText);
```

applyBidiSettings — IWindow

Description: Changes the bidirectional attributes of the window to the values defined in the `IWindow::BidiSettings` object.

Member:
```
virtual IWindow& applyBidiSettings
(const BidiSettings& settings, Boolean childInherit=true,
Boolean refresh=true);
```

arrangeIconView — IContainerControl

Description: Arranges the icon view.

Member:
```
virtual IContainerControl& arrangeIconView ();
```

asGUIStyle — I3StateCheckBox

Description: Converts style bits into style recognizeable by the GUI.
Member:
```
virtual unsigned long asGUIStyle (const IBitFlag& style,
Boolean extended=false);
```

asin — mathSamples — vbsample

Description: asin calculates the arcsine of x, in the range -pi/2 to pi/2 radians.
Member:
```
double asin (double x);
```

assignTextToCmdLineParm0 — IVBStringPart — vbsample

Description: Set text to command line parm 0.
Member:
```
virtual IVBStringPart& assignTextToCmdLineParm0 ();
```

assignTextToCmdLineParm1 — IVBStringPart — vbsample

Description: Set text to command line parm 1.
Member:
```
virtual IVBStringPart& assignTextToCmdLineParm1 ();
```

assignTextToDateToday — IVBStringPart — vbsample

Description: Assign text to today's date.
Member:
```
virtual IVBStringPart& assignTextToDateToday ();
```

assignTextToDefault — IVBStringPart — vbsample

Description: Assign the text attribute to `false`.
Member:
```
virtual IVBStringPart& assignTextToDefault ();
```

assignTextToEmpty — IVBStringPart — vbsample

Description: Assign text attribute to empty.
Member:
```
virtual IVBStringPart& assignTextToEmpty ();
```

assignTextToTimeNow — IVBStringPart — vbsample

Description: Assign text to time now.
Member:
```
virtual IVBStringPart& assignTextToTimeNow ();
```

assignValueToDefault — IVBBooleanPart — vbsample

Description: Assign the value attribute to `false`.
Member:
```
virtual IVBBooleanPart& assignValueToDefault ();
```

assignValueToDefault — IVBDoublePart — vbsample

Description: Assign the value attribute to false.
Member: `virtual IVBDoublePart& assignValueToDefault ();`

assignValueToDefault — IVBLongPart — vbsample

Description: Assign the value attribute to false.
Member: `virtual IVBLongPart& assignValueToDefault ();`

assignValueToDefault — IVBShortPart — vbsample

Description: Assign the value attribute to false.
Member: `virtual IVBShortPart& assignValueToDefault ();`

assignValueToDefault — IVBUnsignedLongPart — vbsample

Description: Assign the value attribute to false.
Member: `virtual IVBUnsignedLongPart& assignValueToDefault ();`

assignValueToDefault — IVBUnsignedShortPart — vbsample

Description: Assign the value attribute to false.
Member: `virtual IVBUnsignedShortPart& assignValueToDefault ();`

assignValueToE — IVBDoublePart — vbsample

Description: Assign value attribute to e.
Member: `virtual IVBDoublePart& assignValueToE ();`

assignValueToFalse — IVBBooleanPart — vbsample

Description: Assign the value attribute to false.
Member: `virtual IVBBooleanPart& assignValueToFalse ();`

assignValueToHighLimit — IVBDoublePart — vbsample

Description: Assign value attribute to high limit
Member: `virtual IVBDoublePart& assignValueToHighLimit ();`

assignValueToHighLimit — IVBLongPart — vbsample

Description: Assign value attribute to high limit
Member: `virtual IVBLongPart& assignValueToHighLimit ();`

assignValueToHighLimit — IVBShortPart — vbsample

Description: Assign value attribute to high limit
Member: `virtual IVBShortPart& assignValueToHighLimit ();`

assignValueToHighLimit — IVBUnsignedLongPart — vbsample

Description: Assign value attribute to high limit
Member: `virtual IVBUnsignedLongPart& assignValueToHighLimit ();`

assignValueToHighLimit — IVBUnsignedShortPart — vbsample

Description: Assign value attribute to high limit
Member: `virtual IVBUnsignedShortPart& assignValueToHighLimit ();`

assignValueToLowLimit — IVBDoublePart — vbsample

Description: Assign value attribute to low limit.
Member: `virtual IVBDoublePart& assignValueToLowLimit ();`

assignValueToLowLimit — IVBLongPart — vbsample

Description: Assign value attribute to low limit.
Member: `virtual IVBLongPart& assignValueToLowLimit ();`

assignValueToLowLimit — IVBShortPart — vbsample

Description: Assign value attribute to low limit.
Member: `virtual IVBShortPart& assignValueToLowLimit ();`

assignValueToLowLimit — IVBUnsignedLongPart — vbsample

Description: Assign value attribute to low limit.
Member: `virtual IVBUnsignedLongPart& assignValueToLowLimit ();`

assignValueToLowLimit — IVBUnsignedShortPart — vbsample

Description: Assign value attribute to low limit.
Member: `virtual IVBUnsignedShortPart& assignValueToLowLimit ();`

assignValueToOne — IVBDoublePart — vbsample

Description: Assign value attribute to 1 (`true`).
Member: `virtual IVBDoublePart& assignValueToOne ();`

assignValueToOne — IVBLongPart — vbsample

Description: Assign value attribute to 1 (`true`).
Member: `virtual IVBLongPart& assignValueToOne ();`

assignValueToOne — IVBShortPart — vbsample

Description: Assign value attribute to 1 (`true`).
Member: `virtual IVBShortPart& assignValueToOne ();`

assignValueToOne — IVBUnsignedLongPart — vbsample

Description: Assign value attribute to 1 (`true`).
Member: `virtual IVBUnsignedLongPart& assignValueToOne ();`

assignValueToOne — IVBUnsignedShortPart — vbsample

Description: Assign value attribute to 1 (`true`).
Member: `virtual IVBUnsignedShortPart& assignValueToOne ();`

assignValueToPI — IVBDoublePart — vbsample

Description: Assign value attribute to PI.
Member: `virtual IVBDoublePart& assignValueToPI ();`

assignValueToRandom — IVBDoublePart — vbsample

Description: Assign value to pseudo-random number
Member: `virtual IVBDoublePart& assignValueToRandom ();`

assignValueToRandom — IVBLongPart — vbsample

Description: Assign value to pseudo-random number
Member: `virtual IVBLongPart& assignValueToRandom ();`

assignValueToRandom — IVBShortPart — vbsample

Description: Assign value to pseudo-random number
Member: `virtual IVBShortPart& assignValueToRandom ();`

assignValueToRandom — IVBUnsignedLongPart — vbsample

Description: Assign value to pseudo-random number
Member: `virtual IVBUnsignedLongPart& assignValueToRandom ();`

assignValueToRandom — IVBUnsignedShortPart — vbsample

Description: Assign value to pseudo-random number
Member: `virtual IVBUnsignedShortPart& assignValueToRandom ();`

assignValueToTrue — IVBBooleanPart — vbsample

Description: Assign the value attribute to `true`.
Member: `virtual IVBBooleanPart& assignValueToTrue ();`

assignValueToZero — IVBDoublePart — vbsample

Description: Assign value attribute to 0 (`false`).
Member: `virtual IVBDoublePart& assignValueToZero ();`

assignValueToZero — IVBLongPart — vbsample

Description: Assign value attribute to 0 (false).
Member: `virtual IVBLongPart& assignValueToZero ();`

assignValueToZero — IVBShortPart — vbsample

Description: Assign value attribute to 0 (false).
Member: `virtual IVBShortPart& assignValueToZero ();`

assignValueToZero — IVBUnsignedLongPart — vbsample

Description: Assign value attribute to 0 (false).
Member: `virtual IVBUnsignedLongPart& assignValueToZero ();`

assignValueToZero — IVBUnsignedShortPart — vbsample

Description: Assign value attribute to 0 (false).
Member: `virtual IVBUnsignedShortPart& assignValueToZero ();`

atan — mathSamples — vbsample

Description: atan and atan2 calculate the arctangent of x and y/x, respectively.
Member: `double atan (double x);`

atof — stdlibSamples — vbsample

Description: atof converts a character string to a double-precision floating-point value.
Member: `double atof (const char* string);`

atoi — stdlibSamples — vbsample

Description: atoi converts a character string to an integer value.
Member: `int atoi (const char* string);`

atol — stdlibSamples — vbsample

Description: atol converts a character string to a long value.
Member: `long int atol (const char* string);`

b2c — IString

Description: Converts a string of binary digits to a normal string of characters.
Member: `static IString b2c (const IString& aString);`

b2d — IString

Description: Converts a string of binary digits to a string of decimal digits.
Member:
```
static IString b2d (const IString& aString);
```

b2x — IString

Description: Converts a string of binary digits to a string of hexadecimal digits.
Member:
```
static IString b2x (const IString& aString);
```

beginEdit — ICnrEditHandler

Description: Called when an edit window has been opened in the container.
Member:
```
virtual Boolean beginEdit (ICnrBeginEditEvent& event);
```

beginFlashing — IFrameWindow

Description: Starts flashing the frame.
Member:
```
virtual IFrameWindow& beginFlashing ();
```

beginUsingFont — IFont

Description: Sets the presentation space to use the font.
Member:
```
virtual IFont& beginUsingFont
(const IPresSpaceHandle& presSpaceHandle);
```

c2b — IString

Description: Converts a normal string of characters to a string of binary digits.
Member:
```
IString& c2b ();
```

c2d — IString

Description: Converts a normal string of characters to a string of decimal digits.
Member:
```
IString& c2d ();
```

c2x — IString

Description: Converts a normal string of characters to a string of hexadecimal digits.
Member:
```
static IString c2x (const IString& aString);
```

calloc — stdlibSamples — vbsample

Description: calloc reserves storage space for an array of num elements, each of length size bytes.
Member:
```
void calloc (size_t num, size_t size);
```

capturePointer — IWindow

Description: If capture is `true`, pointer events will be sent only to this window even if the pointer is outside of this window.

Member:
```
virtual IWindow& capturePointer (Boolean capture=true);
```

ceil — mathSamples — vbsample

Description: ceil computes the smallest integer that is greater than or equal to x.

Member:
```
double ceil (double x);
```

ceilValue — IVBDoublePart — vbsample

Description: Perform ceil function on value.

Member:
```
virtual IVBDoublePart& ceilValue ();
```

center — IString

Description: Centers the receiver within a string of the specified length.

Member:
```
static IString center (const IString& aString,
unsigned length, char padCharacter=' ');
```

centerAt — IRectangle

Description: Moves the rectangle so that its center is at the specified point.

Member:
```
IRectangle& centerAt (const IPoint& point);
```

centeredAt — IRectangle

Description: Same as `IRectangle::centerAt`, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle centeredAt (const IPoint& point) const;
```

change — I0String

Description: Changes occurrences of a specified pattern to a specified replacement string.

Member:
```
static I0String change (const IString& aString,
const IString& inputString, const IString& outputString,
unsigned startPos=0, unsigned numChanges=( unsigned )
UINT_MAX);
```

change — IString

Description: Changes occurrences of a specified pattern to a specified replacement string.

Member:
```
static IString change (const IString& aString,
const IString& inputString, const char* pOutputString,
unsigned startPos=1, unsigned numChanges=( unsigned )
UINT_MAX);
```

changeTextToLowerCase — IVBStringPart — vbsample

Description: Change the text to lower case.

Member:
```
virtual IVBStringPart& changeTextToLowerCase ();
```

changeTextToUpperCase — IVBStringPart — vbsample

Description: Change the text to upper case.

Member:
```
virtual IVBStringPart& changeTextToUpperCase ();
```

charType — IString

Description: Returns the type of the character at the specified index.

Member:
```
IStringEnum::CharType charType (unsigned index) const;
```

charWidth — IFont

Description: Returns the width of a specific single-byte character.

Member:
```
unsigned long charWidth (char c) const;
```

chmod — ioSamples — vbsample

Description: chmod changes the permission setting of the file specified by pathname.

Member:
```
int chmod (char* pathname, int pmode);
```

clear — IEntryField

Description: Replaces the selected text in the entry field with blanks.

Member:
```
virtual IEntryField& clear (unsigned long timestamp=0);
```

clear — IMultiLineEdit

Description: Replaces the currently selected text with blank spaces.

Member:
```
virtual IMultiLineEdit& clear (unsigned long timestamp=0);
```

clearDefaultTransparentColor — IToolBarButton

Description: Resets the default transparent color so that no color will be treated as transparent for subsequent buttons.

Member:
```
static void clearDefaultTransparentColor ();
```

click — IButton

Description: Simulates the user clicking on the button control using the mouse selection button.

Member:
```
virtual IButton& click ();
```

clientRectFor — IFrameWindow

Description: Returns a rectangle indicating the size and position the client window has if the frame is sized and positioned according to the specified rectangle.

Member:
```
virtual IRectangle clientRectFor (const IRectangle& frameRect)
const;
```

close — IClipboard

Description: Closes the clipboard.

Member:
```
virtual IClipboard& close ();
```

close — IDynamicLinkLibrary

Description: Closes a dynamic link library and decrements the reference count.

Member:
```
virtual IDynamicLinkLibrary& close ();
```

close — IFrameWindow

Description: Closes the frame window.

Member:
```
virtual IFrameWindow& close ();
```

close — IMMDevice — vbmm

Description: Closes a device.

Member:
```
virtual IMMDevice& close (CallType call=IMMDevice::wait);
```

close — ioSamples — vbsample

Description: close closes the file associated with the handle.

Member:
```
int close (int handle);
```

closeDoor — IMMRemovableMedia — vbmm

Description: Retracts the tray and closes the door, if possible.

Member:
```
virtual IMMRemovableMedia& closeDoor
(CallType call=IMMDevice::wait);
```

closeEdit — IContainerControl

Description: Closes an open edit field.

Member:
```
virtual IContainerControl& closeEdit ();
```

collapse — IContainerControl

Description: Collapses the descendents of an object in the tree view.

Member:
```
virtual IContainerControl& collapse
(IContainerObject* object);
```

collapse — IToolBar

Description: Displays only the title of tool bar when it is floating.

Member:
```
virtual IToolBar& collapse ();
```

collapseTree — IContainerControl

Description: Collapses all nodes of a tree view.

Member:
```
virtual IContainerControl& collapseTree ();
```

columnAt — IContainerControl

Description: Retrieves the column at the specified 0-based index or cursor.

Member:
```
IContainerColumn* columnAt (unsigned long index) const;
```

columnUnderPoint — IContainerControl

Description: Retrieves the column under the specified point, in window coordinates.

Member:
```
IContainerColumn* columnUnderPoint (const IPoint& point)
const;
```

columnWidth — IMultiCellCanvas

Description: Returns the current width of the specified column, in pixels.

Member:
```
unsigned long columnWidth (unsigned long column) const;
```

compare — IADeque — vbcc

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare (IADeque<Element>const& collection,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

compare — IAEqualitySequence — vbcc

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare
(IAEqualitySequence<Element>const& collection,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

compare — IAQueue — vbcc

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare (IAQueue<Element>const& collection,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

compare — IASequence

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare (IASequence<Element>const& collection,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

compare — IASortedBag — vbcc

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare (IASortedBag<Element>const& bag,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

compare — IASortedSet — vbcc

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare (IASortedSet<Element>const& collection,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

compare — IAStack — vbcc

Description: Lexicographically compares the collection with the given collection.

Member:
```
virtual long compare (IAStack<Element>const& collection,
long(*comparisonFunction)(Element const&,Element const&)
compareFunction) const;
```

connectedDeviceId — IMMDevice — vbmm

Description: Returns the id of the device that is connected to the identified connector of this device.

Member:
```
unsigned long connectedDeviceId (ConnectorType type,
CallType call=IMMDevice::wait) const;
```

containerFromHandle — IContainerControl

Description: Retrieves a container from the list using the specified handle.

Member:
```
static IContainerControl* containerFromHandle
(const IWindowHandle& handle);
```

contains — IRBag — vbcc

Description: Returns True if the collection contains an element equal to the given element.

Member:
```
IBoolean contains (Element const& element) const;
```

contains — IRectangle

Description: If the rectangle contains the specified point or rectangle, true is returned.

Member:
```
Boolean contains (const IPoint& point) const;
```

contains — IREqualitySequence — vbcc

Description: Returns True if the collection contains an element equal to the given element.

Member:
```
IBoolean contains (Element const& element) const;
```

contains — IRSet — vbcc

Description: Returns True if the collection contains an element equal to the given element.

Member:
```
IBoolean contains (Element const& element) const;
```

contains — IRSortedBag — vbcc

Description: Returns True if the collection contains an element equal to the given element.

Member:
```
IBoolean contains (Element const& element) const;
```

contains — IRSortedSet — vbcc

Description: Returns True if the collection contains an element equal to the given element.

Member:
```
IBoolean contains (Element const& element) const;
```

containsAllFrom — IAEqualityCollection — vbcc

Description: Returns True if all the elements of the given collection are contained in the collection.

Member:
```
virtual IBoolean containsAllFrom
(IACollection<Element>const& collection) const;
```

containsApplication — IProfile

Description: If the profile contains data for the specified application name, returns `true`.

Member:
```
virtual Boolean containsApplication (const char* applName)
const;
```

containsKeyLike — IAcceleratorTable

Description: Identifies if the accelerator table contains an entry with the same keystroke (the same character or virtual key and the same Alt, Ctrl, Shift modifiers) as the specified accelerator key.

Member:
```
Boolean containsKeyLike (const IAcceleratorKey& key) const;
```

containsKeyName — IProfile

Description: If the profile contains data for the specified application and key pair, returns `true`.

Member:
```
virtual Boolean containsKeyName (const char* key,
const char* applName=0) const;
```

containsObject — IContainerControl

Description: If the specified object is in the container, `true` is returned.

Member:
```
Boolean containsObject (const IContainerObject* object) const;
```

convertToGUIStyle — IMenuItem

Description: Use this function to convert style bits into the style value that can be processed by the GUI.

Member:
```
virtual unsigned long convertToGUIStyle
(const IBitFlag& style, Boolean extendedOnly=false) const;
```

convertToGUIStyle — IWindow

Description: Use this function to convert style bits into the style value that can be processed by the GUI.

Member:
```
virtual unsigned long convertToGUIStyle
(const IBitFlag& style, Boolean extendedOnly=false) const;
```

convertToWorkspace — IContainerControl

Description: Converts a rectangle from container window coordinates to workspace coordinates.

Member:
```
IRectangle convertToWorkspace
(const IRectangle& windowRectangle, Boolean rightWindow=false)
const;
```

copy

copy — IACollection

Description: Copies the given collection to this collection

Member:
```
virtual void copy (IACollection<Element>const& collection);
```

copy — IADeque — vbcc

Description: Copies the given collection to this collection

Member:
```
virtual void copy (IADeque<Element>const& collection);
```

copy — IAQueue — vbcc

Description: Copies the given collection to this collection

Member:
```
virtual void copy (IAQueue<Element>const& collection);
```

copy — IAStack — vbcc

Description: Copies the given collection to this collection

Member:
```
virtual void copy (IAStack<Element>const& collection);
```

copy — IEntryField

Description: Copies the selected text to the clipboard.

Member:
```
virtual IEntryField& copy (unsigned long timestamp=0);
```

copy — IMMRecordable — vbmm

Description: Copies data from the passed in start position to the passed in end position into the clipboard.

Member:
```
virtual IMMRecordable& copy (const IMMTime& from=IMMTime ( ),
const IMMTime& to=IMMTime ( ), CallType call=IMMDevice::wait)
const;
```

copy — IMultiLineEdit

Description: Copies the currently selected text from the MLE to the clipboard.

Member:
```
virtual IMultiLineEdit& copy (unsigned long timestamp=0);
```

copy — IString

Description: Replaces the receiver's contents with a specified number of replications of itself.

Member:
```
static IString copy (const IString& aString,
unsigned numCopies);
```

copyObjectTo — IContainerControl

Description: Copies an object and its descendents to a new location in the tree view.

Member:
```
virtual IContainerObject* copyObjectTo
(IContainerObject* copyObject,
IContainerObject* parentObject=0,
IContainerControl* newContainer=0,
IContainerObject* afterObject=0,
const IPoint& iconViewLocation=IPoint ( 0 , 0 ));
```

copyValueToDefault — IVBBooleanPart — vbsample

Description: Copy value to default value.

Member:
```
virtual IVBBooleanPart& copyValueToDefault ();
```

copyValueToDefault — IVBDoublePart — vbsample

Description: Copy value to default value.

Member:
```
virtual IVBDoublePart& copyValueToDefault ();
```

copyValueToDefault — IVBLongPart — vbsample

Description: Copy value to default value.

Member:
```
virtual IVBLongPart& copyValueToDefault ();
```

copyValueToDefault — IVBShortPart — vbsample

Description: Copy value to default value.

Member:
```
virtual IVBShortPart& copyValueToDefault ();
```

copyValueToDefault — IVBUnsignedLongPart — vbsample

Description: Copy value to default value.

Member:
```
virtual IVBUnsignedLongPart& copyValueToDefault ();
```

copyValueToDefault — IVBUnsignedShortPart — vbsample

Description: Copy value to default value.

Member:
```
virtual IVBUnsignedShortPart& copyValueToDefault ();
```

cos — mathSamples — vbsample

Description: cos calculates the cosine of x.

Member:
```
double cos (double x);
```

cosh — mathSamples — vbsample

Description: cosh calculates the hyperbolic cosine of x.

Member:
```
double cosh (double x);
```

creat — ioSamples — vbsample

Description: creat either creates a new file or opens and truncates an existing file.

Member:
```
int creat (char* pathname, int pmode);
```

csid — stdlibSamples — vbsample

Description: csid queries the locale and determines the character-set identifier for the specified character c.

Member:
```
int csid (const char* c);
```

cueForPlayback — IMMPlayableDevice — vbmm

Description: Cues the device for playback.

Member:
```
virtual IMMPlayableDevice& cueForPlayback
(CallType call=IMMDevice::wait);
```

cueForRecording — IMMRecordable — vbmm

Description: Cues the device for recording.

Member:
```
virtual IMMRecordable& cueForRecording
(CallType call=IMMDevice::nowait);
```

cut — IEntryField

Description: Removes the selected text from the entry field control and puts it in the clipboard.

Member:
```
virtual IEntryField& cut (unsigned long timestamp=0);
```

cut — IMMRecordable — vbmm

Description: Cuts the data from the passed in start position to the passed in end position into the clipboard.

Member:
```
virtual IMMRecordable& cut (const IMMTime& from=IMMTime ( ),
const IMMTime& to=IMMTime ( ), CallType call=IMMDevice::wait);
```

cut — IMultiLineEdit

Description: Copies the currently selected text from the MLE to the clipboard and then deletes the selected text from the MLE.

Member:
```
virtual IMultiLineEdit& cut (unsigned long timestamp=0);
```

d2b — IString

Description: Converts a string of decimal digits to a string of binary digits.
Member:
```
static IString d2b (const IString& aString);
```

d2c — IString

Description: Converts a string of decimal digits to a normal string of characters.
Member:
```
IString& d2c ();
```

d2x — IString

Description: Converts a string of decimal digits to a string of hexadecimal digits.
Member:
```
IString& d2x ();
```

data — IClipboard

Description: Returns a void* value.
Member:
```
virtual void* data (const char* format);
```

dataAsDate — IContainerColumn

Description: Returns the data referenced by this column in the specified object.
Member:
```
IDate dataAsDate (const IContainerObject* object) const;
```

dataAsIcon — IContainerColumn

Description: Returns the data referenced by this column in the specified object.
Member:
```
IPointerHandle dataAsIcon (const IContainerObject* object)
const;
```

dataAsNumber — IContainerColumn

Description: Returns the data referenced by this column in the specified object.
Member:
```
unsigned long dataAsNumber (const IContainerObject* object)
const;
```

dataAsString — IContainerColumn

Description: Returns the data referenced by this column in the specified object.
Member:
```
IString dataAsString (const IContainerObject* object) const;
```

dataAsTime — IContainerColumn

Description: Returns the data referenced by this column in the specified object.
Member:
```
ITime dataAsTime (const IContainerObject* object) const;
```

dayName — IDate

Description: Returns the name of the receiver's day of the week.

Member:
```
static IString dayName (DayOfWeek aDay);
```

daysInMonth — IDate

Description: Returns the number of days in a specified month of a specified year.

Member:
```
static int daysInMonth (Month aMonth, int aYear);
```

daysInYear — IDate

Description: Returns the number of days in a specified year.

Member:
```
static int daysInYear (int aYear);
```

deleteElementWithApplication — IProfile

Description: Removes the data for all keys for the specified application name.

Member:
```
virtual IProfile& deleteElementWithApplication
(const char* applName=0);
```

deleteElementWithKey — IProfile

Description: Removes the data at the specified application and key name pair.

Member:
```
virtual IProfile& deleteElementWithKey (const char* key,
const char* applName=0);
```

deletePendingEvents — IMMDevice — vbmm

Description: Removes all of the passed in event types from the queue.

Member:
```
virtual IMMDevice& deletePendingEvents
(EventType event=IMMDevice::allEvents);
```

deleteSelection — IMMRecordable — vbmm

Description: Deletes data from the passed in start position to the passed in end position.

Member:
```
virtual IMMRecordable& deleteSelection
(const IMMTime& from=IMMTime ( ), const IMMTime& to=IMMTime (
), CallType call=IMMDevice::wait);
```

dequeue — IRQueue — vbcc

Description: Copies the first element of the collection to the given element, and removes it from the collection.

Member:
```
void dequeue (Element& element);
```

descendentsOf — IContainerControl

Description: Returns the set of all descendents of an object in the tree view.

Member:
```
ICnrObjectSet descendentsOf (IContainerObject* parentObject)
const;
```

deselect — IBaseComboBox

Description: Removes the selection state from the specified item (0-based).

Member:
```
virtual IBaseComboBox& deselect (unsigned long index);
```

deselect — IBaseListBox

Description: Removes the selection state from the 0-based item.

Member:
```
virtual IBaseListBox& deselect (unsigned long index);
```

deselect — ICollectionViewComboBox

Description: Removes the selection state from the specified collection position (1 based) item.

Member:
```
virtual ICollectionViewComboBox<Element,Collection>& deselect
(unsigned long collectionPosition);
```

deselect — ICollectionViewListBox

Description: Removes the selection state from the collection position (1 based) item.

Member:
```
virtual ICollectionViewListBox<Element,Collection>& deselect
(unsigned long collectionPosition);
```

deselect — ISettingButton

Description: Deselects the button.

Member:
```
virtual ISettingButton& deselect ();
```

deselect — IVBContainerControl

Description: Deselect the specified object.

Member:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
deselect (unsigned long collectionPosition);
```

deselectAll — IBaseComboBox

Description: Removes the selection state from the currently selected item in the list box.

Member:
```
virtual IBaseComboBox& deselectAll ();
```

deselectAll — IBaseListBox

Description: Removes the selection state from all items in the list box.

Member:
```
virtual IBaseListBox& deselectAll ();
```

desktopWindow — staticWindowSamples — vbsample

Description: Static function to return the desktop window.

Member:
```
static IWindow* IWindow::desktopWindow ();
```

detailsObjectRectangle — IContainerControl

Description: Returns the details view rectangle.

Member:
```
IRectangle detailsObjectRectangle
(const IContainerObject* object,
const IContainerColumn* column) const;
```

detentPosition — ISlider

Description: Returns the offset of a detent from the home position, in pixels.

Member:
```
virtual unsigned long detentPosition (unsigned long detentId)
const;
```

differenceWith — IRBag — vbcc

Description: Makes the collection the difference between the collection and the given collection.

Member:
```
void differenceWith (IABag<Element>const& bag);
```

differenceWith — IRSet — vbcc

Description: Makes the collection the difference between the collection and the given collection.

Member:
```
void differenceWith (IASet<Element>const& collection);
```

differenceWith — IRSortedBag — vbcc

Description: Makes the collection the difference between the collection and the given collection.

Member:
```
void differenceWith (IASortedBag<Element>const& bag);
```

differenceWith — IRSortedSet — vbcc

Description: Makes the collection the difference between the collection and the given collection.

Member:
```
void differenceWith (IASortedSet<Element>const& collection);
```

disable — IHandler

Description: Disables the handler so that it processes no window events.
Member:
```
virtual IHandler& disable ();
```

disable — IInfoArea

Description: Prevents keyboard and mouse input from being sent to the window.
Member:
```
virtual IWindow& IWindow::disable ();
```

disable — IMenuItem

Description: Disable the menu item.
Member:
```
IMenuItem& setDisabled ();
```

disable — IWindow

Description: Prevents keyboard and mouse input from being sent to the window.
Member:
```
virtual IWindow& disable ();
```

disableAnimateWhenLatched — IAnimatedButton

Description: Removes the animateWhenLatched style for the button.
Member:
```
virtual IAnimatedButton& disableAnimateWhenLatched ();
```

disableAudio — IMMDevice — vbmm

Description: Turns off the audio for the device.
Member:
```
virtual IMMDevice& disableAudio
(AudioChannel channel=IMMDevice::all,
const IMMMillisecondTime& over=IMMMillisecondTime ( ),
CallType call=IMMDevice::wait);
```

disableAutoLatch — ICustomButton

Description: Disables the autoLatch style.
Member:
```
virtual ICustomButton& disableAutoLatch ();
```

disableAutoPlay — IMMAudioCD — vbmm

Description: Sets autoplay off.
Member:
```
IMMAudioCD& disableAutoPlay ();
```

disableAutoScroll — IEntryField

Description: Disables the style autoScroll on an entry field control.
Member:
```
virtual IEntryField& disableAutoScroll ();
```

disableAutoSelect — I3StateCheckBox

Description: Removes the autoSelect style from the three-state check box control.

Member:
```
virtual I3StateCheckBox& disableAutoSelect ();
```

disableAutoSelect — ICheckBox

Description: Removes the style autoSelect from the check box control.

Member:
```
virtual ICheckBox& disableAutoSelect ();
```

disableAutoSelect — IRadioButton

Description: Removes the style autoSelect from the radio button control.

Member:
```
virtual IRadioButton& disableAutoSelect ();
```

disableAutoTab — IEntryField

Description: Disables the style autoTab on an entry field control.

Member:
```
virtual IEntryField& disableAutoTab ();
```

disableCaching — IContainerControl

Description: Stops the dispatch of ICnrQueryDeltaEvents to an ICnrHandler.

Member:
```
virtual IContainerControl& disableCaching ();
```

disableCommand — IEntryField

Description: Disables the style command on an entry field control.

Member:
```
virtual IEntryField& disableCommand ();
```

disableConnector — IMMDevice — vbmm

Description: Turns off the connector specified by type for this device.

Member:
```
virtual IMMDevice& disableConnector (ConnectorType type,
CallType call=IMMDevice::wait);
```

disableContinuousPlay — IMMAudioCD — vbmm

Description: Sets continuous play off.

Member:
```
IMMAudioCD& disableContinuousPlay ();
```

disableCursorSelect — IRadioButton

Description: Adds the radio button style noCursorSelect.

Member:
```
virtual IRadioButton& disableCursorSelect ();
```

disableDataUpdate — IBaseSpinButton

Description: Disables direct editing of the spin field data by the end user.

Member:
```
virtual IBaseSpinButton& disableDataUpdate ();
```

disableDataUpdate — IContainerColumn

Description: Prevents editing of the column data.

Member:
```
virtual IContainerColumn& disableDataUpdate ();
```

disableDataUpdate — IContainerControl

Description: Disables direct editing of the text of an object.

Member:
```
virtual IContainerControl& disableDataUpdate
(IContainerObject* object);
```

disableDataUpdate — IContainerObject

Description: Disables user updates of this object.

Member:
```
virtual IContainerObject& disableDataUpdate
(IContainerControl* container=0);
```

disableDataUpdate — IEntryField

Description: Prevents the insertion or changing of characters in the entry field's text.

Member:
```
virtual IEntryField& disableDataUpdate ();
```

disableDataUpdate — IMultiLineEdit

Description: Prevents changes to the current MLE text.

Member:
```
virtual IMultiLineEdit& disableDataUpdate ();
```

disableDefault — IPushButton

Description: Removes the style defaultButton from the push button.

Member:
```
virtual IPushButton& disableDefault ();
```

disableDragDelete — IToolBarButton

Description: Sets the noDragDelete style for the tool bar button to prevent it from being dropped on a Workplace Shell shredder object.

Member:
```
virtual IToolBarButton& disableDragDelete ();
```

disableDragDrop — IToolBar

Description: Prevents dragging of the tool bar, as well as the objects that reside on the tool bar, such as a tool bar button.

Member:
```
virtual IToolBar& disableDragDrop ();
```

disableDragLines — IMultiCellCanvas

Description: Removes the style dragLines from a multiple-cell canvas.
Member: `virtual IMultiCellCanvas& disableDragLines ();`

disableDrawBackground — IContainerControl

Description: Stops the dispatch of .*ICnrDrawBackgroundEvents ICnrDrawBackgroundEvent to an .*ICnrDrawHandler.
Member: `virtual IContainerControl& disableDrawBackground ();`

disableDrawItem — IBaseListBox

Description: Disables the style drawItem for the list box.
Member: `virtual IBaseListBox& disableDrawItem ();`

disableDrawItem — IContainerControl

Description: Stops the dispatch of .*ICnrDrawItemEvents ICnrDrawItemEvent to an .*ICnrDrawHandler.
Member:
```
virtual IContainerControl& disableDrawItem ();
```

disableDrawItem — IProgressIndicator

Description: Sets the handleDrawItem style to off.
Member:
```
virtual IProgressIndicator& disableDrawItem ();
```

disableDrop — IContainerControl

Description: Disables an object from receiving a drop.
Member:
```
virtual IContainerControl& disableDrop
(IContainerObject* object);
```

disableDrop — IContainerObject

Description: Prevents any object from being dropped on this object.
Member:
```
virtual IContainerObject& disableDrop
(IContainerControl* container=0);
```

disableExtendedSelect — IBaseListBox

Description: Disables the style extendedSelect for the list box.
Member:
```
virtual IBaseListBox& disableExtendedSelect ();
```

disableFastSpin — IBaseSpinButton

Description: Disables fast spinning of the spin button.
Member:
```
virtual IBaseSpinButton& disableFastSpin ();
```

disableFillBackground — IStaticText

Description: Draws text over the current background.
Member: `virtual IStaticText& disableFillBackground ();`

disableFlyOverHelp — IVBFlyText

Description: Removes flyover help.
Member: `IVBFlyText& disableFlyOverHelp ();`

disableGridLines — IMultiCellCanvas

Description: Removes the style gridLines from a multiple-cell canvas.
Member: `virtual IMultiCellCanvas& disableGridLines ();`

disableGroup — IControl

Description: Removes the group style from a control.
Member: `virtual IControl& disableGroup ();`

disableHalftone — IStaticText

Description: Removes the halftone from the text.
Member: `virtual IStaticText& disableHalftone ();`

disableHeadingUpdate — IContainerColumn

Description: Prevents editing of the heading.
Member: `virtual IContainerColumn& disableHeadingUpdate ();`

disableHeadphones — IMMMasterAudio — vbmm

Description: Turns the headphones' settings off.
Member:
```
virtual IMMMasterAudio& disableHeadphones
(IMMDevice::CallType call=IMMDevice::wait);
```

disableHelp — IPushButton

Description: Removes the style help from the push button.
Member: `virtual IPushButton& disableHelp ();`

disableInsertMode — IEntryField

Description: Enables the entry field for overtype mode and changes the cursor's appearance.
Member: `virtual IEntryField& disableInsertMode ();`

disableLatching — ICustomButton

Description: Sets the style of the button so that the user cannot latch or unlatch it.

Member:
```
virtual ICustomButton& disableLatching ();
```

disableMargin — IEntryField

Description: Disables the style margin on an entry field control.

Member:
```
virtual IEntryField& disableMargin ();
```

disableMisfitFiltering — IToolBar

Description: Removes the filterMisfits style from the tool bar.

Member:
```
IToolBar& disableMisfitFiltering ();
```

disableMonitoring — IMMAmpMixer — vbmm

Description: Does not allow the signal from an input device to be heard as it is being routed to another device.

Member:
```
virtual IMMAmpMixer& disableMonitoring
(CallType call=IMMDevice::wait);
```

disableMouseClickFocus — IButton

Description: Prevents the button from receiving the focus when the user selects it using the mouse.

Member:
```
virtual IButton& disableMouseClickFocus ();
```

disableMultipleSelect — IBaseListBox

Description: Disables the style multipleSelect for the list box.

Member:
```
virtual IBaseListBox& disableMultipleSelect ();
```

disableNoAdjustPosition — IBaseListBox

Description: Disables the style noAdjustPosition for the list box.

Member:
```
virtual IBaseListBox& disableNoAdjustPosition ();
```

disableNotification — IRBag — vbcc

Description: Stops the object from sending notifications to registered observers.

Member:
```
virtual IStandardNotifier& disableNotification ();
```

disableNotification — IRDeque — vbcc

Description: Stops the object from sending notifications to registered observers.

Member:
```
virtual IStandardNotifier& disableNotification ();
```

disableNotification — IREqualitySequence — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRHeap — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRQueue — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRSequence

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRSet — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRSortedBag — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRSortedSet — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IRStack — vbcc

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IStandardNotifier

Description: Stops the object from sending notifications to registered observers.
Member: `virtual IStandardNotifier& disableNotification ();`

disableNotification — IWindow

Description: Causes the window to stop sending notifications to all added observer objects.
Member: `virtual IWindow& disableNotification ();`

disableRibbonStrip — IProgressIndicator

Description: Sets the ribbonStrip style to off.
Member: `virtual IProgressIndicator& disableRibbonStrip ();`

disableSizeToGraphic — IGraphicPushButton

Description: Removes the style sizeToGraphic.
Member: `virtual IGraphicPushButton& disableSizeToGraphic ();`

disableSnapToTick — IProgressIndicator

Description: Sets the snapToTickMark style to off.
Member: `virtual IProgressIndicator& disableSnapToTick ();`

disableSpeakers — IMMMasterAudio — vbmm

Description: Turns the speakers' setting off.
Member:
```
virtual IMMMasterAudio& disableSpeakers
(IMMDevice::CallType call=IMMDevice::wait);
```

disableStrikeout — IStaticText

Description: Removes the line drawn through the text.
Member: `virtual IStaticText& disableStrikeout ();`

disableSystemCommand — IPushButton

Description: Removes the style systemCommand from the push button.
Member: `virtual IPushButton& disableSystemCommand ();`

disableTabStop — IControl

Description: Removes the tabStop style from a control, therefore preventing this control from being tabbed to.
Member: `virtual IControl& disableTabStop ();`

disableTitleUpdate — IContainerControl

Description: Disables the container title from being edited.
Member: `virtual IContainerControl& disableTitleUpdate ();`

disableTrace — ITrace

Description: Disables trace entries from being written.
Member: `static void disableTrace ();`

disableUnderscore — IStaticText

Description: Removes the line beneath the text.
Member:
```
virtual IStaticText& disableUnderscore ();
```

disableUpdate — IWindow

Description: Prevents changes to a window from being drawn on the screen.
Member:
```
virtual IWindow& disableUpdate ();
```

disableWordWrap — IMultiLineEdit

Description: Disables word-wrap mode in the MLE.
Member:
```
virtual IMultiLineEdit& disableWordWrap ();
```

disableWriteLineNumber — ITrace

Description: Disables the tracing of line number information.
Member:
```
static void disableWriteLineNumber ();
```

disableWritePrefix — ITrace

Description: Disables the writing of the process ID, the thread ID, and the output line number to trace.
Member:
```
static void disableWritePrefix ();
```

discard — IEntryField

Description: Deletes the selected text.
Member:
```
virtual IEntryField& discard ();
```

discard — IMultiLineEdit

Description: Deletes all currently selected text.
Member:
```
virtual IMultiLineEdit& discard ();
```

dismiss — IFrameWindow

Description: Closes a frame window by hiding it and, if modal, completes the calls to `IFrameWindow::showModally` that triggered the window's display.
Member:
```
virtual IFrameWindow& dismiss (unsigned long result=0);
```

dispatchRemainingHandlers — IWindow

Description: Dispatches the event to all handlers that the event has not yet been dispatched to.
Member:
```
Boolean dispatchRemainingHandlers (IEvent& event,
Boolean callDefProc=true);
```

distanceFrom — IPair

Description: Returns the distance from some other ordered pair.

Member:
```
double distanceFrom (const IPair& aPair) const;
```

div — stdlibSamples — vbsample

Description: div calculates the quotient and remainder of the division of numerator by denominator.

Member:
```
div_t div (int numerator, int denominator);
```

divideValue — IVBDoublePart — vbsample

Description: Perform divide operator on value.

Member:
```
virtual IVBDoublePart& divideValue (double divideValue=2);
```

divideValue — IVBLongPart — vbsample

Description: Perform divide operator on value.

Member:
```
virtual IVBLongPart& divideValue (long divideValue=2);
```

divideValue — IVBShortPart — vbsample

Description: Perform divide operator on value.

Member:
```
virtual IVBShortPart& divideValue (short divideValue=2);
```

divideValue — IVBUnsignedLongPart — vbsample

Description: Perform divide operator on value.

Member:
```
virtual IVBUnsignedLongPart& divideValue
(unsigned long divideValue=2);
```

divideValue — IVBUnsignedShortPart — vbsample

Description: Perform divide operator on value.

Member:
```
virtual IVBUnsignedShortPart& divideValue
(unsigned short divideValue=2);
```

dotProduct — IPair

Description: Returns the dot product with another ordered pair.

Member:
```
long dotProduct (const IPair& aPair) const;
```

dup — ioSamples — vbsample

Description: dup associates a second file handle with a currently open file.

Member:
```
int dup (int handle);
```

dup2 — ioSamples — vbsample

Description: dup2 makes handle2 refer to the currently open file associated with handle1.

Member:
```
int dup2 (int handle1, int handle2);
```

editColumnTitle — IContainerControl

Description: Opens an edit field on a specified column's heading.

Member:
```
virtual IContainerControl& editColumnTitle
(IContainerColumn* column);
```

editContainerTitle — IContainerControl

Description: Opens an edit field on the container title.

Member:
```
virtual IContainerControl& editContainerTitle ();
```

editObject — IContainerControl

Description: Opens an edit field on a specified object.

Member:
```
virtual IContainerControl& editObject
(IContainerObject* object, IContainerColumn* column=0);
```

elementAt — ITextSpinButton

Description: Returns the string at the cursor or index position.

Member:
```
virtual IString elementAt (unsigned long index) const;
```

elementAtPosition — IRDeque — vbcc

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementAtPosition — IREqualitySequence — vbcc

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementAtPosition — IRQueue — vbcc

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementAtPosition — IRSequence

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementAtPosition — IRSortedBag — vbcc

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementAtPosition — IRSortedSet — vbcc

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementAtPosition — IRStack — vbcc

Description: Returns a reference to the element at the given position in the collection.

Member:
```
Element const& elementAtPosition (IPosition position) const;
```

elementWithKey — IProfile

Description: Reads the application data and returns it as an `IString`.

Member:
```
virtual IString elementWithKey (const char* key,
const char* applName=0) const;
```

empty — IClipboard

Description: Clears the clipboard of all formats of data and establishes the window provided on the constructor as the clipboard owner.

Member:
```
virtual IClipboard& empty ();
```

enable — IInfoArea

Description: Enables the window to accept keyboard and mouse input.

Member:
```
virtual IWindow& IWindow::enable (Boolean enabled=true);
```

enable — IMenuItem

Description: Enable the menu item.

Member:
```
IMenuItem& setDisabled (void c=false);
```

enable — IWindow

Description: Enables the window to accept keyboard and mouse input.

Member:
```
virtual IWindow& enable (Boolean enabled=true);
```

enableConnector — IMMDevice — vbmm

Description: Turns on or turns off the connector specified by type for this device.

Member:
```
virtual IMMDevice& enableConnector (ConnectorType type,
Boolean enable=true, CallType call=IMMDevice::wait);
```

enableDataUpdate — IContainerControl

Description: Enables direct editing of the text of a specified object.

Member:
```
virtual IContainerControl& enableDataUpdate
(IContainerObject* object, Boolean enable=true);
```

enableDrop — IContainerControl

Description: Enables a specified object to be the target of a drag event.

Member:
```
virtual IContainerControl& enableDrop
(IContainerObject* object, Boolean enable=true);
```

enableHeadphones — IMMMasterAudio — vbmm

Description: Turns the headphones' settings either off or on.

Member:
```
virtual IMMMasterAudio& enableHeadphones (Boolean enable=true,
IMMDevice::CallType call=IMMDevice::wait);
```

enableSpeakers — IMMMasterAudio — vbmm

Description: Turns the speakers' setting either on or off.

Member:
```
virtual IMMMasterAudio& enableSpeakers (Boolean enable=true,
IMMDevice::CallType call=IMMDevice::wait);
```

enableTrace — ITrace

Description: Enables trace entries to be written.

Member:
```
static void enableTrace ();
```

enableUpdate — IWindow

Description: Draws changes to a window on the screen.

Member:
```
virtual IWindow& enableUpdate (Boolean enableWindow=true);
```

enableWriteLineNumber — ITrace

Description: Enables the tracing of line number information.

Member:
```
static void enableWriteLineNumber ();
```

enableWritePrefix — ITrace

Description: Enables the writing of the process ID, the thread ID, and the output line number to trace.

Member:
```
static void enableWritePrefix ();
```

endEdit — ICnrEditHandler

Description: Called when editing has ended.

Member:
```
virtual Boolean endEdit (ICnrEndEditEvent& event);
```

endFlashing — IFrameWindow

Description: Stops flashing the frame window.

Member:
```
virtual IFrameWindow& endFlashing ();
```

endUsingFont — IFont

Description: Restores the presentation space to the default font.

Member:
```
virtual IFont& endUsingFont
(const IPresSpaceHandle& presSpaceHandle);
```

enqueue — IRQueue — vbcc

Description: Adds the element to the collection, and sets the cursor to the added element.

Member:
```
void enqueue (Element const& element);
```

erf — mathSamples — vbsample

Description: erf calculates the error function.

Member:
```
double erf (double x);
```

exit — stdlibSamples — vbsample

Description: exit returns control to the host environment from the program.

Member:
```
void exit (int status);
```

exp — mathSamples — vbsample

Description: exp calculates the exponential function of a floating-point argument x (e to the exponent x, where e equals 2.

Member:
```
double exp (double x);
```

expand — IContainerControl

Description: Expands the tree view to show the descendents of the specified parent object.

Member:
```
virtual IContainerControl& expand (IContainerObject* object);
```

expand — IToolBar

Description: Displays the entire tool bar when it is floating.

Member:
```
virtual IToolBar& expand (Boolean expand=true);
```

expandBy — IRectangle

Description: Moves the corners of the rectangle outward from the center by the specified amount.

Member:
```
IRectangle& expandBy (Coord coord);
```

expandedBy — IRectangle

Description: Same as `IRectangle::expandBy`, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle expandedBy (const IPair& pair) const;
```

expandTree — IContainerControl

Description: Expands all nodes of a tree view.

Member:
```
virtual IContainerControl& expandTree ();
```

exportSelectedTextToFile — IMultiLineEdit

Description: Saves the currently marked text to the specified file and returns the number of bytes written to the file.

Member:
```
virtual unsigned long exportSelectedTextToFile
(const char* fileName, EOLFormat type=IMultiLineEdit::cfText);
```

exportToFile — IMultiLineEdit

Description: Saves the contents of the MLE to the specified file and returns the number of bytes written to the file.

Member:
```
virtual unsigned long exportToFile (const char* fileName,
EOLFormat type=IMultiLineEdit::cfText);
```

fabs — mathSamples — vbsample

Description: fabs calculates the absolute value of the floating-point argument x.

Member:
```
double fabs (double x);
```

fclose — stdioSamples — vbsample

Description: fclose closes a stream pointed to by stream.

Member:
```
int fclose (FILE* stream);
```

fdopen — stdioSamples — vbsample

Description: fdopen associates an input or output stream with the file identified by handle.

Member:
```
FILE fdopen (int handle, char* type);
```

feof — stdioSamples — vbsample

Description: feof indicates whether the end-of-file flag is set for the given stream.

Member:
```
int feof (FILE* stream);
```

ferror — stdioSamples — vbsample

Description: ferror tests for an error in reading from or writing to the given stream.

Member:
```
int ferror (FILE* stream);
```

fflush — stdioSamples — vbsample

Description: fflush causes the system to empty the buffer associated with the specified output stream, if possible.

Member:
```
int fflush (FILE* stream);
```

fgetpos — stdioSamples — vbsample

Description: fgetpos stores the current position of the file pointer associated with stream into the object pointed to by pos.

Member:
```
int fgetpos (FILE* stream, fpos_t* pos);
```

fgetws — stdioSamples — vbsample

Description: fgetws reads wide characters from the stream into the array pointed to by wcs.

Member:
```
wchar_t fgetws (wchar_t* wcs, int n, FILE* stream);
```

fileno — stdioSamples — vbsample

Description: fileno determines the file handle currently associated with stream.

Member:
```
int fileno (FILE* stream);
```

filter — IContainerControl

Description: Filters the container by calling the specified IContainerControl::FilterFn::isMemberOf function for each object in the container.

Member:
```
virtual IContainerControl& filter
(const IContainerControl::FilterFn& filterObject);
```

floor — mathSamples — vbsample

Description: floor calculates the largest integer that is less than or equal to x.

Member:
```
double floor (double x);
```

floorValue — IVBDoublePart — vbsample

Description: Perform floor function on value.

Member:
```
virtual IVBDoublePart& floorValue ();
```

flyHelpText — IFlyOverHelpHandler

Description: Returns the short help text for a window if you have dynamically added help text for the window.

Member:
```
virtual IString flyHelpText (const IWindowHandle& handle)
const;
```

fmod — mathSamples — vbsample

Description: fmod calculates the floating-point remainder of x/y.

Member:
```
double fmod (double x, double y);
```

fopen — stdioSamples — vbsample

Description: fopen opens the file specified by filename.

Member:
```
FILE fopen (const char* filename, const char* mode);
```

format — IClipboard

Description: Returns the requested clipboard format.

Member:
```
static IString format (const FormatHandle& handle);
```

formatAsHandle — IClipboard

Description: Returns a clipboard format as a handle.

Member:
```
static FormatHandle formatAsHandle (const char* format);
```

fputc — stdioSamples — vbsample

Description: fputc converts c to an `unsigned char` and then writes c to the output stream at the current position and advances the file position appropriately.

Member:
```
int fputc (int c, FILE* stream);
```

fputs — stdioSamples — vbsample

Description: fputs copies string to the output stream at the current position.

Member:
```
int fputs (const char* string, FILE* stream);
```

fputwc — stdioSamples — vbsample

Description: fputwc converts the wide character wc to a multibyte character and writes it to the output stream pointed to by stream at the current position.

Member:
```
wint_t fputwc (wint_t wc, FILE* stream);
```

fputws — stdioSamples — vbsample

Description: fputws converts the wide-character string wcs to a multibyte-character string and writes it to stream as a multibyte character string.

Member:
```
int fputws (const wchar_t* wcs, FILE* stream);
```

frameRectFor — IFrameWindow

Description: Returns a rectangle indicating the frame window size and position required to produce the specified client window.

Member:
```
virtual IRectangle frameRectFor (const IRectangle& clientRect)
const;
```

fread — stdioSamples — vbsample

Description: fread reads up to count items of size length from the input stream and stores them in the given buffer.

Member:
```
size_t fread (void* buffer, size_t size, size_t count,
FILE* stream);
```

free — stdlibSamples — vbsample

Description: free frees a block of storage.

Member:
```
void free (void* ptr);
```

freopen — stdioSamples — vbsample

Description: freopen closes the file currently associated with stream and reassigns stream to the file specified by filename.

Member:
```
FILE freopen (const char* filename, const char* mode,
FILE* stream);
```

frexp — mathSamples — vbsample

Description: fscanf reads data from the current position of the specified stream into the locations given by the entries in argument-list, if any.

Member:
```
double frexp (double x, int* expptr);
```

fseek — stdioSamples — vbsample

Description: fseek changes the current file position associated with stream to a new location within the file.

Member:
```
int fseek (FILE* stream, long int offset, int origin);
```

fsetpos — stdioSamples — vbsample

Description: fsetpos moves any file position associated with stream to a new location within the file according to the value pointed to by pos.

Member:
```
int fsetpos (FILE* stream, const fpos_t* pos);
```

ftell — stdioSamples — vbsample

Description: ftell finds the current position of the file associated with stream.

Member:
```
long int ftell (FILE* stream);
```

fwrite — stdioSamples — vbsample

Description: fwrite writes up to count items, each of size bytes in length, from buffer to the output stream.

Member:
```
size_t fwrite (const void* buffer, size_t size, size_t count,
FILE* stream);
```

gamma — mathSamples — vbsample

Description: gamma computes ln(|G(x)|).

Member:
```
double gamma (double x);
```

getc — stdioSamples — vbsample

Description: getc reads a single character from the current stream position and advances the stream position to the next character.

Member:
```
int getc (FILE* stream);
```

getenv — stdlibSamples — vbsample

Description: getenv searches the list of environment variables for an entry corresponding to varname.

Member:
```
char getenv (const char* varname);
```

gets — stdioSamples — vbsample

Description: gets reads a line from the standard input stream stdin and stores it in buffer.

Member:
```
char gets (char* buffer);
```

getwc — stdioSamples — vbsample

Description: getwc reads the next multibyte character from stream, converts it to a wide character, and advances the associated file position indicator for stream.

Member:
```
wint_t getwc (FILE* stream);
```

goToEntry — IMMAudioCD — vbmm

Description: Moves the current position to the passed in location.

Member:
```
virtual IMMAudioCD& goToEntry (unsigned long index);
```

handleCursoredChange — IContainerObject

Description: Called when cursored emphasis changes on an object and the `IContainerControl` containing the object has an ICnrHandler.

Member:
```
virtual void handleCursoredChange
(IContainerControl* container, Boolean acquired);
```

handleException — IWindow

Description: Called when the library detects an exception during the dispatch of an event.

Member:
```
virtual Boolean handleException
(IException& dispatcherException, IEvent& exceptionEvent);
```

handleFor — IFrameWindow

Description: Returns the IWindowHandle corresponding to a standard frame control.

Member:
```
virtual IWindowHandle handleFor (const Style& standardControl)
const;
```

handleInuseChange — IContainerObject

Description: Called when in-use emphasis changes on an object and the IContainerControl containing the object has an ICnrHandler.

Member:
```
virtual void handleInuseChange (IContainerControl* container,
Boolean acquired);
```

handleOpen — IContainerObject

Description: Called when the user either selects the Enter key when an object has the cursor or double-clicks the mouse select button on an object and the IContainerControl containing the object has an ICnrHandler.

Member:
```
virtual void handleOpen (IContainerControl* container);
```

handleSelectedChange — IContainerObject

Description: Called when selection emphasis changes on an object and the IContainerControl containing the object has an ICnrHandler.

Member:
```
virtual void handleSelectedChange
(IContainerControl* container, Boolean acquired);
```

handleTreeCollapse — IContainerObject

Description: Called when this object is a parent in the tree view and the descendent objects have been collapsed.

Member:
```
virtual void handleTreeCollapse
(IContainerControl* container);
```

handleTreeExpand — IContainerObject

Description: Called when this object is a parent in the tree view and the descendent objects have been expanded.

Member:
```
virtual void handleTreeExpand (IContainerControl* container);
```

hasData — IClipboard

Description: If the clipboard has data of the passed format, returns true.

Member:
```
virtual Boolean hasData (const char* format) const;
```

helpWindow — IHelpWindow

Description: Returns the help window that is associated with the specified application window.

Member:
```
static IHelpWindow* helpWindow (const IWindow* window);
```

hide — IContainerColumn

Description: Hides the column by making it invisible.

Member:
```
virtual IContainerColumn& hide ();
```

hide — IContainerObject

Description: Makes this object invisible by filtering it out.

Member:
```
virtual IContainerObject& hide
(IContainerControl* container=0);
```

hide — IWindow

Description: Hides the window.

Member:
```
virtual IWindow& hide ();
```

hideDetailsViewTitles — IContainerControl

Description: Removes column headings from details view.

Member:
```
virtual IContainerControl& hideDetailsViewTitles ();
```

hideList — IBaseComboBox

Description: Hides the list box.

Member:
```
virtual IBaseComboBox& hideList ();
```

hideObject — IContainerControl

Description: Makes a specified object invisible by filtering it out of the container.

Member:
```
virtual IContainerControl& hideObject
(IContainerObject* object);
```

hidePanelIds — IHelpWindow

Description: Removes the identifier for a help panel from its title bar text.

Member:
```
virtual IHelpWindow& hidePanelIds ();
```

hideSeparators — IContainerColumn

Description: Removes either.

Member:
```
virtual IContainerColumn& hideSeparators (const DataStyle&
separatorStyles=IContainerColumn::horizontalSeparator |
IContainerColumn::verticalSeparator);
```

hideSourceEmphasis — IWindow

Description: Called by a pop-up menu handler to notify the window to remove pop-up menu emphasis.

Member:
```
virtual IWindow& hideSourceEmphasis ();
```

hideSplitBar — IContainerControl

Description: Removes the split bar from the details view work area.

Member:
```
virtual IContainerControl& hideSplitBar ();
```

hideTitle — IContainerControl

Description: Removes the container title from the container.

Member:
```
virtual IContainerControl& hideTitle ();
```

hideTitleSeparator — IContainerControl

Description: Removes the title separator from the container window.

Member:
```
virtual IContainerControl& hideTitleSeparator ();
```

hideTreeLine — IContainerControl

Description: Does not show the lines connecting parents to children's records.

Member:
```
virtual IContainerControl& hideTreeLine ();
```

highlight — IButton

Description: Sets the button's highlight state.

Member:
```
virtual IButton& highlight (Boolean highlight=true);
```

hypot — mathSamples — vbsample

Description: hypot calculates the length of the hypotenuse of a right-angled triangle based on the lengths of two sides side1 and side2.

Member:
```
double hypot (double side1, double side2);
```

iconRectangle — IContainerControl

Description: Returns the rectangle bounding an object's icon.

Member:
```
IRectangle iconRectangle (const IContainerObject* object,
Boolean includeText=false) const;
```

immediateDescendentsOf — IContainerControl

Description: Returns the set of immediate descendents only of the specified object in the tree view.

Member:
```
ICnrObjectSet immediateDescendentsOf
(IContainerObject* parentObject) const;
```

importFromFile — IMultiLineEdit

Description: Inserts the contents of the specified file into the MLE at the current cursor position and returns the number of bytes imported.

Member:
```
virtual unsigned long importFromFile (const char* fileName,
EOLFormat type=IMultiLineEdit::cfText);
```

includes — IRange

Description: Returns `true` if the range contains the specified coordinate value.

Member:
```
Boolean includes (Coord aValue) const;
```

includes — IString

Description: If the receiver contains the specified search string, `true` is returned.

Member:
```
Boolean includes (const IString& aString) const;
```

indexOf — IString

Description: Returns the byte index of the first occurrence of the specified string within the receiver.

Member:
```
unsigned indexOf (const char* pString, unsigned startPos=1)
const;
```

indexOfAnyBut — IString

Description: Returns the index of the first character of the receiver that is not in the specified set of characters.

Member:
```
unsigned indexOfAnyBut (const IString& validChars,
unsigned startPos=1) const;
```

indexOfAnyOf — IString

Description: Returns the index of the first character of the receiver that is a character in the specified set of characters.

Member:
```
unsigned indexOfAnyOf (char searchChar, unsigned startPos=1)
const;
```

indexOfPhrase — IString

Description: Returns the position of the first occurrence of the specified phrase in the receiver.

Member:
```
unsigned indexOfPhrase (const IString& wordString,
unsigned startWord=1) const;
```

indexOfWord — IString

Description: Returns the index of the specified white-space-delimited word in the receiver.

Member:
```
unsigned indexOfWord (unsigned wordNumber) const;
```

insert — I0String

Description: Inserts the specified string at the specified location.

Member:
```
static I0String insert (const IString& aString,
const IString& anInsert, unsigned index=( unsigned ) UINT_MAX,
char padCharacter=' ');
```

insert — IPointArray

Description: Inserts a point before the index specified.

Member:
```
IPointArray& insert (unsigned long index,
const IPoint& point);
```

insert — IString

Description: Inserts the specified string after the specified location.

Member:
```
IString& insert (const IString& aString, unsigned index=0,
char padCharacter=' ');
```

integerWithKey — IProfile

Description: Reads the application data and returns it as a `long` integer.

Member:
```
virtual long integerWithKey (const char* key,
const char* applName=0) const;
```

intersectionWith — IRBag — vbcc

Description: Makes the collection the intersection of the collection and the given collection.

Member:
```
void intersectionWith (IABag<Element>const& bag);
```

intersectionWith — IRSet — vbcc

Description: Makes the collection the intersection of the collection and the given collection.

Member:
```
void intersectionWith (IASet<Element>const& collection);
```

intersectionWith — IRSortedBag — vbcc

Description: Makes the collection the intersection of the collection and the given collection.

Member:
```
void intersectionWith (IASortedBag<Element>const& bag);
```

intersectionWith — IRSortedSet — vbcc

Description: Makes the collection the intersection of the collection and the given collection.

Member:
```
void intersectionWith (IASortedSet<Element>const& collection);
```

intersects — IRectangle

Description: If the rectangle and specified rectangle overlap, `true` is returned.

Member:
```
Boolean intersects (const IRectangle& rectangle) const;
```

isAbbreviationFor — IString

Description: If the receiver is a valid abbreviation of the specified string, `true` is returned.

Member:
```
Boolean isAbbreviationFor (const IString& fullString,
unsigned minAbbrevLength=0) const;
```

isAnExtension — IFrameWindow

Description: Returns `true` if the specified `IWindow` object is a frame extension.

Member:
```
virtual Boolean isAnExtension (const IWindow* window) const;
```

isatty — ioSamples — vbsample

Description: isatty determines whether the given handle is associated with a character device (a keyboard, display, or printer or serial port).

Member:
```
int isatty (int handle);
```

isCollapsed — IContainerControl

Description: Queries whether the specified tree-view node object is currently collapsed.

Member:
```
Boolean isCollapsed (const IContainerObject* object) const;
```

isColumnExpandable — IMultiCellCanvas

Description: If the specified column can be expanded when the canvas is larger than its minimum size, `true` is returned.

Member:
```
Boolean isColumnExpandable (unsigned long column) const;
```

isColumnRight — IContainerControl

Description: If the specified column is to the right of the split bar, `true` is returned.

Member:
```
Boolean isColumnRight (const IContainerColumn* column) const;
```

isConnectionSupported — IMMDevice — vbmm

Description: Returns `true` if the connector is valid for this device.

Member:
```
Boolean isConnectionSupported (ConnectorType type,
CallType call=IMMDevice::wait) const;
```

isConnectorEnabled — IMMDevice — vbmm

Description: Returns `true` if the connector is enabled for this device.

Member:
```
Boolean isConnectorEnabled (ConnectorType type,
CallType call=IMMDevice::wait) const;
```

isCursored — IContainerControl

Description: Queries whether the specified object has cursored emphasis.

Member:
```
Boolean isCursored (const IContainerObject* object) const;
```

isDropOnAble — IContainerControl

Description: Queries whether the specified object is set to receive a direct manipulation drop.

Member:
```
Boolean isDropOnAble (const IContainerObject* object) const;
```

isEntryPoint32Bit — IDynamicLinkLibrary

Description: If the entry point is a 32-bit function, returns `true`.

Member:
```
Boolean isEntryPoint32Bit (unsigned long procedureOrdinal)
const;
```

isExpanded — IContainerControl

Description: Queries whether the specified tree-view node object is currently expanded.

Member:
```
Boolean isExpanded (const IContainerObject* object) const;
```

isInUse — IContainerControl

Description: Queries whether the specified object has in-use emphasis.

Member:
```
Boolean isInUse (const IContainerObject* object) const;
```

isLayoutDistorted — IWindow

Description: Determines whether changes have been made in a window that require updating the layout of the window in a canvas.

Member:
```
virtual Boolean isLayoutDistorted
(unsigned long layoutAttribute) const;
```

isLeapYear — IDate

Description: If the specified year is a leap year, `true` is returned.

Member:
```
static Boolean isLeapYear (int aYear);
```

isLike — IAcceleratorKey

Description: Identifies if two accelerator keys use the same keystroke.

Member:
```
Boolean isLike (const IAcceleratorKey& key) const;
```

isLike — IString

Description: If the receiver matches the specified pattern, which can contain wildcard characters, `true` is returned.

Member:
```
Boolean isLike (const IString& aPattern, char zeroOrMore='*',
char anyChar='?') const;
```

isMoveValid — IContainerControl

Description: Determines whether an object and its descendents can be moved to a new location.

Member:
```
virtual Boolean isMoveValid (IContainerObject* moveObject,
IContainerObject* newParentObject=0,
IContainerControl* newContainer=0,
IContainerObject* afterObject=0);
```

isRowExpandable — IMultiCellCanvas

Description: If the specified row can be expanded when the canvas is larger than its minimum size, `true` is returned.

Member:
```
Boolean isRowExpandable (unsigned long row) const;
```

isSelected — IBaseComboBox

Description: Returns `true` if the specified 0-based item is currently selected.

Member:
```
virtual Boolean isSelected (unsigned long index) const;
```

isSelected — IBaseListBox

Description: Returns the selection state of a 0-based item.

Member:
```
virtual Boolean isSelected (unsigned long index) const;
```

isSelected — IContainerControl

Description: Queries whether the specified object has selection emphasis.

Member:
```
Boolean isSelected (const IContainerObject* object) const;
```

isSource — IContainerControl

Description: Queries whether the specified object has source emphasis.

Member:
```
Boolean isSource (const IContainerObject* object) const;
```

isTarget — IContainerControl

Description: Queries whether the specified object is a target of direct manipulation.

Member:
```
Boolean isTarget (const IContainerObject* object) const;
```

isValid — IDate

Description: Indicates whether the specified date is valid.

Member:
```
static Boolean isValid (int aDay, Month aMonth, int aYear);
```

isWriteable — IContainerControl

Description: Returns `true` if the object can be edited.

Member:
```
Boolean isWriteable (const IContainerObject* object) const;
```

itemHandle — IBaseComboBox

Description: Returns the handle of the specified list box 0-based item.

Member:
```
virtual unsigned long itemHandle (unsigned long index) const;
```

itemHandle — IBaseListBox

Description: Returns the handle of the specified list box 0-based item.

Member:
```
virtual unsigned long itemHandle (unsigned long index) const;
```

itemText — IBaseComboBox

Description: Returns the text of the specified 0-based item in the list box portion of the combination box.

Member:
```
virtual IString itemText (unsigned long index) const;
```

itemText — IBaseListBox

Description: Returns the text of the specified 0-based item in the list box.

Member:
```
virtual IString itemText (unsigned long index) const;
```

justifyData — IContainerColumn

Description: Sets the justification of the data to a horizontal and vertical alignment.

Member:
```
virtual IContainerColumn& justifyData (VerticalAlignment
verticalAlignment=IContainerColumn::centeredVertically,
HorizontalAlignment horizontalAlignment=IContainerColumn::centered);
```

justifyHeading — IContainerColumn

Description: Sets the justification of the heading to a horizontal and vertical alignment.

Member:
```
virtual IContainerColumn& justifyHeading (VerticalAlignment
verticalAlignment=IContainerColumn::centeredVertically,
HorizontalAlignment horizontalAlignment=IContainerColumn::centered);
```

labs — stdlibSamples — vbsample

Description: labs produces the absolute value of its long integer argument n.

Member:
```
long int labs (long int n);
```

lastIndexOf — IString

Description: Returns the index of the last occurrence of the specified string or character.

Member:
```
unsigned lastIndexOf (const IString& aString,
unsigned startPos=(unsigned) UINT_MAX) const;
```

lastIndexOfAnyBut — IString

Description: Returns the index of the last character not in the specified string or character.

Member:
```
unsigned lastIndexOfAnyBut (const IString& validChars,
unsigned startPos=(unsigned) UINT_MAX) const;
```

lastIndexOfAnyOf — IString

Description: Returns the index of the last character in the specified string or character.

Member:
```
unsigned lastIndexOfAnyOf (const IString& searchChars,
unsigned startPos=(unsigned) UINT_MAX) const;
```

ldexp — mathSamples — vbsample

Description: ldexp calculates the value of x * (2**exp).

Member:
```
double ldexp (double x, int exp);
```

ldiv — stdlibSamples — vbsample

Description: ldiv calculates the quotient and remainder of the division of numerator by denominator.

Member:
```
ldiv_t ldiv (long int numerator, long int denominator);
```

leftJustify — IString

Description: Left-justifies the receiver in a string of the specified length.

Member:
```
IString& leftJustify (unsigned length, char padCharacter=' ');
```

lengthOfWord — IString

Description: Returns the length of the specified white-space-delimited word in the receiver.

Member:
```
unsigned lengthOfWord (unsigned wordNumber) const;
```

lineFrom — IString

Description: Returns the next line from the specified input stream.

Member:
```
static IString lineFrom (istream& aStream, char delim='\n');
```

loadAccelTable — IResourceLibrary

Description: Loads an accelerator table from a resource file.

Member:
```
virtual IAccelTblHandle loadAccelTable
(unsigned long accelTableId) const;
```

loadBitmap — IResourceLibrary

Description: Loads a bitmap from a resource file.

Member:
```
virtual IBitmapHandle loadBitmap (unsigned long bitmapId,
const ISize& bitmapSize, Boolean cached=true) const;
```

loadDialog — IResourceLibrary

Description: Loads a dialog resource, creates a frame window, and sets the dialog as the client of the frame window.

Member:
```
virtual IWindowHandle loadDialog (unsigned long dialogId,
IWindow* dialogParent, IWindow* dialogOwner,
IWinProc* dialogProcedure, void* dialogCreateParameters)
const;
```

loadHelpTable — IResourceLibrary

Description: Loads a help table from a resource file.

Member:
```
virtual IResourceLibrary& loadHelpTable
(IWindow* helpInstance, unsigned long helpTableId) const;
```

loadIcon — IResourceLibrary

Description: Loads an icon from a resource file.

Member:
```
virtual IPointerHandle loadIcon (unsigned long iconId,
Boolean cached=true) const;
```

loadMenu — IResourceLibrary

Description: Loads a menu resource from a resource file.

Member:
```
virtual IWindowHandle loadMenu (unsigned long menuId,
IWindow* menuOwner) const;
```

loadMessage — IResourceLibrary

Description: Loads a message resource from a resource file using the specified message identifer.

Member:
```
virtual IString loadMessage (unsigned long messageId) const;
```

loadOnThread — IMMFileMedia — vbmm

Description: Loads the given file into memory, by creating a thread to do the actual loading.

Member:
```
virtual IMMFileMedia& loadOnThread (const IString& filename,
Boolean readOnly=false);
```

loadPointer — IResourceLibrary

Description: Loads a pointer from a resource file.

Member:
```
virtual IPointerHandle loadPointer (unsigned long iconId,
Boolean cached=true) const;
```

loadString — IResourceLibrary

Description: Loads a string resource from a resource file using the specified string identifer.

Member:
```
virtual IString loadString (unsigned long stringId) const;
```

locateOrAdd — IRBag — vbcc

Description: Locates an element in the collection that is equal to the given element.

Member:
```
IBoolean locateOrAdd (Element const& element);
```

locateOrAdd — IREqualitySequence — vbcc

Description: Locates an element in the collection that is equal to the given element.

Member:
```
IBoolean locateOrAdd (Element const& element);
```

locateOrAdd — IRSet — vbcc

Description: Locates an element in the collection that is equal to the given element.

Member:
```
IBoolean locateOrAdd (Element const& element);
```

locateOrAdd — IRSortedBag — vbcc

Description: Locates an element in the collection that is equal to the given element.

Member:
```
IBoolean locateOrAdd (Element const& element);
```

locateOrAdd — IRSortedSet — vbcc

Description: Locates an element in the collection that is equal to the given element.

Member:
```
IBoolean locateOrAdd (Element const& element);
```

locateText — IBaseComboBox

Description: Returns the item number of the list box item matching the search string.

Member:
```
virtual unsigned long locateText (const char* searchString,
Boolean caseSensitive=true,
SearchType search=IBaseComboBox::exactMatch,
unsigned long index=IBaseComboBox::first) const;
```

locateText — IBaseListBox

Description: Returns the item number of the list box item matching the search string.

Member:
```
virtual unsigned long locateText (const char* searchString,
Boolean caseSensitive=true,
SearchType search=IBaseListBox::exactMatch,
unsigned long index=IBaseListBox::first) const;
```

log — mathSamples — vbsample

Description: log calculates the natural logarithm (base e) of x.

Member:
```
double log (double x);
```

log10 — mathSamples — vbsample

Description: log10 calculates the base 10 logarithm of x.

Member:
```
double log10 (double x);
```

logicalAndValue — IVBBooleanPart — vbsample

Description: Perform logical and operator on value.

Member:
```
virtual IVBBooleanPart& logicalAndValue (Boolean andValue);
```

logicalNotValue — IVBBooleanPart — vbsample

Description: Perform logical negation on value.
Member:
```
virtual IVBBooleanPart& logicalNotValue ();
```

logicalOrValue — IVBBooleanPart — vbsample

Description: Perform logical or operator on value.
Member:
```
virtual IVBBooleanPart& logicalOrValue (Boolean orValue);
```

longHelpText — IFlyOverHelpHandler

Description: Returns the long help text for a window if you have dynamically added long help text for the window.
Member:
```
virtual IString longHelpText (const IWindowHandle& handle)
const;
```

lowerCase — IString

Description: Translates all upper-case letters in the receiver to lower-case.
Member:
```
static IString lowerCase (const IString& aString);
```

lseek — ioSamples — vbsample

Description: lseek moves any file pointer associated with handle to a new location that is offset bytes from the origin.
Member:
```
long lseek (int handle, long offset, int origin);
```

malloc — stdlibSamples — vbsample

Description: malloc reserves a block of storage of size bytes.
Member:
```
void malloc (size_t size);
```

matchForMnemonic — IWindow

Description: Returns the first child window using the specified character as a mnemonic.
Member:
```
virtual IWindowHandle matchForMnemonic
(unsigned short character) const;
```

maximize — IFrameWindow

Description: If the window has a maximize button, this function maximizes the window.
Member:
```
virtual IFrameWindow& maximize ();
```

maximum — IPair

Description: Returns an ordered pair whose coordinates are the maximum of the corresponding coordinates of the IPair and the specified IPair.

Member:
```
IPair maximum (const IPair& aPair) const;
```

mblen — stdlibSamples — vbsample

Description: mblen determines the length in bytes of the multibyte character pointed to by string.

Member:
```
int mblen (const char* string, size_t n);
```

mbstowcs — stdlibSamples — vbsample

Description: mbstowcs converts the multibyte character string pointed to by string into the wide-character array pointed to by dest.

Member:
```
size_t mbstowcs (wchar_t* dest, const char* string,
size_t len);
```

mbtowc — stdlibSamples — vbsample

Description: mbtowc first determines the length of the multibyte character pointed to by string.

Member:
```
int mbtowc (wchar_t* pwc, const char* string, size_t n);
```

menuSelected — IVBCheckMenuHandler

Member:
```
virtual Boolean menuSelected (IMenuEvent& menuEvent);
```

minimize — IFrameWindow

Description: Minimizes the window.

Member:
```
virtual IFrameWindow& minimize ();
```

minimum — IPair

Description: Returns an ordered pair whose coordinates are the minimum of the corresponding coordinates of the IPair and the specified IPair.

Member:
```
IPair minimum (const IPair& aPair) const;
```

minTextWidth — IFont

Description: Returns the width of the widest word.

Member:
```
unsigned long minTextWidth (const char* line) const;
```

modf — mathSamples — vbsample

Description: modf breaks down the floating-point value x into fractional and integral parts.

Member:
```
double modf (double x, double* intptr);
```

monthName — IDate

Description: Returns the name of the receiver's month.

Member:
```
static IString monthName (Month aMonth);
```

moveAfter — IToolBar

Description: Moves the window so that it immediately follows the reference window.

Member:
```
virtual IToolBar& moveAfter (IWindow* window,
IWindow* referenceWindow, Boolean startNewGroup=false);
```

moveBefore — IToolBar

Description: Moves the window so that it immediately precedes the reference window.

Member:
```
virtual IToolBar& moveBefore (IWindow* window,
IWindow* referenceWindow, Boolean startNewGroup=false);
```

moveBy — IRectangle

Description: Moves the rectangle by the amount specified by aPair.

Member:
```
IRectangle& moveBy (const IPair& pair);
```

movedBy — IRectangle

Description: Same as `IRectangle::moveBy`, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle movedBy (const IPair& pair) const;
```

movedTo — IRectangle

Description: Same as `IRectangle::moveTo`, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle movedTo (const IPoint& point) const;
```

moveIconTo — IContainerControl

Description: Moves the specified object to a new workspace location within the icon view.

Member:
```
virtual IContainerControl& moveIconTo
(IContainerObject* object, const IPoint& point);
```

moveObjectTo — IContainerControl

Description: Moves the specified object and its descendents to the specified new location.

Member:
```
virtual Boolean moveObjectTo (IContainerObject* moveObject,
IContainerObject* newParentObject=0,
IContainerControl* newContainer=0,
IContainerObject* afterObject=0,
const IPoint& iconViewLocation=IPoint ( 0 , 0 ));
```

moveSizeToClient — IFrameWindow

Description: Sizes and positions the frame window around the specified client window rectangle.

Member:
```
virtual IFrameWindow& moveSizeToClient
(const IRectangle& clientRect);
```

moveToFirst — IToolBar

Description: Moves the window so that it is the first window in the tool bar.

Member:
```
virtual IToolBar& moveToFirst (IWindow* window,
Boolean startNewGroup=false);
```

moveToLast — IToolBar

Description: Moves the window so that it is the last window in the tool bar.

Member:
```
virtual IToolBar& moveToLast (IWindow* window,
Boolean startNewGroup=false);
```

multiLineEdit — ICnrEditHandler

Description: Called to return an `IMultiLineEdit` wrapper.

Member:
```
virtual IMultiLineEdit* multiLineEdit
(const IWindowHandle& handleMultiLineEdit);
```

multiplyValue — IVBDoublePart — vbsample

Description: Perform multiply operator on value.

Member:
```
virtual IVBDoublePart& multiplyValue (double multiplyValue=2);
```

multiplyValue — IVBLongPart — vbsample

Description: Perform multiply operator on value.

Member:
```
virtual IVBLongPart& multiplyValue (long multiplyValue=2);
```

multiplyValue — IVBShortPart — vbsample

Description: Perform multiply operator on value.

Member:
```
virtual IVBShortPart& multiplyValue (short multiplyValue=2);
```

multiplyValue — IVBUnsignedLongPart — vbsample

Description: Perform multiply operator on value.

Member:
```
virtual IVBUnsignedLongPart& multiplyValue
(unsigned long multiplyValue=2);
```

multiplyValue — IVBUnsignedShortPart — vbsample

Description: Perform multiply operator on value.

Member:
```
virtual IVBUnsignedShortPart& multiplyValue
(unsigned short multiplyValue=2);
```

nextPage — INotebook

Description: Returns an IPageHandle object that references the page following the specified page.

Member:
```
virtual IPageHandle nextPage
(const IPageHandle& referencePage) const;
```

nlsCompare — IContainerControl

Description: Compares two specified strings consisting of a national character set.

Member:
```
static long nlsCompare (const char* text1, const char* text2);
```

notebookSize — INotebook

Description: Returns the size that the notebook must be in order to contain the specified page.

Member:
```
virtual ISize notebookSize (const IPageHandle& page) const;
```

notifyObservers — IStandardNotifier

Description: Notifies all observers in an object's observer list.

Member:
```
virtual IStandardNotifier& notifyObservers
(const INotificationEvent& anEvent);
```

notifyObservers — IWindow

Description: Notifies all observers in part's collection.

Member:
```
virtual IWindow& notifyObservers
(const INotificationEvent& event);
```

notifyOwner — IFrameWindow

Description: Posts a command event to the frame window's (or dialog's) owner.

Member:
```
virtual IFrameWindow& notifyOwner (unsigned long id,
ICommandEvent::Source source=ICommandEvent::unknown,
Boolean pointerDevice=false);
```

numberOfOccurrences — IRBag — vbcc

Description: Returns the number of occurrences of the given element in the collection.

Member:
```
INumber numberOfOccurrences (Element const& element) const;
```

numberOfOccurrences — IREqualitySequence — vbcc

Description: Returns the number of occurrences of the given element in the collection.

Member:
```
INumber numberOfOccurrences (Element const& element) const;
```

numberOfOccurrences — IRSortedBag — vbcc

Description: Returns the number of occurrences of the given element in the collection.

Member:
```
INumber numberOfOccurrences (Element const& element) const;
```

numberOfTicks — IProgressIndicator

Description: Returns the number of ticks for the specified scale.

Member:
```
unsigned long numberOfTicks (Scale scale) const;
```

objectAt — IContainerControl

Description: Returns an object at the specified index or cursor position in the container.

Member:
```
virtual IContainerObject* objectAt (unsigned long index)
const;
```

objectUnderPoint — IContainerControl

Description: Retrieves the object under the specified point, in window coordinates.

Member:
```
IContainerObject* objectUnderPoint (const IPoint& point)
const;
```

objectWindow — staticWindowSamples — vbsample

Description: Static function to return object window.

Member:
```
static IWindow* IWindow::objectWindow ();
```

occurrencesOf — IString

Description: Returns the number of occurrences of the specified IString, char*, char, or IStringTest.

Member:
```
unsigned occurrencesOf (const IString& aString,
unsigned startPos=1) const;
```

open — IClipboard

Description: Opens the clipboard.

Member:
```
virtual IClipboard& open ();
```

open — IDynamicLinkLibrary

Description: Opens a dynamic link library and increments the reference count.

Member:
```
virtual IDynamicLinkLibrary& open ();
```

open — IMMDevice — vbmm

Description: Opens the device or file based on the passed in string.

Member:
```
virtual IMMDevice& open (const IString& fileOrDevice=IString (
), Boolean shareable=true, CallType call=IMMDevice::wait);
```

open — ioSamples — vbsample

Description: open opens the file specified by pathname and prepares the file for subsequent reading or writing as defined by oflag.

Member:
```
int open (char* pathname, int oflag, int pmode);
```

openDoor — IMMRemovableMedia — vbmm

Description: Opens the door and ejects the tray, if possible.

Member:
```
virtual IMMRemovableMedia& openDoor (Boolean open=true,
CallType call=IMMDevice::wait);
```

openOnThread — IMMDevice — vbmm

Description: Opens the file or device by creating a thread to do the actual opening.

Member:
```
virtual IMMDevice& openOnThread
(const IString& fileOrDevice=IString ( ),
Boolean shareable=true, CallType call=IMMDevice::wait);
```

operator != — IABag — vbcc

Description: Operator not equal.

Member:
```
virtual IBoolean operator!= (IABag<Element>const& bag) const;
```

operator != — IAcceleratorKey

Description: Specifies if two accelerator keys use a different keystroke or run a different action.

Member:
```
Boolean operator!= (const IAcceleratorKey& key) const;
```

operator != — IAddress — vbsample

Member:
```
Boolean operator!= (const IAddress* aValue) const;
```

operator != — IAEqualitySequence — vbcc

Description: Operator not equal.

Member:
```
virtual IBoolean operator!=
(IAEqualitySequence<Element>const& collection) const;
```

operator != — IASet — vbcc

Description: Operator not equal.

Member:
```
virtual IBoolean operator!= (IASet<Element>const& collection)
const;
```

operator != — IASortedBag — vbcc

Description: Operator not equal.

Member:
```
virtual IBoolean operator!= (IASortedBag<Element>const& bag)
const;
```

operator != — IASortedSet — vbcc

Description: Operator not equal.

Member:
```
virtual IBoolean operator!=
(IASortedSet<Element>const& collection) const;
```

operator != — IColor

Description: Returns `true` if the colors are not identical.

Member:
```
Boolean operator!= (const IColor& color) const;
```

operator != — ICompany — vbsample

Member:
```
Boolean operator!= (const ICompany& aValue) const;
```

operator != — ICustomer — vbsample

Member:
```
Boolean operator!= (const ICustomer* aValue) const;
```

operator != — IDate

Description: If the `IDate` objects represent different dates, `true` is returned.

Member:
```
Boolean operator!= (const IDate& aDate) const;
```

operator != — IMMTime — vbmm

Description: Returns `true` if this time is not equal to the passed in time.

Member:
```
Boolean operator!= (const IMMTime& time) const;
```

operator != — IOrderedRecord — vbsample

Description: `True` if and only if the records are not identical.

Member:
```
virtual IBoolean operator!= (const IOrderedRecord& aRecord)
const;
```

operator != — IPair

Description: `True` if either coordinate differs.

Member:
```
Boolean operator!= (const IPair& aPair) const;
```

operator != — IPointArray

Description: Returns `true` if the arrays are not the same length or the points are not identical or both.

Member:
```
operator!= (const IPointArray& pointArray) const;
```

operator != — IRectangle

Description: If the rectangles differ, `true` is returned.

Member:
```
Boolean operator!= (const IRectangle& rectangle) const;
```

operator != — ITime

Description: Compares two objects to determine whether they are not equal.

Member:
```
Boolean operator!= (const ITime& aTime) const;
```

operator != — IVBBooleanPart — vbsample

Member:
```
Boolean operator!= (const IVBBooleanPart& aValue) const;
```

operator != — IVBDoublePart — vbsample

Member:
```
Boolean operator!= (const IVBDoublePart& aValue) const;
```

operator != — IVBLogicalAndPart — vbsample

Member:
```
Boolean operator!= (const IVBLogicalAndPart* aValue) const;
```

operator != — IVBLogicalOrPart — vbsample

Member:
```
Boolean operator!= (const IVBLogicalOrPart* aValue) const;
```

operator != — IVBLongPart — vbsample

Member: `Boolean operator!= (const IVBLongPart& aValue) const;`

operator != — IVBShortPart — vbsample

Member: `Boolean operator!= (const IVBShortPart& aValue) const;`

operator != — IVBStringPart — vbsample

Member: `Boolean operator!= (const IVBStringPart* aValue) const;`

operator != — IVBUnsignedLongPart — vbsample

Member: `Boolean operator!= (const IVBUnsignedLongPart& aValue) const;`

operator != — IVBUnsignedShortPart — vbsample

Member: `Boolean operator!= (const IVBUnsignedShortPart& aValue) const;`

operator %= — IPair

Description: Replaces the coordinates with the remainder when divided by those of the following specified parameter.

Member: `IPair& operator%= (const IPair& aPair);`

operator & — IRectangle

Description: Returns a rectangle representing the intersection of the specified rectangles.

Member: `IRectangle operator& (const IRectangle& rectangle) const;`

operator & — IString

Description: Performs bitwise AND.

Member: `IString operator& (const IString& aString) const;`

operator &= — IRectangle

Description: Resets the rectangle to its intersection with the specified rectangle.

Member: `IRectangle& operator&= (const IRectangle& rectangle);`

operator &= — IString

Description: Performs bitwise AND and replaces the receiver.

Member: `IString& operator&= (const IString& aString);`

operator * — IMngPointer — vbcc

Description: Return the value of the managed pointer.

Member:
```
Element& operator* () const;
```

operator *= — IPair

Description: Multiplies the coordinates by those of the specified parameter.

Member:
```
IPair& operator*= (double multiplier);
```

operator + — IDate

Description: Adds an integral number of days to the left-hand operand, yielding a new IDate.

Member:
```
IDate operator+ (int numDays) const;
```

operator + — IMMTime — vbmm

Description: Returns the sum of this time and the passed in time.

Member:
```
IMMTime operator+ (const IMMTime& time) const;
```

operator + — IString

Description: Concatenates two strings.

Member:
```
IString operator+ (const IString& aString) const;
```

operator + — ITime

Description: Adds two objects.

Member:
```
ITime operator+ (const ITime& aTime) const;
```

operator += — IDate

Description: Adds an integral number of days to the left-hand operand, assigning the result to that operand.

Member:
```
IDate& operator+= (int numDays);
```

operator += — IMMTime — vbmm

Description: Adds the passed in time from this time object

Member:
```
IMMTime& operator+= (const IMMTime& time);
```

operator += — IPair

Description: Adds the coordinates of the specified aPair to the coordinates of an ordered pair.

Member:
```
IPair& operator+= (const IPair& aPair);
```

operator += — IString

Description: Concatenates the specified string to the receiver and replaces the receiver.

Member:
```
IString& operator+= (const char* pString);
```

operator += — ITime

Description: Adds two objects and stores the result in the receiver.

Member:
```
ITime& operator+= (const ITime& aTime);
```

operator - — IDate

Description: Subtracts an integral number of days from the left-hand operand, yielding a new IDate.

Member:
```
long operator- (const IDate& aDate) const;
```

operator - — IMMTime — vbmm

Description: Returns the result of this time minus the passed in time.

Member:
```
IMMTime operator- (const IMMTime& time) const;
```

operator - — IPair

Description: Returns an ordered pair whose coordinates are the difference between the corresponding coordinates of pair1 and pair2.

Member:
```
IPair operator- () const;
```

operator - — ITime

Description: Subtracts one object from another.

Member:
```
ITime operator- (const ITime& aTime) const;
```

operator -= — IDate

Description: Subtracts an integral number of days from the right-hand operand, assigning the result to that operand.

Member:
```
IDate& operator-= (int numDays);
```

operator -= — IMMTime — vbmm

Description: Subtracts the passed in time from this time object.

Member:
```
IMMTime& operator-= (const IMMTime& time);
```

operator -= — IPair

Description: Subtracts the coordinates specified in aPair from the IPair coordinates.

Member:
```
IPair& operator-= (const IPair& aPair);
```

operator

operator -= — ITime

Description: Subtracts one object from another and stores the result in the receiver.

Member:
```
ITime& operator-= (const ITime& aTime);
```

operator /= — IPair

Description: Divides the coordinates by those of the second specified parameter.

Member:
```
IPair& operator/= (double divisor);
```

operator < — IDate

Description: If the left-hand operand represents a date prior to the date represented by the right-hand operand, `true` is returned.

Member:
```
Boolean operator< (const IDate& aDate) const;
```

operator < — IMMTime — vbmm

Description: Returns `true` if this time is less than the passed in time.

Member:
```
Boolean operator< (const IMMTime& time) const;
```

operator < — IOrderedRecord — vbsample

Description: `True` if and only if the first record is less than the second, when compared as IStrings.

Member:
```
virtual IBoolean operator< (const IOrderedRecord& aRecord)
const;
```

operator < — IPair

Description: `True` if both coordinates are less than those of the specified aPair.

Member:
```
Boolean operator< (const IPair& aPair) const;
```

operator < — ITime

Description: Compares two objects to determine whether one is less than the other.

Member:
```
Boolean operator< (const ITime& aTime) const;
```

operator <= — IDate

Description: If the left-hand operand represents a date prior to or identical to the date represented by the right-hand operand, `true` is returned.

Member:
```
Boolean operator<= (const IDate& aDate) const;
```

operator <= — IMMTime — vbmm

Description: Returns `true` if this time is less than or equal to the passed in time.

Member:
```
Boolean operator<= (const IMMTime& time) const;
```

operator <= — IOrderedRecord — vbsample

Description: Equivalent to '(record1 < record2) || (record1 == record2)'.

Member:
```
virtual IBoolean operator<= (const IOrderedRecord& aRecord)
const;
```

operator <= — IPair

Description: `True` if both coordinates are less than or equal.

Member:
```
Boolean operator<= (const IPair& aPair) const;
```

operator <= — ITime

Description: Compares two objects to determine whether one is less than or equal to the other.

Member:
```
Boolean operator<= (const ITime& aTime) const;
```

operator = — IAcceleratorKey

Description: Use this assignment operator to set the values used by an existing accelerator key object to those used by another `IAcceleratorKey` object.

Member:
```
IAcceleratorKey& operator= (const IAcceleratorKey& key);
```

operator = — IAcceleratorTable

Description: Use this assignment operator to initialize an accelerator table with the same entries as the specified one.

Member:
```
IAcceleratorTable& operator=
(const IAcceleratorTable& accelTbl);
```

operator = — IColor

Description: Replaces the color.

Member:
```
IColor& operator= (const IColor& color);
```

operator = — IFont

Description: Assigns the value of one font object to another.

Member:
```
IFont& operator= (const IFont& rhs);
```

operator

operator = — IMMAudioBuffer — vbmm

Description: Copies the audio buffer information from the passed in audio buffer.

Member:
```
IMMAudioBuffer& operator= (const IMMAudioBuffer& audioBuffer);
```

operator = — IMMAudioCDContents — vbmm

Description: Sets the contents to be the same as another table of contents.

Member:
```
IMMAudioCDContents& operator=
(const IMMAudioCDContents& newContents);
```

operator = — IMMTime — vbmm

Description: Sets the time to be equal to the passed in time.

Member:
```
IMMTime& operator= (const IMMTime& time);
```

operator = — IMngPointer — vbcc

Description: Replaces the contents of the managed pointer.

Member:
```
IMngPointer<Element>& operator=
(IMngPointer<Element>const& ptr);
```

operator = — IPointArray

Member:
```
IPointArray& operator= (const IPointArray& pointArray);
```

operator = — IProfile

Description: Assigns the member data of an object of this class to another object of this class.

Member:
```
IProfile& operator= (const IProfile& aProfile);
```

operator = — IRecord — vbsample

Description: Replaces the contents of the record.

Member:
```
IRecord& operator= (const IString& aRecord);
```

operator = — IResourceLibrary

Description: Use this operator to assign the member data of an object of this class to another object of this class.

Member:
```
IResourceLibrary& operator=
(const IResourceLibrary& resLibrary);
```

operator = — IStandardNotifier

Description: Assigns the contents of one notifier object to another.

Member:
```
IStandardNotifier& operator=
(const IStandardNotifier& aStandardNotifier);
```

operator = — IString

Description: Replaces the contents of the string.

Member:
```
IString& operator= (const IString& aString);
```

operator = — IVBagOnBase — vbcc

Description: Assignment operator.

Member:
```
IVBagOnBase<Element,Base>& operator=
(IVBagOnBase<Element,Base>const& c);
```

operator = — IVDequeOnBase — vbcc

Description: Assignment operator.

Member:
```
IVDequeOnBase<Element,Base>& operator=
(IVDequeOnBase<Element,Base>const& c);
```

operator = — IVEqualitySequenceOnBase — vbcc

Description: Assignment operator.

Member:
```
IVEqualitySequenceOnBase<Element,Base>& operator=
(IVEqualitySequenceOnBase<Element,Base>const& c);
```

operator = — IVHeapOnBase — vbcc

Description: Assignment operator.

Member:
```
IVHeapOnBase<Element,Base>& operator=
(IVHeapOnBase<Element,Base>const& c);
```

operator = — IVQueueOnBase — vbcc

Description: Assignment operator.

Member:
```
IVQueueOnBase<Element,Base>& operator=
(IVQueueOnBase<Element,Base>const& c);
```

operator = — IVSequenceOnBase

Description: Assignment operator,

Member:
```
IVSequenceOnBase<Element,Base>& operator=
(IVSequenceOnBase<Element,Base>const& c);
```

operator = — IVSetOnBase — vbcc

Description: Assignment operator.

Member:
```
IVSetOnBase<Element,Base>& operator=
(IVSetOnBase<Element,Base>const& c);
```

operator = — IVSortedBagOnBase — vbcc

Description: Assignment operator.

Member:
```
IVSortedBagOnBase<Element,Base>& operator=
(IVSortedBagOnBase<Element,Base>const& c);
```

operator = — IVSortedSetOnBase — vbcc

Description: Assignment operator.

Member:
```
IVSortedSetOnBase<Element,Base>& operator=
(IVSortedSetOnBase<Element,Base>const& c);
```

operator = — IVStackOnBase — vbcc

Description: Assignment operator.

Member:
```
IVStackOnBase<Element,Base>& operator=
(IVStackOnBase<Element,Base>const& c);
```

operator == — IABag — vbcc

Description: Operator compare equal.

Member:
```
virtual IBoolean operator== (IABag<Element>const& bag) const;
```

operator == — IAcceleratorKey

Description: Specifies if two accelerator keys use the same keystroke and run the same action.

Member:
```
Boolean operator== (const IAcceleratorKey& key) const;
```

operator == — IAddress — vbsample

Member:
```
Boolean operator== (const IAddress* aValue) const;
```

operator == — IAEqualitySequence — vbcc

Description: Operator compare equal.

Member:
```
virtual IBoolean operator==
(IAEqualitySequence<Element>const& collection) const;
```

operator == — IASet — vbcc

Description: Operator compare equal.

Member:
```
virtual IBoolean operator== (IASet<Element>const& collection)
const;
```

operator == — IASortedBag — vbcc

Description: Operator compare equal.

Member:
```
virtual IBoolean operator== (IASortedBag<Element>const& bag)
const;
```

operator == — IASortedSet — vbcc

Description: Operator compare equal.

Member:
```
virtual IBoolean operator==
(IASortedSet<Element>const& collection) const;
```

operator == — IColor

Description: Returns `true` if the colors are identical.

Member:
```
Boolean operator== (const IColor& color) const;
```

operator == — ICompany — vbsample

Member:
```
Boolean operator== (const ICompany& aValue) const;
```

operator == — IContainerControl

Description: If two containers are the same (that is, their storage location is identical), `true` is returned.

Member:
```
Boolean operator== (const IContainerControl& container);
```

operator == — IContainerObject

Description: If the objects are at the same storage location, `true` is returned.

Member:
```
Boolean operator== (const IContainerObject& object);
```

operator == — ICustomer — vbsample

Member:
```
Boolean operator== (const ICustomer& aValue) const;
```

operator == — IDate

Description: If the `IDate` objects represent the same date, `true` is returned.

Member:
```
Boolean operator== (const IDate& aDate) const;
```

operator == — IMMTime — vbmm

Description: Returns `true` if this time is equal to the passed in time.

Member:
```
Boolean operator== (const IMMTime& time) const;
```

operator == — IOrderedRecord — vbsample

Description: `True` if and only if the records are identical.

Member:
```
virtual IBoolean operator== (const IOrderedRecord& aRecord)
const;
```

operator

operator == — IPair

Description: True if both coordinates match those of the specified aPair.

Member:
```
Boolean operator== (const IPair& aPair) const;
```

operator == — IPointArray

Description: Returns true if the arrays are the same length and have identical points.

Member:
```
Boolean operator== (const IPointArray& pointArray) const;
```

operator == — IRectangle

Description: If the two rectangles are identical, true is returned.

Member:
```
Boolean operator== (const IRectangle& rectangle) const;
```

operator == — ITime

Description: Compares two objects to determine whether they are equal.

Member:
```
Boolean operator== (const ITime& aTime) const;
```

operator == — IVBBooleanPart — vbsample

Member:
```
Boolean operator== (const IVBBooleanPart* aValue) const;
```

operator == — IVBDoublePart — vbsample

Member:
```
Boolean operator== (const IVBDoublePart& aValue) const;
```

operator == — IVBLogicalAndPart — vbsample

Member:
```
Boolean operator== (const IVBLogicalAndPart& aValue) const;
```

operator == — IVBLogicalOrPart — vbsample

Member:
```
Boolean operator== (const IVBLogicalOrPart& aValue) const;
```

operator == — IVBLongPart — vbsample

Member:
```
Boolean operator== (const IVBLongPart* aValue) const;
```

operator == — IVBShortPart — vbsample

Member:
```
Boolean operator== (const IVBShortPart* aValue) const;
```

operator == — IVBStringPart — vbsample

Member:
```
Boolean operator== (const IVBStringPart* aValue) const;
```

operator == — IVBUnsignedLongPart — vbsample

Member:
```
Boolean operator== (const IVBUnsignedLongPart& aValue) const;
```

operator == — IVBUnsignedShortPart — vbsample

Member: `Boolean operator== (const IVBUnsignedShortPart& aValue) const;`

operator > — IDate

Description: If the left-hand operand represents a date subsequent to the date represented by the right-hand operand, `true` is returned.

Member: `Boolean operator> (const IDate& aDate) const;`

operator > — IMMTime — vbmm

Description: Returns `true` if this time is greater than the passed in time.

Member: `Boolean operator> (const IMMTime& time) const;`

operator > — IOrderedRecord — vbsample

Description: Equivalent to '!(record1 <= record2)'.

Member: `virtual IBoolean operator> (const IOrderedRecord& aRecord) const;`

operator > — IPair

Description: `True` if both coordinates are greater than those of the specified aPair.

Member: `Boolean operator> (const IPair& aPair) const;`

operator > — ITime

Description: Compares two objects to determine whether one is greater than the other.

Member: `Boolean operator> (const ITime& aTime) const;`

operator >= — IDate

Description: If the left-hand operand represents a date subsequent to or identical to the date represented by the right-hand operand, `true` is returned.

Member: `Boolean operator>= (const IDate& aDate) const;`

operator >= — IMMTime — vbmm

Description: Returns `true` if this time is greater than or equal to the passed in time.

Member: `Boolean operator>= (const IMMTime& time) const;`

operator >= — IOrderedRecord — vbsample

Description: Equivalent to '!(record1 < record2)'.

Member:
```
virtual IBoolean operator>= (const IOrderedRecord& aRecord)
const;
```

operator >= — IPair

Description: `True` if both coordinates are greater than or equal.

Member:
```
Boolean operator>= (const IPair& aPair) const;
```

operator >= — ITime

Description: Compares two objects to determine whether one is greater than or equal to the other.

Member:
```
Boolean operator>= (const ITime& aTime) const;
```

operator char * — IString

Description: Returns a `char*` pointer to the string's contents.

Member:
```
operator char* () const;
```

operator const char * — ICLibErrorInfo

Description: Returns the error text.

Member:
```
virtual operator const char* () const;
```

operator const char * — IGUIErrorInfo

Description: Returns the error text.

Member:
```
virtual operator const char* () const;
```

operator const char * — IMMErrorInfo — vbmm

Description: Returns the error text.

Member:
```
virtual operator const char* () const;
```

operator const char * — ISystemErrorInfo

Description: Returns the error text.

Member:
```
virtual operator const char* () const;
```

operator delete — IContainerObject

Description: Deallocates the storage of an object of this class.

Member:
```
void operator delete (void* pointer);
```

operator new — IContainerObject

Description: Allocates storage for an object of this class.
Member:
```
void* operator new (size_t size);
```

operator signed char * — IString

Description: Returns a signed char* pointer to the string's contents.
Member:
```
operator signed char* () const;
```

operator unsigned char * — IString

Description: Returns an unsigned char* pointer to the string's contents.
Member:
```
operator unsigned char* () const;
```

operator unsigned long — IMMTime — vbmm

Description: Returns the time as a unsigned long where each time unit is equal to 1/3000 of a second.
Member:
```
virtual operator unsigned long () const;
```

operator unsigned long — IResourceId

Description: Returns the identifier used for the resource.
Member:
```
operator unsigned long () const;
```

operator [] — IPointArray

Description: Returns a reference to the point at the specified index.
Member:
```
IPoint& operator[] (unsigned long index);
```

operator [] — IString

Description: Returns a reference to the specified character of the string.
Member:
```
const char& operator[] (unsigned index) const;
```

operator ^ — IString

Description: Performs bitwise XOR.
Member:
```
IString operator^ (const IString& aString) const;
```

operator ^= — IString

Description: Performs bitwise XOR and replaces the receiver.
Member:
```
IString& operator^= (const IString& aString);
```

operator •orValue

operator | — IRectangle

Description: Returns the rectangle representing the union of the specified rectangles.

Member:
```
IRectangle operator| (const IRectangle& rectangle) const;
```

operator | — IString

Description: Performs bitwise OR.

Member:
```
IString operator| (const char* pString) const;
```

operator |= — IRectangle

Description: Resets the rectangle to its union with the specified rectangle.

Member:
```
IRectangle& operator|= (const IRectangle& rectangle);
```

operator |= — IString

Description: Performs bitwise OR and replaces the receiver with the resulting string.

Member:
```
IString& operator|= (const char* pString);
```

operator ~ — IString

Description: Returns the string's bitwise negation (the string's complement).

Member:
```
IString operator ~ () const;
```

operator= — IResourceId

Member:
```
IResourceId& operator= (const IResourceId& resoureId);
```

orValue — IVBLongPart — vbsample

Description: Perform or operator on value.

Member:
```
virtual IVBLongPart& orValue (long orValue=1);
```

orValue — IVBShortPart — vbsample

Description: Perform or operator on value.

Member:
```
virtual IVBShortPart& orValue (short orValue=1);
```

orValue — IVBUnsignedLongPart — vbsample

Description: Perform or operator on value.

Member:
```
virtual IVBUnsignedLongPart& orValue
(unsigned long orValue=1);
```

orValue — IVBUnsignedShortPart — vbsample

Description: Perform or operator on value.

Member:
```
virtual IVBUnsignedShortPart& orValue
(unsigned short orValue=1);
```

overlayWith — I0String

Description: Replaces a specified portion of the receiver's contents with the specified string.

Member:
```
static I0String overlayWith (const IString& aString,
const char* pOverlay, unsigned index=0, char padCharacter='
');
```

overlayWith — IString

Description: Replaces a specified portion of the receiver's contents with the specified string.

Member:
```
static IString overlayWith (const IString& aString,
const IString& anOverlay, unsigned index=1,
char padCharacter=' ');
```

pageSettings — INotebook

Description: Returns information about the specified page.

Member:
```
PageSettings pageSettings (const IPageHandle& page) const;
```

pagesToEnd — INotebook

Description: Returns the number of pages in the notebook from a specified notebook page to the end of the notebook.

Member:
```
virtual unsigned long pagesToEnd (const IPageHandle& page)
const;
```

pagesToMajorTab — INotebook

Description: Returns the number of pages in the notebook between a specified notebook page and the next page that has a major tab.

Member:
```
virtual unsigned long pagesToMajorTab
(const IPageHandle& page) const;
```

pagesToMinorTab — INotebook

Description: Returns the number of pages in the notebook between a specified notebook page and the next page that has a minor tab.

Member:
```
virtual unsigned long pagesToMinorTab
(const IPageHandle& page) const;
```

parentObject — IContainerControl

Description: Returns the parent of an object in the tree view.

Member:
```
virtual IContainerObject* parentObject
(const IContainerObject* childObject) const;
```

paste — IEntryField

Description: Copies text from the clipboard to the entry field control, replacing any selected text.

Member:
```
virtual IEntryField& paste ();
```

paste — IMMRecordable — vbmm

Description: Replaces the data with data from the clipboard, from the passed in start position to the passed in end position.

Member:
```
virtual IMMRecordable& paste (const IMMTime& from=IMMTime ( ),
const IMMTime& to=IMMTime ( ), Boolean convert=true,
CallType call=IMMDevice::wait);
```

paste — IMultiLineEdit

Description: Inserts the contents of the clipboard into the MLE at the cursor position.

Member:
```
virtual IMultiLineEdit& paste ();
```

pause — IMMPlayableDevice — vbmm

Description: If the device is playing, then it pauses the device.

Member:
```
virtual IMMPlayableDevice& pause
(CallType call=IMMDevice::wait);
```

perror — stdioSamples — vbsample

Description: perror prints an error message to stderr.

Member:
```
void perror (const char* string);
```

play — IMMPlayableDevice — vbmm

Description: Starts playing the device from the passed in start position to the passed in end position.

Member:
```
virtual IMMPlayableDevice& play (const IMMTime& from=IMMTime (
), const IMMTime& to=IMMTime ( ), Boolean resumeIfPaused=true,
CallType call=IMMDevice::nowait);
```

playAt — IMMDigitalVideo — vbmm

Description: Plays the video at the specified speed from the start position to the end position.

Member:
```
IMMDigitalVideo& playAt (const IMMSpeed& speed,
const IMMTime& from=IMMTime ( ), const IMMTime& to=IMMTime (
), CallType call=IMMDevice::nowait);
```

playFast — IMMDigitalVideo — vbmm

Description: Plays the video at the fast speed from the start position to the end position with no audio.

Member:
```
IMMDigitalVideo& playFast (const IMMTime& from=IMMTime ( ),
const IMMTime& to=IMMTime ( ),
CallType call=IMMDevice::nowait);
```

playScan — IMMDigitalVideo — vbmm

Description: Plays frames when indexed.

Member:
```
IMMDigitalVideo& playScan (const IMMTime& from=IMMTime ( ),
const IMMTime& to=IMMTime ( ),
CallType call=IMMDevice::nowait);
```

playSlow — IMMDigitalVideo — vbmm

Description: Plays the video at the slow speed from the start position to the end position with no audio.

Member:
```
IMMDigitalVideo& playSlow (const IMMTime& from=IMMTime ( ),
const IMMTime& to=IMMTime ( ),
CallType call=IMMDevice::nowait);
```

pop — IRStack — vbcc

Description: Copies the last element of the collection to the given element, and removes it from the collection.

Member:
```
void pop ();
```

positionBehindSibling — IWindow

Description: Puts this window in the z-order behind the specified sibling window.

Member:
```
virtual IWindow& positionBehindSibling
(const IWindowHandle& siblingWindow);
```

positionBehindSiblings — IWindow

Description: Puts this window in the z-order behind all its sibling windows.

Member:
```
virtual IWindow& positionBehindSiblings ();
```

positionOnSiblings — IWindow

Description: Puts this window in the z-order on top of all its sibling windows.

Member:
```
virtual IWindow& positionOnSiblings ();
```

postEvent — IWindow

Description: Posts an event constructed from the arguments to the window.

Member:
```
virtual const IWindow& postEvent (const IEvent& event) const;
```

pow — mathSamples — vbsample

Description: pow calculates the value of x to the power of y.

Member:
```
double pow (double x, double y);
```

preappendText — IVBStringPart — vbsample

Description: Add to the beginning of text.

Member:
```
virtual IVBStringPart& preappendText
(const IString& appendText);
```

previousPage — INotebook

Description: Returns an IPageHandle object that references the page before the specified page.

Member:
```
virtual IPageHandle previousPage
(const IPageHandle& referencePage) const;
```

procAddress — IDynamicLinkLibrary

Description: Loads the specified procedure address from a dynamic link library.

Member:
```
void* procAddress (const char* procedureName) const;
```

push — IRStack — vbcc

Description: Adds the element to the collection as the last element (as defined for add), and sets the cursor to the added element.

Member:
```
void push (Element const& element);
```

putc — stdioSamples — vbsample

Description: putc converts c to `unsigned char` and then writes c to the output stream at the current position.

Member:
```
int putc (int c, FILE* stream);
```

putchar — stdioSamples — vbsample

Description: putchar is equivalent to putc(c, stdout).

Member:
```
int putchar (int c);
```

putenv — stdlibSamples — vbsample

Description: putenv adds new environment variables or modifies the values of existing environment variables.

Member:
```
int putenv (char* envstring);
```

puts — stdioSamples — vbsample

Description: puts writes the given string to the standard output stream stdout; it also appends a new-line character to the output.

Member:
```
int puts (const char* string);
```

putwc — stdioSamples — vbsample

Description: putwc converts the wide character wc to a multibyte character, and writes it to the stream at the current position.

Member:
```
wint_t putwc (wint_t wc, FILE* stream);
```

rand — stdlibSamples — vbsample

Description: rand generates a pseudo-random integer in the range 0 to RAND_MAX (macro defined in <stdlib.

Member:
```
int rand ();
```

read — ioSamples — vbsample

Description: read reads count bytes from the file associated with handle into buffer.

Member:
```
int read (int handle, char* buffer, unsigned int count);
```

realloc — stdlibSamples — vbsample

Description: realloc changes the size of a previously reserved storage block.

Member:
```
void realloc (void* ptr, size_t size);
```

reallocateString — ICnrEditHandler

Description: Called when text has been modified in the container and storage needs to be reallocated.

Member:
```
virtual Boolean reallocateString
(ICnrReallocStringEvent& event);
```

record — IMMDigitalVideo — vbmm

Description: Starts recording.

Member:
```
virtual IMMDigitalVideo& record (const IMMTime& end=IMMTime(),
Boolean resumeIfPaused=true, CallType call=IMMDevice::nowait);
```

record — IMMRecordable — vbmm

Description: Starts recording at the begin location till it reaches the end location.

Member:
```
virtual IMMRecordable& record (Boolean insert=true,
const IMMTime& begin=IMMTime ( ), const IMMTime& end=IMMTime (
), Boolean resumeIfPaused=true,
CallType call=IMMDevice::nowait);
```

refresh — IContainerObject

Description: Refreshes this object.

Member:
```
virtual IContainerObject& refresh
(IContainerControl* container=0);
```

refresh — IWindow

Description: Depending on which version you use, this function invalidates and redraws either.

Member:
```
virtual IWindow& refresh (RefreshType type=IWindow::paintAll);
```

refreshAllContainers — IContainerControl

Description: Refreshes all containers.

Member:
```
static void refreshAllContainers ();
```

refreshTabs — INotebook

Description: Causes all of the notebook's visible tabs to be repainted.

Member:
```
virtual INotebook& refreshTabs ();
```

registerFormat — IClipboard

Description: Registers the passed string as a private format and returns its format handle.

Member:
```
static FormatHandle registerFormat
(const char* privateFormat);
```

release — IMMDevice — vbmm

Description: Releases exclusive use of the device resources.

Member:
```
virtual IMMDevice& release (CallType call=IMMDevice::wait);
```

releasePointer — IWindow

Description: Causes the window to release the pointer capture (pointer capture is set with the function capturePointer.).

Member:
```
virtual IWindow& releasePointer ();
```

releasePresSpace — IWindow

Description: Releases the window's presentation space handle.

Member:
```
virtual void releasePresSpace
(const IPresSpaceHandle& presentationSpaceHandle) const;
```

remove — I0String

Description: Deletes the specified portion of a string from the receiver.

Member:
```
static I0String remove (const IString& aString,
unsigned startPos, unsigned numChars);
```

remove — IAccelerator

Description: Removes the accelerator without restoring the one that was previously in effect.

Member:
```
IAccelerator& remove ();
```

remove — IComboBox

Description: Removes the specified item from the list box and returns the number of items that remain.

Member:
```
virtual unsigned long remove (unsigned long index);
```

remove — IListBox

Description: Removes the specified item from the list box and returns the count of items that remain.

Member:
```
virtual unsigned long remove (unsigned long index);
```

remove — IPointArray

Description: Removes a point at the specified index.

Member:
```
IPointArray& remove (unsigned long index);
```

remove

remove — IRBag — vbcc

Description: Removes an element in the collection that is equal to the given element.

Member:
```
IBoolean remove (Element const& element);
```

remove — IREqualitySequence — vbcc

Description: Removes an element in the collection that is equal to the given element.

Member:
```
IBoolean remove (Element const& element);
```

remove — IRSet — vbcc

Description: Removes an element in the collection that is equal to the given element.

Member:
```
IBoolean remove (Element const& element);
```

remove — IRSortedBag — vbcc

Description: Removes an element in the collection that is equal to the given element.

Member:
```
IBoolean remove (Element const& element);
```

remove — IRSortedSet — vbcc

Description: Removes an element in the collection that is equal to the given element.

Member:
```
IBoolean remove (Element const& element);
```

remove — IString

Description: Deletes the specified portion of the string (that is, the substring) from the receiver.

Member:
```
static IString remove (const IString& aString,
unsigned startPos);
```

remove — IToolBar

Description: Removes the window from the tool bar.

Member:
```
virtual IToolBar& remove (IWindow* window);
```

remove — stdioSamples — vbsample

Description: remove deletes the file specified by filename.

Member:
```
int remove (const char* filename);
```

removeAll — IEntryField

Description: Deletes the entire contents of the entry field control.
Member: `virtual IEntryField& removeAll ();`

removeAll — IListBox

Description: Removes all items from the list box.
Member: `virtual IListBox& removeAll ();`

removeAll — IMultiLineEdit

Description: Deletes the entire contents of the MLE.
Member: `virtual IMultiLineEdit& removeAll ();`

removeAll — IRBag — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRDeque — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IREqualitySequence — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRHeap — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRQueue — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRSequence

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRSet — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRSortedBag — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRSortedSet — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — IRStack — vbcc

Description: Removes all elements from the collection.
Member: `void removeAll ();`

removeAll — ITextSpinButton

Description: Removes all items in the spin button.
Member: `virtual ITextSpinButton& removeAll ();`

removeAllItems — IVBContainerControl

Description: Removes all items and optionally deletes them.
Member: `virtual IVBContainerControl<Element,Collection,CnrElement>& removeAllElements (const Boolean removeElements=true);`

removeAllKeys — IAcceleratorTable

Description: Empties the accelerator table so that it contains no keys.
Member: `IAcceleratorTable& removeAllKeys ();`

removeAllOccurrences — IRBag — vbcc

Description: Removes all elements from the collection that are equal to the given element, and returns the number of elements removed.
Member: `INumber removeAllOccurrences (Element const& element);`

removeAllOccurrences — IREqualitySequence — vbcc

Description: Removes all elements from the collection that are equal to the given element, and returns the number of elements removed.
Member: `INumber removeAllOccurrences (Element const& element);`

removeAllOccurrences — IRSet — vbcc

Description: Removes all elements from the collection that are equal to the given element, and returns the number of elements removed.
Member: `INumber removeAllOccurrences (Element const& element);`

removeAllOccurrences — IRSortedBag — vbcc

Description: Removes all elements from the collection that are equal to the given element, and returns the number of elements removed.

Member:
```
INumber removeAllOccurrences (Element const& element);
```

removeAllOccurrences — IRSortedSet — vbcc

Description: Removes all elements from the collection that are equal to the given element, and returns the number of elements removed.

Member:
```
INumber removeAllOccurrences (Element const& element);
```

removeAllPages — INotebook

Description: Removes all the pages in the notebook.

Member:
```
virtual INotebook& removeAllPages ();
```

removeAtPosition — IREqualitySequence — vbcc

Description: Removes the element from the collection, which is at the given position.

Member:
```
void removeAtPosition (IPosition position);
```

removeAtPosition — IRSequence

Description: Removes the element from the collection, which is at the given position.

Member:
```
void removeAtPosition (IPosition position);
```

removeAtPosition — IRSortedBag — vbcc

Description: Removes the element from the collection, which is at the given position.

Member:
```
void removeAtPosition (IPosition position);
```

removeAtPosition — IRSortedSet — vbcc

Description: Removes the element from the collection, which is at the given position.

Member:
```
void removeAtPosition (IPosition position);
```

removeBorder — IBaseSpinButton

Description: Removes the border from the spin button.

Member:
```
virtual IBaseSpinButton& removeBorder ();
```

removeBorder — IPushButton

Description: Removes the border from the push button.
Member:
```
virtual IPushButton& removeBorder ();
```

removeButton2Items — IVBContainerControl

Description: Removes the button2ed items and optionally deletes them.
Member:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
removeButton2Elements (const Boolean removeElements=true);
```

removeColumn — IContainerControl

Description: Removes the specified column from the container.
Member:
```
virtual IContainerControl& removeColumn
(const IContainerColumn* column);
```

removeDetent — ISlider

Description: Removes the specified detent from the slider.
Member:
```
virtual ISlider& removeDetent (unsigned long detentId);
```

removeExtension — IFrameWindow

Description: Removes a frame extension (window).
Member:
```
virtual IFrameWindow& removeExtension (IWindow* extension,
Boolean updateDisplay=true);
```

removeFirst — IRDeque — vbcc

Description: Removes the first element from the collection.
Member:
```
void removeFirst ();
```

removeFirst — IREqualitySequence — vbcc

Description: Removes the first element from the collection.
Member:
```
void removeFirst ();
```

removeFirst — IRQueue — vbcc

Description: Removes the first element from the collection.
Member:
```
void removeFirst ();
```

removeFirst — IRSequence

Description: Removes the first element from the collection.
Member:
```
void removeFirst ();
```

removeFirst — IRSortedBag — vbcc

Description: Removes the first element from the collection.

Member:
```
void removeFirst ();
```

removeFirst — IRSortedSet — vbcc

Description: Removes the first element from the collection.

Member:
```
void removeFirst ();
```

removeFromCell — IMultiCellCanvas

Description: Removes a window from the canvas.

Member:
```
virtual IWindow* removeFromCell (unsigned long column,
unsigned long row);
```

removeFromWindowList — IFrameWindow

Description: Removes this frame's entry from the window list.

Member:
```
virtual IFrameWindow& removeFromWindowList ();
```

removeHelpText — IFlyOverHelpHandler

Description: Removes help text added to a window using setHelpText.

Member:
```
virtual IFlyOverHelpHandler& removeHelpText
(const IWindowHandle& handle);
```

removeHelpText — IVBInfoArea

Description: Removes info area help text for a given menu item.

Member:
```
virtual IVBInfoArea& removeHelpText (unsigned long menuId);
```

removeInUse — IContainerControl

Description: Removes in-use emphasis from the specified object.

Member:
```
virtual IContainerControl& removeInUse
(IContainerObject* object);
```

removeInUse — IContainerObject

Description: Removes the in-use emphasis for this object.

Member:
```
virtual IContainerObject& removeInUse
(IContainerControl* container=0);
```

removeKeyLike — IAcceleratorTable

Description: Removes from the table the accelerator key entry with the same keystroke (the same character or virtual key and the same Alt, Ctrl, Shift modifiers) as the specified key.

Member:
```
Boolean removeKeyLike (const IAcceleratorKey& key);
```

removeLast — IRDeque — vbcc

Description: Removes the last element from the collection.
Member:
```
void removeLast ();
```

removeLast — IREqualitySequence — vbcc

Description: Removes the last element from the collection.
Member:
```
void removeLast ();
```

removeLast — IRSequence

Description: Removes the last element from the collection.
Member:
```
void removeLast ();
```

removeLast — IRSortedBag — vbcc

Description: Removes the last element from the collection.
Member:
```
void removeLast ();
```

removeLast — IRSortedSet — vbcc

Description: Removes the last element from the collection.
Member:
```
void removeLast ();
```

removeLast — IRStack — vbcc

Description: Removes the last element from the collection.
Member:
```
void removeLast ();
```

removeLine — IMultiLineEdit

Description: Deletes the specified line of text.
Member:
```
virtual IMultiLineEdit& removeLine (unsigned long lineNumber);
```

removePage — INotebook

Description: Removes the specified page from the notebook.
Member:
```
virtual INotebook& removePage (const IPageHandle& page);
```

removeSelectedItems — IVBContainerControl

Description: Removes the selected items and optionally deletes them.
Member:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
removeSelectedElements (const Boolean removeElements=true);
```

removeTabSection — INotebook

Description: If the specified page is a major tab page, it removes the specified page and all subsequent pages up to the next major tab page.

Member:
```
virtual INotebook& removeTabSection (const IPageHandle& page);
```

removeWords — IString

Description: Deletes the specified words from the receiver's contents.

Member:
```
IString& removeWords (unsigned firstWord, unsigned numWords);
```

rename — stdioSamples — vbsample

Description: rename renames the file specified by oldname to the name given by newname.

Member:
```
int rename (const char* oldname, const char* newname);
```

reset — IAccelerator

Description: Removes the accelerator and restores the one that was previously in effect.

Member:
```
IAccelerator& reset ();
```

resetActiveColor — IWindow

Description: Resets the active color by undoing the previous set.

Member:
```
virtual IWindow& resetActiveColor ();
```

resetActiveTextBackgroundColor — ITitle

Description: Resets the active text background color by undoing a previous set.

Member:
```
virtual ITitle& resetActiveTextBackgroundColor ();
```

resetActiveTextForegroundColor — ITitle

Description: Resets the active text foreground color by undoing a previous set.

Member:
```
virtual ITitle& resetActiveTextForegroundColor ();
```

resetBackgroundColor — IWindow

Description: Resets the background color by undoing a previous set.

Member:
```
virtual IWindow& resetBackgroundColor ();
```

resetBorderColor — IWindow

Description: Resets the border color by undoing a previous set.

Member:
```
virtual IWindow& resetBorderColor ();
```

resetChangedFlag — IMultiLineEdit

Description: Resets the changed flag so that from this point forward changes to the MLE can be detected.

Member:
```
virtual IMultiLineEdit& resetChangedFlag ();
```

resetDisabledBackgroundColor — IWindow

Description: Resets the disabled background color by undoing a previous set.

Member:
```
virtual IWindow& resetDisabledBackgroundColor ();
```

resetDisabledForegroundColor — IWindow

Description: Resets the disabled foreground color by undoing a previous set.

Member:
```
virtual IWindow& resetDisabledForegroundColor ();
```

resetFillColor — IStaticText

Description: Resets the fill color by undoing a previous set.

Member:
```
virtual IStaticText& resetFillColor ();
```

resetFont — IWindow

Description: Causes the window to disregard a font set by a call to setFont.

Member:
```
virtual IWindow& resetFont ();
```

resetForegroundColor — IWindow

Description: Resets the foreground color by undoing a previous set.

Member:
```
virtual IWindow& resetForegroundColor ();
```

resetHiliteBackgroundColor — IWindow

Description: Resets the highlight background color by undoing a previous set.

Member:
```
virtual IWindow& resetHiliteBackgroundColor ();
```

resetHiliteForegroundColor — IWindow

Description: Resets the highlight foreground color by undoing a previous set.

Member:
```
virtual IWindow& resetHiliteForegroundColor ();
```

resetInactiveColor — IWindow

Description: Resets the inactive color by undoing a previous set.

Member:
```
virtual IWindow& resetInactiveColor ();
```

resetInactiveTextBackgroundColor — ITitle

Description: Resets the inactive text background color by undoing a previous set.

Member:
```
virtual ITitle& resetInactiveTextBackgroundColor ();
```

resetInactiveTextForegroundColor — ITitle

Description: Resets the inactive text foreground color by undoing a previous set.

Member:
```
virtual ITitle& resetInactiveTextForegroundColor ();
```

resetLatchedBackgroundColor — ICustomButton

Description: Resets the latched background color by undoing a previous set.

Member:
```
virtual ICustomButton& resetLatchedBackgroundColor ();
```

resetLatchedForegroundColor — ICustomButton

Description: Resets the latched foreground color by undoing a previous set.

Member:
```
virtual ICustomButton& resetLatchedForegroundColor ();
```

resetMajorTabBackgroundColor — INotebook

Description: Resets the major tab background color by changing to the major tab background color that was previously set.

Member:
```
virtual INotebook& resetMajorTabBackgroundColor ();
```

resetMajorTabForegroundColor — INotebook

Description: Resets the major tab foreground color by changing to the major tab foreground color that was previously set.

Member:
```
virtual INotebook& resetMajorTabForegroundColor ();
```

resetMinimumSize — IWindow

Description: Resets the minimum allowable size as if setMinimumSize had not been called.

Member:
```
IWindow& resetMinimumSize ();
```

resetMinorTabBackgroundColor — INotebook

Description: Resets the minor tab background color by changing to the minor tab background color that was previously set.

Member:
```
virtual INotebook& resetMinorTabBackgroundColor ();
```

resetMinorTabForegroundColor — INotebook

Description: Resets the minor tab foreground color by changing to the minor tab foreground color that was previously set.

Member:
```
virtual INotebook& resetMinorTabForegroundColor ();
```

resetPageBackgroundColor — INotebook

Description: Resets the page background color by changing to the page background color that was previously set.

Member:
```
virtual INotebook& resetPageBackgroundColor ();
```

resetShadowColor — IWindow

Description: Resets the shadow color by undoing a previous set.

Member:
```
virtual IWindow& resetShadowColor ();
```

resetSplitBarEdgeColor — ISplitCanvas

Description: Resets the color of the edges of the canvas's split bars, so the default edge color is used.

Member:
```
virtual ISplitCanvas& resetSplitBarEdgeColor ();
```

resetSplitBarMiddleColor — ISplitCanvas

Description: Resets the color of the interior of the canvas's split bars, so that the default color is used.

Member:
```
virtual ISplitCanvas& resetSplitBarMiddleColor ();
```

resetTransparentColor — IToolBarButton

Description: Resets the button so that no color is made transparent when drawing the bitmap.

Member:
```
virtual IToolBarButton& resetTransparentColor ();
```

resize — IPointArray

Description: Increases or decreases the size of the array.

Member:
```
IPointArray& resize (unsigned long newsize);
```

restore — IFrameWindow

Description: Restores a maximized or minimized window to its previous size and position.

Member:
```
virtual IFrameWindow& restore ();
```

resume — IMMPlayableDevice — vbmm

Description: Resumes playback of the device from a paused state, if resume is true.

Member:
```
virtual IMMPlayableDevice& resume (Boolean resume=true,
CallType call=IMMDevice::wait);
```

reverse — IPointArray

Description: Reverses the elements in the array.

Member:
```
IPointArray& reverse ();
```

reverse — IString

Description: Reverses the receiver's contents.

Member:
```
static IString reverse (const IString& aString);
```

reverseText — IVBStringPart — vbsample

Description: Perform reverse on text.

Member:
```
virtual IVBStringPart& reverseText ();
```

rewind — stdioSamples — vbsample

Description: rewind repositions the file pointer associated with stream to the beginning of the file.

Member:
```
void rewind (FILE* stream);
```

rightJustify — IString

Description: Right-justifies the receiver in a string of the specified length.

Member:
```
IString& rightJustify (unsigned length, char padCharacter='
');
```

rowHeight — IMultiCellCanvas

Description: Returns the current height of the specified row, in pixels.

Member:
```
unsigned long rowHeight (unsigned long row) const;
```

rpmatch — stdlibSamples — vbsample

Description: scanf reads data from the standard input stream stdin into the locations given by each entry in argument-list.

Member:
```
int rpmatch (const char* response);
```

save — IMMRecordable — vbmm

Description: Saves the current file.

Member:
```
virtual IMMRecordable& save (CallType call=IMMDevice::nowait);
```

saveAs — IMMRecordable — vbmm

Description: Saves the current files as a different file.

Member:
```
virtual IMMRecordable& saveAs (const IString& filename,
CallType call=IMMDevice::nowait);
```

saveHeadphonesSetting — IMMMasterAudio — vbmm

Description: Saves the current headphones' setting of the operating system.

Member:
```
virtual IMMMasterAudio& saveHeadphonesSetting
(IMMDevice::CallType call=IMMDevice::wait);
```

saveSpeakersSetting — IMMMasterAudio — vbmm

Description: Saves the current speakers' setting of the operating system.

Member:
```
virtual IMMMasterAudio& saveSpeakersSetting
(IMMDevice::CallType call=IMMDevice::wait);
```

saveVolume — IMMMasterAudio — vbmm

Description: Saves the current master volume setting of the operating system.

Member:
```
virtual IMMMasterAudio& saveVolume
(IMMDevice::CallType call=IMMDevice::wait);
```

scaleBy — IPair

Description: Scales the X-coordinate by xFactor, the Y-coordinate by yFactor.

Member:
```
IPair& scaleBy (double xFactor, double yFactor);
```

scaleBy — IRectangle

Description: Scales the rectangle by the specified amount.

Member:
```
IRectangle& scaleBy (Coord coord);
```

scaledBy — IPair

Description: Same as `IPair::scaleBy`, but returns a new `IPair`, leaving the original unmodified.

Member:
```
IPair scaledBy (double xFactor, double yFactor) const;
```

scaledBy — IRectangle

Description: Same as IRectangle::scaleBy, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle scaledBy (double xfactor, double yfactor) const;
```

scroll — IContainerControl

Description: Scrolls the container both horizontally and vertically.

Member:
```
virtual IContainerControl& scroll (long vericalPixels,
long horizontalPixels, Boolean rightSide=false);
```

scrollDetailsHorizontally — IContainerControl

Description: Scrolls either the right or left side of a split details view horizontally.

Member:
```
virtual IContainerControl& scrollDetailsHorizontally
(long horizontalPixels, Boolean rightSide=false);
```

scrollHorizontally — IContainerControl

Description: Scrolls the container horizontally.

Member:
```
virtual IContainerControl& scrollHorizontally (long pixels,
Boolean rightSide=false);
```

scrollToObject — IContainerControl

Description: Scrolls the container to the specified object and, optionally, a specified column into the work area.

Member:
```
virtual IContainerControl& scrollToObject
(const IContainerObject* object,
const IContainerColumn* column, Boolean leftJustify=true);
```

scrollVertically — IContainerControl

Description: Scrolls the container vertically.

Member:
```
virtual IContainerControl& scrollVertically (long pixels);
```

scrollViewHorizontallyTo — IViewPort

Description: Scrolls the view window horizontally.

Member:
```
virtual IViewPort& scrollViewHorizontallyTo
(unsigned long leftOffset);
```

scrollViewVerticallyTo — IViewPort

Description: Scrolls the view window vertically.

Member:
```
virtual IViewPort& scrollViewVerticallyTo
(unsigned long topOffset);
```

seek — IMMPlayableDevice — vbmm

Description: Sets the current position of the device to the passed in position.

Member:
```
virtual IMMPlayableDevice& seek (const IMMTime& to,
CallType call=IMMDevice::wait);
```

seekToEnd — IMMPlayableDevice — vbmm

Description: Moves the current position to the end of the data in the device.

Member:
```
virtual IMMPlayableDevice& seekToEnd
(CallType call=IMMDevice::wait);
```

seekToStart — IMMPlayableDevice — vbmm

Description: Moves the current position to the start of the data in the device.

Member:
```
virtual IMMPlayableDevice& seekToStart
(CallType call=IMMDevice::wait);
```

select — IVBContainerControl

Description: Gives the specified object selection emphasis.

Member:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
select (unsigned long collectionPosition,
Boolean select=true);
```

selectAll — IBaseListBox

Description: Sets the selection state for all items in the list box.

Member:
```
virtual IBaseListBox& selectAll ();
```

selectedCnrElements — IVBContainerControl

Description: Returns a collection of elements corresponding to the selected container objects.

Member:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
selectedElements (IVSequence<CnrElement*>& cnrObjects);
```

selectedElements — ICollectionViewListBox

Description: Returns a collection of elements corresponding to the selected list box items.

Member:
```
virtual ICollectionViewListBox<Element,Collection>&
selectedElements (Collection& elements);
```

selectedElements — IVBContainerControl

Description: Returns a collection of elements corresponding to the selected container items.

Member:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
selectedElements (Collection& elements);
```

selectHalftone — I3StateCheckBox

Description: Selects a halftone three-state check box control.

Member:
```
I3StateCheckBox& selectHalftone ();
```

selectRange — IEntryField

Description: Selects a range of text.

Member:
```
virtual IEntryField& selectRange
(const IRange& range=IRange(0,IEntryField::end),
unsigned long timestamp=0);
```

selectRange — IMultiLineEdit

Description: Selects a range of text.

Member:
```
virtual IMultiLineEdit& selectRange
(constIRange& range=IRange(0,IMultiLineEdit::end),
unsigned long timestamp=0);
```

sendEvent — IWindow

Description: Sends an event constructed from the arguments to the window.

Member:
```
virtual IEventResult sendEvent (const IEvent& event) const;
```

set — IAccelerator

Description: Changes to the new accelerator table.

Member:
```
IAccelerator& set (unsigned long accelResId);
```

setActiveWindow — IHelpWindow

Description: Sets the application window that the Information Presentation Facility (IPF) will treat a subsequent request for contextual or general help as coming from.

Member:
```
virtual IHelpWindow& setActiveWindow
(IFrameWindow* activeWindow, IFrameWindow* relativeWindow=0);
```

setAddressToDefault — ICompany — vbsample

Description: Perform the setAddressToDefault action.

Member:
```
virtual ICompany& setAddressToDefault ();
```

setAddressToDefault — ICustomer — vbsample

Description: Perform the setAddressToDefault action.

Member:
```
virtual ICustomer& setAddressToDefault ();
```

setAllEmphasis — IFont

Description: If you specify true, all of the styles are set on.

Member:
```
virtual IFont& setAllEmphasis (Boolean turnOn=true);
```

setAssociatedWindow — IHelpWindow

Description: Associates an application frame window with the help window.

Member:
```
virtual IHelpWindow& setAssociatedWindow
(IFrameWindow* associatedWindow);
```

setBitmaps — IAnimatedButton

Description: Sets the bitmaps to be used for the button.

Member:
```
virtual IAnimatedButton& setBitmaps
(const IResourceId& firstBitmap, unsigned long count);
```

setBorderSize — IFrameWindow

Description: Set the border size.

Member:
```
IFrameWindow& setBorderSize (unsigned long cx,
unsigned long cy);
```

setbuf — stdioSamples — vbsample

Description: setbuf controls buffering for the specified stream.

Member:
```
void setbuf (FILE* stream, char* buffer);
```

setChangedFlag — IEntryField

Description: Sets the entry field so that the control considers the text to have changed (or not) since the last query.

Member:
```
virtual IEntryField& setChangedFlag (Boolean changeFlag=true);
```

setChangedFlag — IMultiLineEdit

Description: Sets a flag to indicate the MLE contents have changed.

Member:
```
virtual IMultiLineEdit& setChangedFlag (Boolean changed=true);
```

setCharHeight — IFont

Description: Sets the height of the characters of a vector font.

Member:
```
virtual IFont& setCharHeight (unsigned long height,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

setCharSize — IFont

Description: Sets the size of the characters of a vector font.

Member:
```
virtual IFont& setCharSize (const ISize& size,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

setCharWidth — IFont

Description: Sets the width of the characters of a vector font.

Member:
```
virtual IFont& setCharWidth (unsigned long width,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

setCityToDefault — IAddress — vbsample

Description: Perform the setCityToDefault action.

Member:
```
virtual IAddress& setCityToDefault ();
```

setClosed — IContainerObject

Description: Removes the open emphasis in all containers.

Member:
```
virtual IContainerObject& setClosed ();
```

setColumnWidth — IMultiCellCanvas

Description: Defines the smallest width that the column will have, and whether the column is expandable.

Member:
```
virtual IMultiCellCanvas& setColumnWidth
(unsigned long column, unsigned long widthInPixels,
Boolean expandable=false);
```

setCursor — IContainerControl

Description: Gives the specified object cursor emphasis.

Member:
```
virtual IContainerControl& setCursor
(IContainerObject* object);
```

setCustomerListToDefault — ICompany — vbsample

Description: Perform the setCustomerListToDefault action.

Member:
```
virtual ICompany& setCustomerListToDefault ();
```

setData — IClipboard

Description: Copies the passed data buffer and places it on the clipboard with the format specified.

Member:
```
virtual IClipboard& setData (const char* format, void* data,
unsigned long dataLength);
```

setDataOffset — IContainerColumn

Description: Identifies where the data is located in a container object for this column.

Member:
```
virtual IContainerColumn& setDataOffset
(unsigned long dataOffset);
```

setDecrementBitmaps — ICircularSlider

Description: Sets the bitmaps used for decrement buttons of the circular slider.

Member:
```
ICircularSlider& setDecrementBitmaps
(const IBitmapHandle& leftUp, const IBitmapHandle& leftDown);
```

setDeleteColumnsOnClose — IContainerControl

Description: Deletes all columns in the container when the container is deleted.

Member:
```
virtual IContainerControl& setDeleteColumnsOnClose
(Boolean destroy=true);
```

setDeleteObjectsOnClose — IContainerControl

Description: Deletes all objects in the container when the container is deleted.

Member:
```
virtual IContainerControl& setDeleteObjectsOnClose
(Boolean destroy=true);
```

setDestroyOnClose — IFrameWindow

Description: Sets the destroy window flag on or off.

Member:
```
virtual IFrameWindow& setDestroyOnClose
(Boolean destroy=true);
```

setDialogTemplate — IVBFileDialog

Description: Specifies a dialog template resource to be used in place of the system supplied default font dialog.

Member:
```
virtual IVBFileDialog& setDialogTemplate
(const IResourceId& resourceId);
```

setDialogTemplate — IVBFontDialog

Description: Specifies a dialog template resource to be used in place of the system supplied default font dialog.

Member:
```
virtual IVBFontDialog& setDialogTemplate
(const IResourceId& templateId);
```

setDirection — IFont

Description: Sets the direction to draw the font in.

Member:
```
virtual IFont& setDirection (Direction direction);
```

setDisplayPS — IVBFontDialog

Description: Set a display presentation space, the `IFontDialog` will query the presentation space for candidate fonts.

Member:
```
virtual IVBFontDialog& setDisplayPS
(const IPresSpaceHandle& presSpaceHandle);
```

setEditColumn — IContainerControl

Description: Stores the specified column where editing occurs.

Member:
```
virtual IContainerControl& setEditColumn
(IContainerColumn* column);
```

setEditMLE — IContainerControl

Description: Stores the specified multiple-line edit (MLE) field being used for editing.

Member:
```
virtual IContainerControl& setEditMLE
(IMultiLineEdit* editField);
```

setEditObject — IContainerControl

Description: Stores the specified object where editing occurs.

Member:
```
virtual IContainerControl& setEditObject
(IContainerObject* object);
```

setEditRegion — IMultiLineEdit

Description: Sets the size of the edit region so that it covers the entire MLE window.

Member:
```
virtual IMultiLineEdit& setEditRegion ();
```

setExtendedSelection — IContainerControl

Description: Sets the selection mode to extended selection.

Member:
```
virtual IContainerControl& setExtendedSelection ();
```

setExtensionSize — IFrameWindow

Description: Sets a frame extension to the specified size.

Member:
```
virtual IFrameWindow& setExtensionSize (IWindow* extension,
double widthOrHeight);
```

setFamily — IVBFontDialog

Description: Set the default font family.

Member:
```
virtual IVBFontDialog& setFamily (const char* fontFamily);
```

setFocus — IWindow

Description: Sets the input focus to the window.

Member:
```
virtual IWindow& setFocus ();
```

setFontAngle — IFont

Description: Sets the angle to draw the vector font with.

Member:
```
virtual IFont& setFontAngle (const IPoint& point,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

setFontFromFontModally — IVBFontDialog

Description: Results in an application-modal IFontDialog being displayed.

Member:
```
virtual unsigned long setFontFromFontModally (IFont* target,
IWindow* parent=NULL,
const IFontDialog::Style style=IFontDialog::defaultStyle (),
IHandler* handler=0);
```

setFontFromWindowModally — IVBFontDialog

Description: Results in an application-modal IFontDialog being displayed.

Member:
```
virtual unsigned long setFontFromWindowModally
(IWindow* target, IWindow* parent=NULL,
const IFontDialog::Style style=IFontDialog::defaultStyle (),
IHandler* handler=0);
```

setFontShear — IFont

Description: Sets the amount of shear to apply to the vector font.

Member:
```
virtual IFont& setFontShear (const IPoint& point,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

setHandle — IClipboard

Description: Makes the passed handle shareable (so that it can be accessed by other processes), and places it on the clipboard.

Member:
```
virtual IClipboard& setHandle (const char* format,
unsigned long handle);
```

setHelpKey — IAcceleratorKey

Description: Specifies that the accelerator key request contextual help as its action.

Member:
```
IAcceleratorKey& setHelpKey ();
```

setHelpTable — IHelpWindow

Description: Sets the help table for the Information Presentation Facility (IPF) to use for satisfying user requests for contextual and general help.

Member:
```
virtual IHelpWindow& setHelpTable
(const IResourceId& helpTable);
```

setHelpText — IFlyOverHelpHandler

Description: Sets the help text for a window by specifying a string or resouce identifier.

Member:
```
virtual IFlyOverHelpHandler& setHelpText
(const IWindowHandle& handle, const IString& flyText,
const IString& longText=IString ());
```

setHelpText — IVBInfoArea

Description: Sets the text to be shown in the info area for a given menu item.

Member:
```
virtual IVBInfoArea& setHelpText (unsigned long menuId,
const IString& text);
```

setHomePhoneToDefault — ICustomer — vbsample

Description: Perform the setHomePhoneToDefault action.

Member:
```
virtual ICustomer& setHomePhoneToDefault ();
```

setIncrementBitmaps — ICircularSlider

Description: Sets the bitmaps used for increment buttons of the circular slider.

Member:
```
ICircularSlider& setIncrementBitmaps
(const IResourceId& rightUp, const IResourceId& rightDown);
```

setInitialDrive — IVBFileDialog

Description: Drive for which initial information is displayed.

Member:
```
virtual IVBFileDialog& setInitialDrive (const char* drive);
```

setInitialFileType — IVBFileDialog

Description: Extended attribute that is used to filter the initial display of files.

Member:
```
virtual IVBFileDialog& setInitialFileType
(const char* fileType);
```

setInput1 — IVBLogicalAndPart — vbsample

Description: Set the input1 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput1 ();`

setInput1 — IVBLogicalOrPart — vbsample

Description: Set the input1 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput1 ();`

setInput10 — IVBLogicalAndPart — vbsample

Description: Set the input10 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput10 ();`

setInput10 — IVBLogicalOrPart — vbsample

Description: Set the input10 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput10 ();`

setInput2 — IVBLogicalAndPart — vbsample

Description: Set the input2 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput2 ();`

setInput2 — IVBLogicalOrPart — vbsample

Description: Set the input2 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput2 ();`

setInput3 — IVBLogicalAndPart — vbsample

Description: Set the input3 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput3 ();`

setInput3 — IVBLogicalOrPart — vbsample

Description: Set the input3 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput3 ();`

setInput4 — IVBLogicalAndPart — vbsample

Description: Set the input4 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput4 ();`

setInput4 — IVBLogicalOrPart — vbsample

Description: Set the input4 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput4 ();`

setInput5 — IVBLogicalAndPart — vbsample

Description: Set the input5 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput5 ();`

setInput5 — IVBLogicalOrPart — vbsample

Description: Set the input5 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput5 ();`

setInput6 — IVBLogicalAndPart — vbsample

Description: Set the input6 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput6 ();`

setInput6 — IVBLogicalOrPart — vbsample

Description: Set the input6 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput6 ();`

setInput7 — IVBLogicalAndPart — vbsample

Description: Set the input7 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput7 ();`

setInput7 — IVBLogicalOrPart — vbsample

Description: Set the input7 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput7 ();`

setInput8 — IVBLogicalAndPart — vbsample

Description: Set the input8 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput8 ();`

setInput8 — IVBLogicalOrPart — vbsample

Description: Set the input8 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput8 ();`

setInput9 — IVBLogicalAndPart — vbsample

Description: Set the input9 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setInput9 ();`

setInput9 — IVBLogicalOrPart — vbsample

Description: Set the input9 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setInput9 ();`

setInUse — IContainerControl

Description: Gives the specified object in-use emphasis.

Member:
```
virtual IContainerControl& setInUse (IContainerObject* object,
Boolean inUse=true);
```

setItemHandle — IBaseComboBox

Description: Sets the handle of the specified list box 0-based item.

Member:
```
virtual IBaseComboBox& setItemHandle (unsigned long index,
unsigned long handle);
```

setItemHandle — IBaseListBox

Description: Sets the handle of the specified list box 0-based item.

Member:
```
virtual IBaseListBox& setItemHandle (unsigned long index,
unsigned long handle);
```

setItemHeight — IBaseListBox

Description: Sets the height of a list box item, in pixels.

Member:
```
virtual IBaseListBox& setItemHeight (unsigned long newHeight);
```

setItemText — IBaseComboBox

Description: Changes the text of the specified item in the list box portion of the combination box.

Member:
```
virtual IBaseComboBox& setItemText (unsigned long index,
const char* string, Boolean updateEntryField=false);
```

setItemText — IBaseListBox

Description: Changes the text of the specified 0-based item in the list box.

Member:
```
virtual IBaseListBox& setItemText (unsigned long index,
const char* string);
```

setKey — IAcceleratorKey

Description: Assigns the specified keystroke to the accelerator key.

Member:
```
IAcceleratorKey& setKey (const IString& accelKey,
const KeyModifier& modifier=IKey::noModifier);
```

setLayoutDistorted — IWindow

Description: Indicates that changes have occurred in the window causing the layout of the window in a canvas to be updated.

Member:
```
virtual IWindow& setLayoutDistorted
(unsigned long layoutAttributesOn,
unsigned long layoutAttributesOff);
```

setMajorTabSize — INotebook

Description: Sets the size of the notebook's major tabs in pixels.
Member:
```
virtual INotebook& setMajorTabSize
(const ISize& sizeMajorTab);
```

setMaster — IBaseSpinButton

Description: Defines a servant spin button's master.
Member:
```
virtual IBaseSpinButton& setMaster (IBaseSpinButton& master);
```

setMinorTabSize — INotebook

Description: Sets the size of the notebook's minor tabs in pixels.
Member:
```
virtual INotebook& setMinorTabSize
(const ISize& sizeMinorTab);
```

setMixedTargetEmphasis — IContainerControl

Description: Sets the drag mode to mixed-target emphasis.
Member:
```
virtual IContainerControl& setMixedTargetEmphasis ();
```

setMLEHandler — ICnrEditHandler

Description: Stores a handler to be added to the multi-line entry field control.
Member:
```
virtual void setMLEHandler (IHandler* anMLEHandler);
```

setMultipleSelection — IContainerControl

Description: Sets the selection mode to multiple selection.
Member:
```
virtual IContainerControl& setMultipleSelection ();
```

setNameToDefault — ICompany — vbsample

Description: Perform the setNameToDefault action.
Member:
```
virtual ICompany& setNameToDefault ();
```

setNameToDefault — ICustomer — vbsample

Description: Perform the setNameToDefault action.
Member:
```
virtual ICustomer& setNameToDefault ();
```

setNormalTargetEmphasis — IContainerControl

Description: Sets the drag mode to normal-target emphasis.
Member:
```
virtual IContainerControl& setNormalTargetEmphasis ();
```

setNotInput1 — IVBLogicalAndPart — vbsample

Description: Set the notInput1 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput1 ();`

setNotInput1 — IVBLogicalOrPart — vbsample

Description: Set the notInput1 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput1 ();`

setNotInput10 — IVBLogicalAndPart — vbsample

Description: Set the notInput10 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput10 ();`

setNotInput10 — IVBLogicalOrPart — vbsample

Description: Set the notInput10 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput10 ();`

setNotInput2 — IVBLogicalAndPart — vbsample

Description: Set the notInput2 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput2 ();`

setNotInput2 — IVBLogicalOrPart — vbsample

Description: Set the notInput2 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput2 ();`

setNotInput3 — IVBLogicalAndPart — vbsample

Description: Set the notInput3 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput3 ();`

setNotInput3 — IVBLogicalOrPart — vbsample

Description: Set the notInput3 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput3 ();`

setNotInput4 — IVBLogicalAndPart — vbsample

Description: Set the notInput4 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput4 ();`

setNotInput4 — IVBLogicalOrPart — vbsample

Description: Set the notInput4 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput4 ();`

setNotInput5 — IVBLogicalAndPart — vbsample

Description: Set the notInput5 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput5 ();`

setNotInput5 — IVBLogicalOrPart — vbsample

Description: Set the notInput5 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput5 ();`

setNotInput6 — IVBLogicalAndPart — vbsample

Description: Set the notInput6 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput6 ();`

setNotInput6 — IVBLogicalOrPart — vbsample

Description: Set the notInput6 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput6 ();`

setNotInput7 — IVBLogicalAndPart — vbsample

Description: Set the notInput7 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput7 ();`

setNotInput7 — IVBLogicalOrPart — vbsample

Description: Set the notInput7 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput7 ();`

setNotInput8 — IVBLogicalAndPart — vbsample

Description: Set the notInput8 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput8 ();`

setNotInput8 — IVBLogicalOrPart — vbsample

Description: Set the notInput8 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput8 ();`

setNotInput9 — IVBLogicalAndPart — vbsample

Description: Set the notInput9 (Boolean) attribute.
Member: `virtual IVBLogicalAndPart& setNotInput9 ();`

setNotInput9 — IVBLogicalOrPart — vbsample

Description: Set the notInput9 (Boolean) attribute.
Member: `virtual IVBLogicalOrPart& setNotInput9 ();`

setOKButtonText — IVBFileDialog

Description: Sets the text that appears on the OK button.

Member:
```
virtual IVBFileDialog& setOKButtonText (const char* newText);
```

setOpenDialog — IVBFileDialog

Description: Creates an Open File dialog (default).

Member:
```
virtual IVBFileDialog& setOpenDialog ();
```

setOrderedTargetEmphasis — IContainerControl

Description: Sets the drag mode to ordered-target emphasis.

Member:
```
virtual IContainerControl& setOrderedTargetEmphasis ();
```

setPageButtonSize — INotebook

Description: Sets the size in pixels of the page (arrow) buttons that are used to turn the notebook's pages.

Member:
```
virtual INotebook& setPageButtonSize
(const ISize& sizePageButton);
```

setPhoneToDefault — ICompany — vbsample

Description: Perform the setPhoneToDefault action.

Member:
```
virtual ICompany& setPhoneToDefault ();
```

setPosition — IVBFileDialog

Description: Sets the initial placement of the dialog within the parent window.

Member:
```
virtual IVBFileDialog& setPosition (const IPoint& position);
```

setPosition — IVBFontDialog

Description: Sets the initial placement of the dialog within the parent window.

Member:
```
virtual IVBFontDialog& setPosition (const IPoint& position);
```

setPreviewText — IVBFontDialog

Description: Sets the text to display in the font sample box.

Member:
```
virtual IVBFontDialog& setPreviewText
(const char* previewText);
```

setPrinterPS — IVBFontDialog

Description: Set a printer presentation space, the `IFontDialog` will query the presentation space for candidate fonts.

Member:
```
virtual IVBFontDialog& setPrinterPS
(const IPresSpaceHandle& presSpaceHandle);
```

setProgram — IMMAudioCD — vbmm

Description: Sets a new table of contents.

Member:
```
IMMAudioCD& setProgram (const IMMAudioCDContents& program);
```

setRefreshOff — IContainerControl

Description: Disables refreshing the container until IContainerControl::refresh or setRefreshOn is called to enable refreshing it.

Member:
```
virtual IContainerControl& setRefreshOff ();
```

setRefreshOff — IContainerObject

Description: Prevents refreshing of this object.

Member:
```
virtual IContainerObject& setRefreshOff ();
```

setRowHeight — IMultiCellCanvas

Description: Defines the smallest height that the row will have, and whether the row is expandable.

Member:
```
virtual IMultiCellCanvas& setRowHeight (unsigned long row,
unsigned long heightInPixels, Boolean expandable=false);
```

setSaveAsDialog — IVBFileDialog

Description: Creates a Save As dialog.

Member:
```
virtual IVBFileDialog& setSaveAsDialog ();
```

setScrollBar — IScrollBar

Description: Sets the range of all items and the number of items that are displayed.

Member:
```
virtual IScrollBar& setScrollBar
(const IRange& scrollableRange, unsigned long visibleCount);
```

setSelected — IContainerControl

Description: Gives the specified object selection emphasis.

Member:
```
virtual IContainerControl& setSelected
(IContainerObject* object, Boolean select=true);
```

setSeparator — IMenuItem

Description: Makes the menu item a separator.

Member:
```
IMenuItem& setSeparator ();
```

setShaftBreadth — IProgressIndicator

Description: Sets the shaft width in pixels for vertical progress indicators.

Member:
```
virtual IProgressIndicator& setShaftBreadth
(unsigned long breadth);
```

setSingleSelection — IContainerControl

Description: Sets the selection mode to single selection.

Member:
```
virtual IContainerControl& setSingleSelection ();
```

setSizeList — IVBFontDialog

Description: Sets the list of point size choices.

Member:
```
virtual IVBFontDialog& setSizeList (const char* sizeList);
```

setSplitBarThickness — ISplitCanvas

Description: Sets the thickness of the specified area of the split bar.

Member:
```
virtual ISplitCanvas& setSplitBarThickness (SplitBarArea area,
unsigned long thickness);
```

setSplitWindowPercentage — ISplitCanvas

Description: Sets the percentage of the canvas occupied by the specified window.

Member:
```
virtual ISplitCanvas& setSplitWindowPercentage
(IWindow* window, unsigned long percentage);
```

setStateToDefault — IAddress — vbsample

Description: Perform the setStateToDefault action.

Member:
```
virtual IAddress& setStateToDefault ();
```

setStatusText — INotebook

Description: Sets the status text for the specified page of the notebook.

Member:
```
virtual INotebook& setStatusText
(const IPageHandle& referencePage, const char* statusText);
```

setStreetToDefault — IAddress — vbsample

Description: Perform the setStreetToDefault action.

Member:
```
virtual IAddress& setStreetToDefault ();
```

setTab — IMultiLineEdit

Description: Sets the number of pixels between tab stops inside the MLE.

Member:
```
virtual IMultiLineEdit& setTab
(unsigned long tabPixelInterval);
```

setTabBitmap — INotebook

Description: Sets the tab bitmap for the specified page of the notebook.

Member:
```
virtual INotebook& setTabBitmap
(const IPageHandle& referencePage,
const IBitmapHandle& bitmap);
```

setTabText — INotebook

Description: Sets the tab text for the specified page of the notebook.

Member:
```
virtual INotebook& setTabText
(const IPageHandle& referencePage, const char* tabText);
```

setTickLength — IProgressIndicator

Description: Sets the length of one or all ticks on the progress indicator scale.

Member:
```
virtual IProgressIndicator& setTickLength
(unsigned long tickNumber, unsigned long length);
```

setTicks — IProgressIndicator

Description: For both scale1 and scale2, this function sets the following.

Member:
```
virtual IProgressIndicator& setTicks (Scale scale,
unsigned long numberOfTicks, unsigned long tickSpacing=0);
```

setTickText — IProgressIndicator

Description: Sets the text associated with the tick at the specified index.

Member:
```
virtual IProgressIndicator& setTickText
(unsigned long tickNumber, const char* text);
```

setTitle — IHelpWindow

Description: Sets the title bar text for the help window.

Member:
```
virtual IHelpWindow& setTitle (const char* titleText);
```

setTitle — IMessageBox

Description: Sets the message box's title.

Member:
```
virtual IMessageBox& setTitle (const char* title);
```

setTitle — IVBFileDialog

Description: Sets the dialog's title.

Member:
```
virtual IVBFileDialog& setTitle (const char* newTitle);
```

setTitle — IVBFontDialog

Description: Sets the font dialog's title.

Member:
```
virtual IVBFontDialog& setTitle (const char* title);
```

setTitleAlignment — IContainerControl

Description: Sets the alignment attributes for the title.

Member:
```
virtual IContainerControl& setTitleAlignment
(TitleAlignment alignment=IContainerControl::centered);
```

setTitleText — ITitle

Description: Sets all three components of a title (object text, view text, and view number) at once.

Member:
```
virtual ITitle& setTitleText (const char* objectName,
const char* viewName=0, unsigned long viewNum=0);
```

setToDefault — IAddress — vbsample

Description: Perform the setToDefault action.

Member:
```
virtual IAddress& setToDefault ();
```

setTrackTitle — IMMAudioCD — vbmm

Description: Sets the track's title.

Member:
```
IMMAudioCD& setTrackTitle (const IString& title,
unsigned long track);
```

setTreeExpandIconSize — IContainerControl

Description: Changes the size of the expanded and collapsed icons.

Member:
```
virtual IContainerControl& setTreeExpandIconSize
(const ISize& sizeIcon);
```

setTreeItemIcons — IContainerControl

Description: Sets the expanded and collapsed icons for a tree view.

Member:
```
virtual IContainerControl& setTreeItemIcons
(const IResourceId& expanded, const IResourceId& collapsed);
```

setTreeViewIndent — IContainerControl

Description: Sets the distance that children are offset horizontally from their parent.

Member:
```
virtual IContainerControl& setTreeViewIndent
(long indentPixels=-1);
```

setUserData — INotebook

Description: Sets the application data into the specified page's reserved storage.

Member:
```
virtual INotebook& setUserData
(const IPageHandle& referencePage, unsigned long userData);
```

setUsingHelp — IHelpWindow

Description: Sets the help panel to use instead of the default Using Help panel provided by the Information Presentation Facility (IPF).

Member:
```
virtual IHelpWindow& setUsingHelp (unsigned long panelId);
```

setvbuf — stdioSamples — vbsample

Description: setvbuf allows control over the buffering strategy and buffer size for a specified stream.

Member:
```
int setvbuf (FILE* stream, char* buf, int type, size_t size);
```

setWindow — IMMDigitalVideo — vbmm

Description: Use this function to supply your own window where the video is played.

Member:
```
IMMDigitalVideo& setWindow (const IWindow& window,
CallType call=IMMDevice::wait);
```

setWindow — INotebook

Description: Associates the specified note page with the application page window.

Member:
```
virtual INotebook& setWindow
(const IPageHandle& referencePage, IWindow* pageWindow=0);
```

setWindowFont — IFont

Description: Sets the font for the specified `IWindow` to the current font.

Member:
```
virtual Boolean setWindowFont (IWindow* window) const;
```

setWorkPhoneToDefault — ICustomer — vbsample

Description: Perform the setWorkPhoneToDefault action.

Member:
```
virtual ICustomer& setWorkPhoneToDefault ();
```

setZipToDefault — IAddress — vbsample

Description: Perform the setZipToDefault action.

Member:
```
virtual IAddress& setZipToDefault ();
```

show — IHelpWindow

Description: Show help panel.

Member:
```
virtual IHelpWindow& show (const char* panelName);
```

show — IMessageBox

Description: Shows a message box.

Member:
```
virtual IMessageBox::Response show (const char* message,
Severity severity, unsigned long helpId=0);
```

show — IVBFileDialog

Description: Show the file dialog.

Member:
```
virtual IVBFileDialog& show (IWindow* parent,
const IFileDialog::Style style=IFileDialog::defaultStyle (),
IHandler* handler=0);
```

show — IVBFontDialog

Description: Results in a `IFontDialog` being made visible.

Member:
```
virtual IVBFontDialog& show (IWindow* parent,
const IFontDialog::Style style=IFontDialog::defaultStyle (),
IHandler* handler=0);
```

show — IWindow

Description: Makes the window visible.

Member:
```
virtual IWindow& show (Boolean visible=true);
```

showContentsHelp — IHelpWindow

Description: Show contents help.

Member:
```
virtual IHelpWindow& show (void c=IHelpWindow::contents);
```

showCustomized — IMessageBox

Description: Shows a customized message box.

Member:
```
virtual IMessageBox::Response show (const char* message,
Style style, unsigned long helpId=0);
```

showDetailsView — IContainerControl

Description: Sets the current view to details view.

Member:
```
virtual IContainerControl& showDetailsView ();
```

showErrorInfo — IMessageBox

Description: Shows a message box.

Member:
```
virtual IMessageBox::Response show (const IErrorInfo& error,
unsigned long helpId=0);
```

showException — IMessageBox

Description: Shows a message box.

Member:
```
virtual IMessageBox::Response show
(const IException& exception, unsigned long helpId=0);
```

showFlowedNameView — IContainerControl

Description: Sets the current view to the flowed name view.

Member:
```
virtual IContainerControl& showFlowedNameView ();
```

showFlowedTextView — IContainerControl

Description: Sets the current view to the flowed text view.

Member:
```
virtual IContainerControl& showFlowedTextView ();
```

showFromFont — IVBFontDialog

Description: Results in a `IFontDialog` being made visible.

Member:
```
virtual IVBFontDialog& showFromFont (IFont* target,
IWindow* parent,
const IFontDialog::Style style=IFontDialog::defaultStyle (),
IHandler* handler=0);
```

showGeneralHelp — IHelpWindow

Description: Show general help.

Member:
```
virtual IHelpWindow& show (void c=IHelpWindow::general);
```

showHelpPanel — IHelpWindow

Description: Show help panel.

Member:
```
virtual IHelpWindow& show (const char* panelName);
```

showIconView — IContainerControl

Description: Sets the current view to the icon view.

Member:
```
virtual IContainerControl& showIconView ();
```

showIndexHelp — IHelpWindow

Description: Show index help.

Member:
```
virtual IHelpWindow& show (void c=IHelpWindow::index);
```

showKeysHelp — IHelpWindow

Description: Show keys help.

Member:
```
virtual IHelpWindow& show (void c=IHelpWindow::keys);
```

showList — IBaseComboBox

Description: Shows or hides the list box.

Member:
```
virtual IBaseComboBox& showList (Boolean show=true);
```

showMiniIcons — IContainerControl

Description: Shows mini icons in any non-text view.

Member:
```
virtual IContainerControl& showMiniIcons (Boolean mini=true);
```

showModally — IFrameWindow

Description: Displays the frame window in application-modal mode (all other application windows are disabled).

Member:
```
virtual unsigned long showModally ();
```

showModally — IVBFileDialog

Description: Show modally the file dialog.

Member:
```
virtual unsigned long showModally (IWindow* parent=NULL,
const IFileDialog::Style style=IFileDialog::defaultStyle (),
IHandler* handler=0);
```

showModally — IVBFontDialog

Description: Results in an application-modal `IFontDialog` being displayed.

Member:
```
virtual unsigned long showModally (IWindow* parent=NULL,
const IFontDialog::Style style=IFontDialog::defaultStyle (),
IHandler* handler=0);
```

showNameView — IContainerControl

Description: Sets the current view to the non-flowed name view.

Member:
```
virtual IContainerControl& showNameView ();
```

showObject — IContainerControl

Description: Shows the specified object that is currently invisible or filtered from the container.

Member:
```
virtual IContainerControl& showObject
(IContainerObject* object, Boolean visible=true);
```

showPanelIds — IHelpWindow

Description: Adds the identifier for a help panel to its title bar text.

Member:
```
virtual IHelpWindow& showPanelIds (Boolean visibleId=true);
```

showSeparators — IContainerColumn

Description: Adds either.

Member:
```
virtual IContainerColumn& showSeparators (const DataStyle&
separatorStyles=IContainerColumn::horizontalSeparator |
IContainerColumn::verticalSeparator);
```

showSourceEmphasis — IWindow

Description: Called by a pop-up menu handler to notify the window to draw pop-up menu emphasis.

Member:
```
virtual IWindow& showSourceEmphasis (Boolean show=true);
```

showSplitBar — IContainerControl

Description: Adds the split bar to the details view work area.

Member:
```
virtual IContainerControl& showSplitBar
(Boolean showSplitBar=true);
```

showTextView — IContainerControl

Description: Sets the current view to the non-flowed text view.

Member:
```
virtual IContainerControl& showTextView ();
```

showTreeIconView — IContainerControl

Description: Displays the objects as icons in a tree to represent their relationship to one another.

Member:
```
virtual IContainerControl& showTreeIconView ();
```

showTreeLine — IContainerControl

Description: Shows the lines connecting parents to children's records.

Member:
```
virtual IContainerControl& showTreeLine
(long treeLinePixelWidth=-1);
```

showTreeNameView — IContainerControl

Description: Sets the current view to the tree name view.

Member:
```
virtual IContainerControl& showTreeNameView ();
```

showTreeTextView — IContainerControl

Description: Displays the objects in a tree to represent their relationship to one another.

Member:
```
virtual IContainerControl& showTreeTextView ();
```

showUsingHelp — IHelpWindow

Description: Show using help.

Member:
```
virtual IHelpWindow& show (void c=IHelpWindow::using);
```

shrinkBy — IRectangle

Description: Moves the corners of the rectangle inward toward the center by the specified amount, either a scalar or a point.

Member:
```
IRectangle& shrinkBy (Coord coord);
```

shrunkBy — IRectangle

Description: Same as IRectangle::shrinkBy, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle shrunkBy (const IPair& pair) const;
```

sin — mathSamples — vbsample

Description: sin calculates the sine of x, with x expressed in radians.

Member:
```
double sin (double x);
```

sinh — mathSamples — vbsample

Description: sinh calculates the hyperbolic sine of x, with x expressed in radians.

Member:
```
double sinh (double x);
```

sizeBy — IRectangle

Description: Scales the rectangle by the specified value, leaving the rectangle at the same location because the bottom-left point remains fixed.

Member:
```
IRectangle& sizeBy (double factor);
```

sizedBy — IRectangle

Description: Same as IRectangle::sizeBy, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle sizedBy (const IPair& pair) const;
```

sizedTo — IRectangle

Description: Same as `IRectangle::sizeTo`, but returns a new rectangle, leaving the original unmodified.

Member:
```
IRectangle sizedTo (const IPair& pair) const;
```

sort — IREqualitySequence — vbcc

Description: Sorts the collection such that the elements occur in ascending order.

Member:
```
void sort
(long(*comparisonFunction)(Element const&,Element const&)
sortFunction);
```

sort — IRSequence

Description: Sorts the collection such that the elements occur in ascending order.

Member:
```
void sort
(long(*comparisonFunction)(Element const&,Element const&)
sortFunction);
```

sortByIconText — IContainerControl

Description: Sorts the container by the icon's text.

Member:
```
virtual IContainerControl& sortByIconText
(Boolean ascending=true);
```

space — IString

Description: Modifies the receiver so that all words are separated by the specified number of blanks.

Member:
```
IString& space (unsigned numSpaces=1, char spaceChar=' ');
```

spinDown — IBaseSpinButton

Description: Spins the button down the specified number of times.

Member:
```
virtual IBaseSpinButton& spinDown (unsigned long spinBy=1);
```

spinTo — INumericSpinButton

Description: Spins the spin button until the specified value displays.

Member:
```
virtual INumericSpinButton& spinTo (long aValue,
Boolean spinToClosest=false);
```

spinTo — ITextSpinButton

Description: Spins the button to the specified cursor or index position.

Member:
```
virtual ITextSpinButton& spinTo (unsigned long index);
```

spinUp — IBaseSpinButton

Description: Spins the button up the specified number of times.
Member:
```
virtual IBaseSpinButton& spinUp (unsigned long spinBy=1);
```

splitBarThickness — ISplitCanvas

Description: Returns the thickness of the specified area of the split bar.
Member:
```
unsigned long splitBarThickness (SplitBarArea area);
```

splitWindowPercentage — ISplitCanvas

Description: Returns the percentage of the width or height of the canvas currently occupied by the specified window.
Member:
```
unsigned long splitWindowPercentage (IWindow* window);
```

sqrt — mathSamples — vbsample

Description: sqrt calculates the nonnegative value of the square root of x.
Member:
```
double sqrt (double x);
```

squareValue — IVBDoublePart — vbsample

Description: Square the value attribute.
Member:
```
virtual IVBDoublePart& squareValue ();
```

squareValue — IVBLongPart — vbsample

Description: Square the value attribute.
Member:
```
virtual IVBLongPart& squareValue ();
```

squareValue — IVBShortPart — vbsample

Description: Square the value attribute.
Member:
```
virtual IVBShortPart& squareValue ();
```

squareValue — IVBUnsignedLongPart — vbsample

Description: Square the value attribute.
Member:
```
virtual IVBUnsignedLongPart& squareValue ();
```

squareValue — IVBUnsignedShortPart — vbsample

Description: Square the value attribute.
Member:
```
virtual IVBUnsignedShortPart& squareValue ();
```

srand — stdlibSamples — vbsample

Description: srand sets the starting point for producing a series of pseudo-random integers.

Member:
```
void srand (unsigned int seed);
```

start — IFrameWindow

Description: Creates a frame handler and adds it to the (newly created) frame window.

Member:
```
IFrameWindow& start (const IWindowHandle& hwnd);
```

start — IHandler

Description: Attaches the handler to the specified IWindow object.

Member:
```
virtual IHandler& handleEventsFor (IWindow* window);
```

start — IInfoArea

Description: Attaches the handler of the information area to the specified frame window.

Member:
```
virtual IInfoArea& handleEventsFor (IFrameWindow* frame);
```

startAnimation — IAnimatedButton

Description: Starts the animation of the button.

Member:
```
virtual IAnimatedButton& startAnimation
(unsigned long index=0);
```

startPositionTracking — IMMPlayableDevice — vbmm

Description: Starts position tracking.

Member:
```
virtual IMMPlayableDevice& startPositionTracking
(const IMMTime& timeInterval, CallType call=IMMDevice::wait);
```

startScanningBackward — IMMAudioCD — vbmm

Description: Causes the audio CD device to search backward at high speed.

Member:
```
IMMAudioCD& startScanningBackward ();
```

startScanningForward — IMMAudioCD — vbmm

Description: Starts the cd device to search forward at high speed.

Member:
```
IMMAudioCD& startScanningForward ();
```

stepFrame — IMMPlayableDevice — vbmm

Description: Steps the play one or more time units forward or backward.

Member:
```
virtual IMMPlayableDevice& stepFrame (unsigned long frames=1,
Boolean forward=true, CallType call=IMMDevice::wait);
```

stop — IHandler

Description: Detaches the handler from the specified IWindow object.

Member:
```
virtual IHandler& stopHandlingEventsFor (IWindow* window);
```

stop — IInfoArea

Description: Detaches the handler of the information area from the specified frame window.

Member:
```
virtual IInfoArea& stopHandlingEventsFor
(IFrameWindow* frame);
```

stop — IMMPlayableDevice — vbmm

Description: Stops playback of the device.

Member:
```
virtual IMMPlayableDevice& stop
(CallType call=IMMDevice::wait);
```

stopAnimation — IAnimatedButton

Description: Stops the animation of the button.

Member:
```
virtual IAnimatedButton& stopAnimation ();
```

stopPositionTracking — IMMPlayableDevice — vbmm

Description: Stops position tracking.

Member:
```
virtual IMMPlayableDevice& stopPositionTracking
(CallType call=IMMDevice::wait);
```

strip — IString

Description: Strips both leading and trailing character or characters.

Member:
```
static IString strip (const IString& aString,
const char* pStringOfChars);
```

stripBlanks — IString

Description: Strips both leading and trailing white space.

Member:
```
static IString stripBlanks (const IString& aString);
```

stripBlanksOnText — IVBStringPart — vbsample

Description: Strip blanks on text.
Member:
```
virtual IVBStringPart& stripBlanksOnText ();
```

stripLeading — IString

Description: Strips the leading character or characters.
Member:
```
static IString stripLeading (const IString& aString,
const IString& aString);
```

stripLeadingBlanks — IString

Description: Strips the leading character or characters.
Member:
```
static IString stripLeadingBlanks (const IString& aString);
```

stripTrailing — IString

Description: Strips the trailing character or characters.
Member:
```
static IString stripTrailing (const IString& aString,
char aChar);
```

stripTrailingBlanks — IString

Description: Strips the trailing character or characters.
Member:
```
static IString stripTrailingBlanks (const IString& aString);
```

strtod — stdlibSamples — vbsample

Description: strtod converts a character string to a double-precision value.
Member:
```
double strtod (const char* nptr, char** endptr);
```

strtol — stdlibSamples — vbsample

Description: strtol converts a character string to a long-integer value.
Member:
```
long int strtol (const char* nptr, char** endptr, int base);
```

strtold — stdlibSamples — vbsample

Description: strtold converts a character string pointed to by nptr to a long double value.
Member:
```
long double strtold (const char* nptr, char** endptr);
```

subString — IString

Description: Returns a specified substring of the receiver.
Member:
```
IString subString (unsigned startPos) const;
```

subtractValue — IVBDoublePart — vbsample

Member:
```
virtual IVBDoublePart& subtractValue (double subtractValue=1);
```

subtractValue — IVBLongPart — vbsample

Member: `virtual IVBLongPart& subtractValue (long subtractValue=1);`

subtractValue — IVBShortPart — vbsample

Member: `virtual IVBShortPart& subtractValue (short subtractValue=1);`

subtractValue — IVBUnsignedLongPart — vbsample

Member:
```
virtual IVBUnsignedLongPart& subtractValue
(unsigned long subtractValue=1);
```

subtractValue — IVBUnsignedShortPart — vbsample

Member:
```
virtual IVBUnsignedShortPart& subtractValue
(unsigned short subtractValue=1);
```

supportsCommand — IMMDevice — vbmm

Description: Returns `true` if the device supports the passed in command.

Member:
```
Boolean supportsCommand (IMMNotifyEvent::Command command,
CallType call=IMMDevice::wait) const;
```

swab — stdlibSamples — vbsample

Description: swab copies n bytes from source, swaps each pair of adjacent bytes, and stores the result at destination.

Member: `void swab (char* source, char* destination, int n);`

system — stdlibSamples — vbsample

Description: system passes the command string to a command processor to be run.

Member: `int system (char* string);`

tan — mathSamples — vbsample

Description: tan calculates the tangent of x, where x is expressed in radians.

Member: `double tan (double x);`

tanh — mathSamples — vbsample

Description: tanh calculates the hyperbolic tangent of x, where x is expressed in radians.

Member: `double tanh (double x);`

tempnam — stdioSamples — vbsample

Description: tempnam creates a temporary file name in another directory.

Member:
```
char tempnam (char* dir, char* prefix);
```

textLines — IFont

Description: Returns the number of lines required to fit the specified text using the specified maximum line width.

Member:
```
unsigned long textLines (const char* text,
unsigned long lineWidth) const;
```

textRectangle — IContainerControl

Description: Returns the text rectangle in window coordinates.

Member:
```
IRectangle textRectangle (const IContainerObject* object)
const;
```

textWidth — IFont

Description: Returns the width of the specified string.

Member:
```
unsigned long textWidth (const char* text) const;
```

throwCLibError — ICLibErrorInfo

Description: Creates an ICLibErrorInfo object and uses the text from it to the following: 1.

Member:
```
static void throwCLibError (const char* functionName,
const IExceptionLocation& location,
IErrorInfo::ExceptionType name=accessError,
IException::Severity severity=recoverable);
```

throwError — IErrorInfo

Description: Creates an IErrorInfo object and uses it to do the following: 1.

Member:
```
void throwError (const IExceptionLocation& location,
IErrorInfo::ExceptionType name=accessError,
IException::Severity severity=recoverable,
IException::Severity severity=recoverable);
```

throwGUIError — IGUIErrorInfo

Description: Creates an IGUIErrorInfo object and uses the text from it to do the following: 1.

Member:
```
static void throwGUIError (const char* functionName,
const IExceptionLocation& location,
IErrorInfo::ExceptionType name=accessError,
IException::Severity severity=recoverable);
```

throwMMError — IMMErrorInfo — vbmm

Description: Creates an IMMErrorInfo object and uses the text from it to do the following: 1.

Member:
```
static void throwMMError (unsigned long errorId,
const char* functionName, const IExceptionLocation& location,
IErrorInfo::ExceptionType name=accessError,
IException::Severity severity=recoverable);
```

throwSystemError — ISystemErrorInfo

Description: This function is used by the ITHROWSYSTEMERROR macro.

Member:
```
static void throwSystemError (unsigned long systemErrorId,
const char* functionName, const IExceptionLocation& location,
IErrorInfo::ExceptionType name=accessError,
IException::Severity severity=recoverable);
```

tickLength — IProgressIndicator

Description: Returns the length, in pixels, of the tick at the specified index.

Member:
```
unsigned long tickLength (unsigned long tickNumber) const;
```

tickPosition — IProgressIndicator

Description: Returns the pixel position of the tick at the specified index.

Member:
```
IPoint tickPosition (unsigned long tickNumber) const;
```

tickSpacing — IProgressIndicator

Description: Returns the number of pixels between ticks for the specified scale.

Member:
```
unsigned long tickSpacing (Scale scale) const;
```

tickText — IProgressIndicator

Description: Returns the text associated with the tick at the specified index.

Member:
```
IString tickText (unsigned long tickNumber) const;
```

tmpfile — stdioSamples — vbsample

Description: tmpfile creates a temporary binary file.

Member:
```
FILE tmpfile ();
```

tmpnam — stdioSamples — vbsample

Description: tmpnam produces a valid file name that is not the same as the name of any existing file.

Member:
```
char tmpnam (char* string);
```

trackBackward — IMMAudioCD — vbmm

Description: Moves the current position backwards the passed in number of tracks.

Member:
```
IMMAudioCD& trackBackward (unsigned long decrement=1);
```

trackForward — IMMAudioCD — vbmm

Description: Move the current position forwards the passed in number of tracks.

Member:
```
IMMAudioCD& trackForward (unsigned long increment=1);
```

trackTitle — IMMAudioCD — vbmm

Description: Returns the track title.

Member:
```
IString trackTitle (unsigned long track) const;
```

translate — IString

Description: Converts all of the receiver's characters that are in the first specified string to the corresponding character in the second specified string.

Member:
```
IString& translate (const char* pInputChars,
const char* pOutputChars, char padCharacter=' ');
```

transpose — IPair

Description: Swaps the coordinates of the ordered pair.

Member:
```
IPair& transpose ();
```

tryToLoadBitmap — IResourceLibrary

Description: Attempts to load a bitmap from a resource file.

Member:
```
virtual IBitmapHandle tryToLoadBitmap (unsigned long bitmapId,
const ISize& bitmapSize, Boolean cached=true) const;
```

tryToLoadIcon — IResourceLibrary

Description: Attempts to load an icon from a resource file.

Member:
```
virtual IPointerHandle tryToLoadIcon (unsigned long iconId,
Boolean cached=true) const;
```

tryToLoadMessage — IResourceLibrary

Description: Attempts to load a message resource from a resource file.

Member:
```
virtual IString tryToLoadMessage (unsigned long messageId)
const;
```

tryToLoadString — IResourceLibrary

Description: Attempts to load a string resource from a resource file.

Member:
```
virtual IString tryToLoadString (unsigned long stringId)
const;
```

turnToPage — INotebook

Description: Moves a specified page to the top of the notebook.

Member:
```
virtual INotebook& turnToPage (const IPageHandle& page);
```

umask — ioSamples — vbsample

Description: umask sets the file permission mask of the environment for the currently running process to the mode specified by pmode.

Member:
```
int umask (int pmode);
```

undo — IMultiLineEdit

Description: Restores the MLE contents to the state they were in before the last change.

Member:
```
virtual IMultiLineEdit& undo ();
```

undo — ISubmenu

Description: Reverses all the changes made to the menu using the `ISubmenu` member functions.

Member:
```
virtual ISubmenu& undo ();
```

ungetc — stdioSamples — vbsample

Description: ungetc pushes the `unsigned` character c back onto the given input stream.

Member:
```
int ungetc (int c, FILE* stream);
```

ungetwc — stdioSamples — vbsample

Description: ungetwc pushes the wide character by wc back onto the input stream.

Member:
```
wint_t ungetwc (wint_t wc, FILE* stream);
```

unhighlight — IButton

Description: Turns off the button's highlight state.

Member:
```
virtual IButton& unhighlight ();
```

unionWith — IRBag — vbcc

Description: Makes the collection the union of the collection and the given collection.

Member:
```
void unionWith (IABag<Element>const& bag);
```

unionWith — IRSet — vbcc

Description: Makes the collection the union of the collection and the given collection.

Member:
```
void unionWith (IASet<Element>const& collection);
```

unionWith — IRSortedBag — vbcc

Description: Makes the collection the union of the collection and the given collection.

Member:
```
void unionWith (IASortedBag<Element>const& bag);
```

unionWith — IRSortedSet — vbcc

Description: Makes the collection the union of the collection and the given collection.

Member:
```
void unionWith (IASortedSet<Element>const& collection);
```

unlatch — ICustomButton

Description: Puts the button in the unlatched (default) state.

Member:
```
virtual ICustomButton& unlatch ();
```

unlink — stdioSamples — vbsample

Description: unlink deletes the file specified by pathname.

Member:
```
int unlink (const char* pathname);
```

update — IAcceleratorTable

Description: Replaces the accelerator keys used by the specified frame window with the entries in the accelerator table object.

Member:
```
IAcceleratorTable& update (IFrameWindow& frame);
```

update — IFrameWindow

Description: Reconfigures all of the controls and extensions for the frame window.

Member:
```
virtual IFrameWindow& update ();
```

upperCase — IString

Description: Translates all lower-case letters in the receiver to upper-case.

Member:
```
static IString upperCase (const IString& aString);
```

useBitmapOnly — IFont

Description: Sets whether the IFont object uses only bit-map fonts.

Member:
```
virtual IFont& useBitmapOnly (Boolean bitmapOnly=true,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

useDefaultWindow — IMMDigitalVideo — vbmm

Description: Specifies that the digital video class creates and manages its own video window.

Member:
```
IMMDigitalVideo& useDefaultWindow
(CallType call=IMMDevice::wait);
```

useExtensionMinimumSize — IFrameWindow

Description: Sizes the width or height of the frame extension based on the minimum size of the window it contains.

Member:
```
virtual IFrameWindow& useExtensionMinimumSize
(IWindow* extension);
```

useNonPropOnly — IFont

Description: Sets whether the IFont object uses only non-proportional fonts.

Member:
```
virtual IFont& useNonPropOnly
(Boolean nonProportionalOnly=true,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

useVectorOnly — IFont

Description: Sets whether the IFont object uses only vector fonts.

Member:
```
virtual IFont& useVectorOnly (Boolean vectorOnly=true,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

wcsid — stdlibSamples — vbsample

Description: wcsid determines the character set identifier for the specified wide character wc.

Member:
```
int wcsid (const wchar_t c);
```

wcstombs — stdlibSamples — vbsample

Description: wcstombs converts the wide-character string pointed to by string into the multibyte array pointed to by dest.

Member:
```
size_t wcstombs (char* dest, const wchar_t* string, size_t n);
```

wctob — stdioSamples — vbsample

Description: wctomb converts the wide character wc into a multibyte character and stores it in the location pointed to by string.

Member:
```
int wctob (wint_t wc);
```

wctomb — stdlibSamples — vbsample

Description: wctomb converts the wide character wc into a multibyte character and stores it in the location pointed to by string.

Member:
```
int wctomb (char* string, wchar_t wc);
```

window — INotebook

Description: Returns the application page window associated with the specified notebook page.

Member:
```
virtual IWindow* window (const IPageHandle& page) const;
```

windowInCell — IMultiCellCanvas

Description: Returns the child window occupying the specified cell.

Member:
```
virtual IWindow* windowInCell (unsigned long startingColumn,
unsigned long startingRow) const;
```

word — IString

Description: Returns a copy of the specified white-space-delimited word in the receiver.

Member:
```
IString word (unsigned wordNumber) const;
```

wordIndexOfPhrase — IString

Description: Returns the word number of the first word in the receiver that matches the specified phrase.

Member:
```
unsigned wordIndexOfPhrase (const IString& aPhrase,
unsigned startWord=1) const;
```

words — IString

Description: Returns a substring of the receiver that starts at a specified word and is comprised of a specified number of words.

Member:
```
IString words (unsigned firstWord, unsigned numWords) const;
```

write — ioSamples — vbsample

Description: write writes count bytes from buffer into the file associated with handle.

Member:
```
int write (int handle, const void* buffer,
unsigned int count);
```

write — ITrace

Description: Writes the specified text.

Member:
```
static void write (const IString& text);
```

writeToQueue — ITrace

Description: Sets the location for output to \\QUEUES\\PRINTF32.

Member:
```
static void writeToQueue ();
```

writeToStandardError — ITrace

Description: Sets the location for output to the standard error stream.

Member:
```
static void writeToStandardError ();
```

writeToStandardOutput — ITrace

Description: Sets the location for output to the standard output stream.

Member:
```
static void writeToStandardOutput ();
```

x2b — IString

Description: Converts a string of hexadecimal digits to a string of binary digits.

Member:
```
IString& x2b ();
```

x2c — IString

Description: Converts a string of hexadecimal digits to a normal string of characters.

Member:
```
static IString x2c (const IString& aString);
```

x2d — IString

Description: Converts a string of hexadecimal digits to a string of decimal digits.

Member:
```
static IString x2d (const IString& aString);
```

_alloca — stdlibSamples — vbsample

Description: _alloca is a built-in function that temporarily allocates size bytes of storage space from the program's stack.

Member:
```
void _alloca (size_t size);
```

_atold — stdlibSamples — vbsample

Description: _atold converts a character string pointed to by nptr to a `long double` value.

Member:
```
long double _atold (const char* nptr);
```

_cabs — mathSamples — vbsample

Description: _cabs calculates the absolute value of a complex number.

Member:
```
double _cabs (struct complex z);
```

_chsize — ioSamples — vbsample

Description: _chsize lengthens or cuts off the file associated with handle to the length specified by size.

Member:
```
int _chsize (int handle, long size);
```

_clear87 — floatSamples — vbsample

Description: _clear87 gets the floating-point status word and then clears it.

Member:
```
unsigned int _clear87 ();
```

_control87 — floatSamples — vbsample

Description: _control87 gets the current floating-point control word and then sets it.

Member:
```
unsigned int _control87 (unsigned int new, unsigned int mask);
```

_crotl — stdlibSamples — vbsample

Description: The _crotl and _crotr functions rotate the character value by shift bits.

Member:
```
unsigned char _crotl (unsigned char value, int shift);
```

_debug_calloc — stdlibSamples — vbsample

Description: _debug_calloc is the debug version of calloc.

Member:
```
void _debug_calloc (size_t num, size_t size, const char* file,
size_t line);
```

_debug_free — stdlibSamples — vbsample

Description: _debug_free is the debug version of free.

Member:
```
void _debug_free (void* ptr, const char* file, size_t line);
```

_debug_heapmin — stdlibSamples — vbsample

Description: _debug_heapmin is the debug version of _heapmin.
Member:
```
int _debug_heapmin (const char* file, size_t line);
```

_debug_malloc — stdlibSamples — vbsample

Description: _debug_malloc is the debug version of malloc.
Member:
```
void _debug_malloc (size_t size, const char* file,
size_t line);
```

_debug_realloc — stdlibSamples — vbsample

Description: _debug_realloc is the debug version of realloc.
Member:
```
void _debug_realloc (void* ptr, size_t size, const char* file,
size_t line);
```

_debug_tcalloc — stdlibSamples — vbsample

Description: _debug_tcalloc is the debug version of _tcalloc.
Member:
```
void _debug_tcalloc (size_t num, size_t size,
const char* file, size_t line);
```

_debug_tfree — stdlibSamples — vbsample

Description: _debug_tfree is the debug version of _tfree.
Member:
```
void _debug_tfree (void* ptr, const char* file, size_t line);
```

_debug_theapmin — stdlibSamples — vbsample

Description: _debug_theapmin is the debug version of _theapmin.
Member:
```
int _debug_theapmin (const char* file, size_t line);
```

_debug_tmalloc — stdlibSamples — vbsample

Description: _debug_tmalloc is the debug version of _tmalloc.
Member:
```
void _debug_tmalloc (size_t size, const char* file,
size_t line);
```

_debug_trealloc — stdlibSamples — vbsample

Description: The _debug_trealloc function is the debug version of _trealloc.
Member:
```
void _debug_trealloc (void* ptr, size_t size,
const char* file, size_t line);
```

_dump_allocated_delta — stdlibSamples — vbsample

Description: dup associates a second file handle with a currently open file.
Member:
```
void _dump_allocated_delta (int nbytes);
```

_ecvt — stdlibSamples — vbsample

Description: _ecvt converts the floating-point number value to a character string.

Member:
```
char _ecvt (double value, int ndigits, int* decptr,
int* signptr);
```

_endthread — stdlibSamples — vbsample

Description: _endthread ends a thread that you previously created with _beginthread.

Member:
```
void _endthread ();
```

_exit — stdlibSamples — vbsample

Description: _exit ends the calling process without calling functions registered by _onexit or atexit.

Member:
```
void _exit (int status);
```

_fcloseall — stdioSamples — vbsample

Description: _fcloseall closes all open streams, except stdin, stdout, and stderr.

Member:
```
int _fcloseall ();
```

_fcvt — stdlibSamples — vbsample

Description: _fcvt converts the floating-point number value to a character string.

Member:
```
char _fcvt (double value, int ndec, int* decptr,
int* signptr);
```

_fgetchar — stdioSamples — vbsample

Description: _fgetchar reads a single character from the stdin stream.

Member:
```
int _fgetchar ();
```

_filelength — ioSamples — vbsample

Description: _filelength returns the length, in bytes, of the file associated with handle.

Member:
```
long _filelength (int handle);
```

_flushall — stdioSamples — vbsample

Description: _flushall causes the system to write to file the contents of all buffers associated with open output streams (including stdin, stdout, and stderr).

Member:
```
int _flushall ();
```

_fpreset — floatSamples — vbsample

Description: _fpreset resets the floating-point unit to the default state that the math library requires to function correctly.

Member:
```
void _fpreset ();
```

_fputchar — stdioSamples — vbsample

Description: _fputchar writes the single character c to the stdout stream at the current position.

Member:
```
int _fputchar (int c);
```

_freemod — stdlibSamples — vbsample

Description: _freemod frees all references to a dynamic link library (DLL) for the calling process.

Member:
```
int _freemod (unsigned long module_handle);
```

_fullpath — stdlibSamples — vbsample

Description: _fullpath gets the full path name of the given partial path name partialpath, and stores it in pathbuf.

Member:
```
char _fullpath (char* pathbuf, char* partialpath, size_t n);
```

_gcvt — stdlibSamples — vbsample

Description: _gcvt converts a floating-point value to a character string pointed to by buffer.

Member:
```
char _gcvt (double value, int ndec, char* buffer);
```

_heapmin — stdlibSamples — vbsample

Description: _heapmin returns all unused memory from the default runtime heap to the operating system.

Member:
```
int _heapmin ();
```

_heap_check — stdlibSamples — vbsample

Description: _heapchk checks the default storage heap for minimal consistency by checking all allocated and freed objects on the heap.

Member:
```
void _heap_check ();
```

_itoa — stdlibSamples — vbsample

Description: _itoa converts the digits of the given value to a character string that ends with a null character and stores the result in string.

Member:
```
char _itoa (int value, char* string, int radix);
```

_j0 — mathSamples — vbsample

Description: Bessel functions solve certain types of differential equations.

Member:
```
double _j0 (double x);
```

_loadmod — stdlibSamples — vbsample

Description: _loadmod loads a dynamic link library (DLL) for the calling process.

Member:
```
int _loadmod (char* module_name,
unsigned long* module_handle);
```

_lrotl — stdlibSamples — vbsample

Description: _lrotl and _lrotr rotate the `unsigned long` integer value by shift bits.

Member:
```
unsigned long _lrotl (unsigned long value, int shift);
```

_ltoa — stdlibSamples — vbsample

Description: _ltoa converts the digits of the given `long` integer value to a character string that ends with a null character and stores the result in string.

Member:
```
char _ltoa (long value, char* string, int radix);
```

_makepath — stdlibSamples — vbsample

Description: _makepath creates a single path name, composed of a drive letter, directory path, file name, and file name extension.

Member:
```
void _makepath (char* path, char* drive, char* dir,
char* fname, char* ext);
```

_matherr — mathSamples — vbsample

Description: _matherr processes errors generated by the functions in the math library.

Member:
```
int _matherr (struct exception* x);
```

_msize — stdlibSamples — vbsample

Description: _msize determines the number of bytes that were allocated to the pointer argument ptr.

Member:
```
size_t _msize (void* ptr);
```

_onexit — stdlibSamples — vbsample

Description: _onexit records the address of a function func to call when the program ends normally.

Member:
```
onexit_t _onexit (onexit_t func);
```

_rmtmp — stdioSamples — vbsample

Description: _rmtmp closes and deletes all temporary files in all directories that are held open by the calling process.

Member:
```
int _rmtmp ();
```

_rotl — stdlibSamples — vbsample

Description: These functions take a 4-byte integer value and rotate it by shift bits.

Member:
```
unsigned int _rotl (unsigned int value, int shift);
```

_searchenv — stdlibSamples — vbsample

Description: _searchenv searches for the target file in the specified domain.

Member:
```
void _searchenv (char* name, char* env_var, char* path);
```

_setmode — ioSamples — vbsample

Description: _setmode sets the translation mode of the file given by handle to mode.

Member:
```
int _setmode (int handle, int mode);
```

_set_crt_msg_handle — stdioSamples — vbsample

Description: _set_crt_msg_handle changes the file handle to which runtime messages are sent, which is usually file handle 2, to fh.

Member:
```
int _set_crt_msg_handle (int fh);
```

_sopen — ioSamples — vbsample

Description: _sopen opens the file specified by pathname and prepares the file for subsequent shared reading or writing as defined by oflag and shflag.

Member:
```
int _sopen (char* pathname, int oflag, int shflag, int pmode);
```

_splitpath — stdlibSamples — vbsample

Description: _splitpath decomposes an existing path name path into its four components.

Member:
```
void _splitpath (char* path, char* drive, char* dir,
char* fname, char* ext);
```

_status87 — floatSamples — vbsample

Description: _status87 gets the current floating-point status word.

Member:
```
unsigned int _status87 ();
```

_tcalloc — stdlibSamples — vbsample

Description: _tcalloc allocates tiled memory for an array of number elements, each of length size bytes, and initializes all bits of each element to 0.

Member:
```
void _tcalloc (size_t number, size_t size);
```

_tdump_allocated_delta — stdlibSamples — vbsample

Description: _tell gets the current position of the file pointer associated with handle.

Member:
```
void _tdump_allocated_delta (int nbytes);
```

_tell — ioSamples — vbsample

Description: _tell gets the current position of the file pointer associated with handle.

Member:
```
long _tell (int handle);
```

_tfree — stdlibSamples — vbsample

Description: _tfree frees the tiled memory pointed to by ptr that has been allocated by one of the memory management functions.

Member:
```
void _tfree (void* ptr);
```

_threadstore — stdlibSamples — vbsample

Description: _threadstore provides access to a private thread pointer that is initialized to NULL.

Member:
```
void _threadstore ();
```

_tmalloc — stdlibSamples — vbsample

Description: _tmalloc allocates tiled memory for an object of size size, the value of which is indeterminate.

Member:
```
void _tmalloc (size_t size);
```

_trealloc — stdlibSamples — vbsample

Description: _trealloc changes the size of the tiled memory pointed to ptr to the specified size, in bytes.

Member:
```
void _trealloc (void* ptr, size_t size);
```

_ultoa — stdlibSamples — vbsample

Description: _ultoa converts the digits of the given `unsigned long` value to a null-terminated character string and stores the result in string.

Member:
```
char _ultoa (unsigned long value, char* string, int radix);
```

__eof — ioSamples — vbsample

Description: __eof determines whether the file pointer has reached the end-of-file for the file associated with handle.

Member:
```
int __eof (int handle);
```

__parmdwords — stdlibSamples — vbsample

Description: The __parmdwords function returns the hidden parameter passed in the AL register for _System linkage calls.

Member:
```
unsigned char __parmdwords ();
```

Chapter 165. Attributes

acquired (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device is ready.

Get:
```
Boolean isAcquired (CallType call=IMMDevice::wait) const;
```

actionType (ICommand::ActionType) — IAcceleratorKey

Description: Identifies if the accelerator key runs an application command, system command, or invokes help.

Get:
```
ICommand::ActionType actionType () const;
```

activeColor (IColor) — IWindow

Description: Returns the active color value of the window area.

Get:
```
virtual IColor activeColor () const;
```

Set:
```
virtual IWindow& setActiveColor (const IColor& activeColor);
```

ID:
```
activeColorId
```

activeTextBackgroundColor (IColor) — ITitle

Description: Returns the active text background color value of the title area.

Get:
```
virtual IColor activeTextBackgroundColor () const;
```

Set:
```
virtual ITitle& setActiveTextBackgroundColor
(const IColor& activeTextBackgroundColor);
```

ID:
```
activeTextBackgroundColorId
```

© Copyright IBM Corp. 1995, 1996

activeTextForegroundColor (IColor) — ITitle

Description: Returns the active text foreground color value of the title area.

Get:
```
virtual IColor activeTextForegroundColor () const;
```

Set:
```
virtual ITitle& setActiveTextForegroundColor
(const IColor& activeTextForegroundColor);
```

ID:
```
activeTextForegroundColorId
```

address (IAddress*) — ICompany — vbsample

Description: Query the address (IAddress*) attribute.

Get:
```
virtual IAddress* address () const;
```

Set:
```
virtual ICompany& setAddress (IAddress* address);
```

ID:
```
addressId
```

address (IAddress*) — ICustomer — vbsample

Description: Query the address (IAddress*) attribute.

Get:
```
virtual IAddress* address () const;
```

Set:
```
virtual ICustomer& setAddress (IAddress* address);
```

ID:
```
addressId
```

aliasName (IString) — IMMDevice — vbmm

Description: Returns the alias associated with the device during the open procedure.

Get:
```
IString aliasName () const;
```

alignment (IBaseSpinButton::Alignment) — IBaseSpinButton

Description: Returns the current alignment in the spin field of this spin button object.

Get:
```
Alignment alignment () const;
```

Set:
```
virtual IBaseSpinButton& setAlignment
(Alignment alignment=IBaseSpinButton::left);
```

alignment (IEntryField::Alignment) — IEntryField

Description: Returns the current alignment for this entry field object.

Get:
```
Alignment alignment () const;
```

Set:
```
virtual IEntryField& setAlignment (Alignment alignment);
```

alignment (IProgressIndicator::Alignment) — IProgressIndicator

Description: Returns the current alignment of this progress indicator object.

Get:
```
Alignment alignment () const;
```

alignment (ISetCanvas::Alignment) — ISetCanvas

Description: Returns the enumerator for deck alignment.
Get:
```
Alignment alignment () const;
```
Set:
```
virtual ISetCanvas& setAlignment (Alignment alignment);
```

alignment (IStaticText::Alignment) — IStaticText

Description: Returns the current alignment for this static text object.
Get:
```
Alignment alignment () const;
```
Set:
```
virtual IStaticText& setAlignment
(Alignment alignment=IStaticText::topLeft);
```

allowsDragDelete (Boolean) — IToolBarButton

Description: Returns `true` if the noDragDelete style is not set for the tool bar button.
Get:
```
Boolean allowsDragDelete () const;
```
Set:
```
virtual IToolBarButton& enableDragDelete
(Boolean enable=true);
```

allowsDragDrop (Boolean) — IToolBar

Description: Returns `true` if the tool bar supports drag and drop.
Get:
```
Boolean allowsDragDrop () const;
```
Set:
```
virtual IToolBar& enableDragDrop (Boolean enable=true);
```

allowsMouseClickFocus (Boolean) — IButton

Description: Returns whether the button can receive the focus when clicked with the mouse.
Get:
```
Boolean allowsMouseClickFocus () const;
```
Set:
```
virtual IButton& enableMouseClickFocus
(Boolean allowsMouseClickFocus=true);
```

alphabetic (Boolean) — IString

Description: If all the characters are in {'A'-'Z','a'-'z'}, `true` is returned.
Get:
```
Boolean isAlphabetic () const;
```

alphanumeric (Boolean) — IString

Description: If all the characters are in {'A'-'Z','a'-'z','0'-'9'}, `true` is returned.
Get:
```
Boolean isAlphanumeric () const;
```

animatedWhenLatched (Boolean) — IAnimatedButton

Description: Returns `true` if the animateWhenLatched style is set.

Get:
```
Boolean isAnimatedWhenLatched () const;
```

Set:
```
virtual IAnimatedButton& enableAnimateWhenLatched
(Boolean enable=true);
```

animationRate (unsigned long) — IAnimatedButton

Description: Returns the current animation rate for the button.

Get:
```
unsigned long animationRate () const;
```

Set:
```
virtual IAnimatedButton& setAnimationRate
(unsigned long animationRate=1000);
```

animationStarted (Boolean) — IAnimatedButton

Description: Returns `true` if the button is currently animated.

Get:
```
Boolean isAnimationStarted () const;
```

anyElement (Element const) — IRBag — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRDeque — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IREqualitySequence — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRHeap — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRQueue — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRSequence

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRSet — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRSortedBag — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRSortedSet — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

anyElement (Element const) — IRStack — vbcc

Description: Returns a reference to an arbitrary element of the collection.

Get:
```
Element const& anyElement () const;
```

area (IRectangle::Coord) — IRectangle

Description: Returns the area of the rectangle.

Get:
```
Coord area () const;
```

areDetailsViewTitlesVisible (Boolean) — IContainerControl

Description: Queries whether the column titles are currently displayed for details view.

Get:
```
Boolean areDetailsViewTitlesVisible () const;
```

Set:
```
virtual IContainerControl& showDetailsViewTitles
(Boolean areDetailsViewTitlesVisible=true);
```

areHeadphonesEnabled (Boolean) — IMMMasterAudio — vbmm

Description: Returns if the headphones' setting is enabled for the passed in source (either the current setting or the saved setting); otherwise, returns `false`.

Get:
```
Boolean areHeadphonesEnabled
(SettingSource source=IMMMasterAudio::current,
IMMDevice::CallType call=IMMDevice::wait) const;
```

areSpeakersEnabled (Boolean) — IMMMasterAudio — vbmm

Description: Returns the `true` if the speakers' setting is enabled for the passed in source (either the current setting or the saved setting); otherwise, returns `false`.

Get:
```
Boolean areSpeakersEnabled
(SettingSource source=IMMMasterAudio::current,
IMMDevice::CallType call=IMMDevice::wait) const;
```

armPixelOffset (unsigned long) — IProgressIndicator

Description: Returns the offset, in pixels, of the arm from the home position.

Get:
```
virtual unsigned long armPixelOffset () const;
```

Set:
```
virtual IProgressIndicator& moveArmToPixel
(unsigned long armPixelOffset);
```

ID: `armChangeId`

armRange (IRange) — ICircularSlider

Description: Returns the range of values over which the arm can be moved.

Get:
```
IRange armRange () const;
```

Set:
```
ICircularSlider& setArmRange (const IRange& armRange);
```

armRange (unsigned long) — IProgressIndicator

Description: Returns the number of pixels over which the arm can move.

Get:
```
virtual unsigned long armRange () const;
```

armSize (ISize) — ISlider

Description: Returns the size of the arm in pixels.

Get:
```
ISize armSize () const;
```

Set:
```
virtual ISlider& setArmSize (const ISize& armSize);
```

armTickOffset (unsigned long) — IProgressIndicator

Description: Returns the position of the arm as a tick number.

Get:
```
virtual unsigned long armTickOffset () const;
```

Set:
```
virtual IProgressIndicator& moveArmToTick
(unsigned long armTickOffset);
```

ID: armChangeId

asACCEL (ACCEL) — IAcceleratorKey

Description: Converts an accelerator key object into a data structure defined by the presentation system for use with its APIs.

Get:
```
ACCEL asACCEL () const;
```

asDebugInfo (IString) — IBase

Description: This function obtains the diagnostic version of an object's contents.

Get:
```
IString asDebugInfo () const;
```

asDebugInfo (IString) — IInfoArea

Description: This function obtains the diagnostic version of an object's contents.

Get:
```
IString IWindow::asDebugInfo () const;
```

asDouble (double) — IString

Description: Returns, as a double, the number that the string represents.

Get:
```
double asDouble () const;
```

asInt (long) — IString

Description: Returns the number that the string represents as a long integer.

Get:
```
long asInt () const;
```

asMMTime (unsigned long) — IMMTime — vbmm

Description: Returns this time value as an unsigned long that can be used to create an IMMTime.

Get:
```
virtual unsigned long asMMTime () const;
```

asPOINTL (struct _POINTL) — IPoint

Description: Renders the point as a POINTL structure.

Get:
```
struct _POINTL asPOINTL () const;
```

asRECTL (struct _RECTL) — IRectangle

Description: Converts the rectangle into a system dependent structure.
Get:
```
struct _RECTL asRECTL () const;
```

asRGBLong (long) — IColor

Description: Returns the red, green, and blue color values combined into a `long` integer.
Get:
```
long asRGBLong () const;
```

asSeconds (long) — ITime

Description: Returns the number of seconds since midnight.
Get:
```
long asSeconds () const;
```

asSIZEL (SIZEL) — ISize

Description: Returns the `ISize` as a SIZEL structure.
Get:
```
SIZEL asSIZEL () const;
```

asString (IString) — IBase

Description: This function obtains the standard version of an object's contents.
Get:
```
IString asString () const;
```

asString (IString) — IInfoArea

Description: This function obtains the standard version of an object's contents.
Get:
```
IString IWindow::asString () const;
```

asUnsigned (unsigned long) — IString

Description: Returns, as an `unsigned long`, the integer that the string represents.
Get:
```
unsigned long asUnsigned () const;
```

audioEnabled (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the audio for the passed in channel is turned on.

Get:
```
Boolean isAudioEnabled (AudioChannel channel=IMMDevice::all,
CallType call=IMMDevice::wait);
```

Set:
```
virtual IMMDevice& enableAudio (Boolean enable=true,
AudioChannel audioEnabled=IMMDevice::all,
const IMMMillisecondTime& over=IMMMillisecondTime (),
CallType call=IMMDevice::wait);
```

autoDeleteObject (Boolean) — IWindow

Description: Returns `true` if the window object is deleted when a destroy event is dispatched to the window.

Get:
```
Boolean isAutoDeleteObject () const;
```

Set:
```
IWindow& setAutoDeleteObject (Boolean autoDeleteObject=true);
```

autoDestroyWindow (Boolean) — IWindow

Description: Returns `true` if the presentation window is destroyed when the window object is deleted.

Get:
```
Boolean isAutoDestroyWindow () const;
```

Set:
```
IWindow& setAutoDestroyWindow
(Boolean autoDestroyWindow=false);
```

autoLatchEnabled (Boolean) — ICustomButton

Description: Returns `true` if the autoLatch style is set for the button.

Get:
```
virtual Boolean isAutoLatchEnabled () const;
```

Set:
```
virtual ICustomButton& enableAutoLatch
(Boolean autoLatchEnabled=true);
```

autoPlayEnabled (Boolean) — IMMAudioCD — vbmm

Description: Returns `true`, if autoplay is turned on.

Get:
```
Boolean isAutoPlayEnabled () const;
```

Set:
```
IMMAudioCD& enableAutoPlay (Boolean autoPlayEnabled=true);
```

autoScroll (Boolean) — IEntryField

Description: Queries whether the style autoScroll is set on an entry field control.

Get:
```
Boolean isAutoScroll () const;
```

Set:
```
virtual IEntryField& enableAutoScroll
(Boolean autoScroll=true);
```

autoSelect (Boolean) — I3StateCheckBox

Description: If the three-state check box control has the autoSelect style set, `true` is returned.

Get:
```
virtual Boolean isAutoSelect () const;
```

Set:
```
virtual I3StateCheckBox& enableAutoSelect
(Boolean autoSelect=true);
```

autoSelect (Boolean) — ICheckBox

Description: If the check box control has the style autoSelect set, `true` is returned.

Get:
```
virtual Boolean isAutoSelect () const;
```

Set:
```
virtual ICheckBox& enableAutoSelect (Boolean autoSelect=true);
```

autoSelect (Boolean) — IRadioButton

Description: If the radio button control has the style autoSelect set, `true` is returned.

Get:
```
virtual Boolean isAutoSelect () const;
```

Set:
```
virtual IRadioButton& enableAutoSelect
(Boolean autoSelect=true);
```

autoTab (Boolean) — IEntryField

Description: Queries whether the style autoTab is set on an entry field control.

Get:
```
Boolean isAutoTab () const;
```

Set:
```
virtual IEntryField& enableAutoTab (Boolean autoTab=true);
```

available (Boolean) — ICLibErrorInfo

Description: If the error text is available, `true` is returned.

Get:
```
virtual Boolean isAvailable () const;
```

available (Boolean) — IGUIErrorInfo

Description: If the error information is available, `true` is returned.

Get:
```
virtual Boolean isAvailable () const;
```

available (Boolean) — IMMErrorInfo — vbmm

Description: If there is error text available for the error, returns `true`.

Get:
```
virtual Boolean isAvailable () const;
```

available (Boolean) — ISystemErrorInfo

Description: If the error information is available, `true` is returned.

Get:
```
virtual Boolean isAvailable () const;
```

avgCharWidth (unsigned long) — IFont

Description: Returns the average character's width.

Get:
```
unsigned long avgCharWidth () const;
```

backgroundColor (IColor) — IWindow

Description: Returns the background color value of the window area.

Get:
```
virtual IColor backgroundColor () const;
```

Set:
```
virtual IWindow& setBackgroundColor
(const IColor& backgroundColor);
```

ID:
```
backgroundColorId
```

balance (unsigned long) — IMMAmpMixer — vbmm

Description: Returns the balance, where 0 is defined as full left balance and 100 is defined as full right balance.

Get:
```
unsigned long balance (CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMAmpMixer& setBalance (unsigned long balance,
CallType call=IMMDevice::wait);
```

bass (unsigned long) — IMMAmpMixer — vbmm

Description: Returns the bass, where 0% is the least amount of bass and 100% is the most amount of bass.

Get:
```
unsigned long bass (CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMAmpMixer& setBass (unsigned long bass,
CallType call=IMMDevice::wait);
```

bidiSupported (Boolean) — IWindow

Description: Returns `true` if the window supports querying and setting of bidirectional attributes.

Get:
```
Boolean isBidiSupported () const;
```

binaryDigits (Boolean) — IString

Description: If all the characters are either 0 or 1, `true` is returned.

Get:
```
Boolean isBinaryDigits () const;
```

binding (INotebook::Binding) — INotebook

Description: Returns the type of binding used for the notebook.

Get:
```
virtual Binding binding () const;
```

Set:
```
virtual INotebook& setBinding (Binding binding);
```

bitmap (IBitmapHandle) — IAnimatedButton

Description: Returns the handle of the specifed bitmap.

Get:
```
virtual IBitmapHandle bitmap (unsigned long index=0) const;
```

bitmap (IBitmapHandle) — IBitmapControl

Description: Returns the handle to the bitmap.

Get:
```
IBitmapHandle bitmap () const;
```

Set:
```
virtual IBitmapControl& setBitmap
(ISystemBitmapHandle::Identifier bitmap);
```

bitmap (IBitmapHandle) — IClipboard

Description: If data of the format `IClipboard::bitmapFormat` exists on the clipboard, this function creates a copy of the bitmap and returns its IBitmapHandle.

Get:
```
virtual IBitmapHandle bitmap ();
```

Set:
```
virtual IClipboard& setBitmap (const IBitmapHandle& bitmap);
```

bitmap (Boolean) — IFont

Description: If the `IFont` object uses a bit-map font, `true` is returned.

Get:
```
Boolean isBitmap () const;
```

bitmap (IBitmapHandle) — IGraphicPushButton

Description: Returns the handle of the currently set bit map.

Get:
```
IBitmapHandle bitmap () const;
```

Set:
```
virtual IGraphicPushButton& setGraphic (unsigned long bitmap);
```

bitmap (IBitmapHandle) — IMenuItem

Description: Returns the bit-map handle of the item.

Get:
```
IBitmapHandle bitmap () const;
```

Set:
```
IMenuItem& setBitmap (unsigned long bitmap);
```

bitmap (IBitmapHandle) — IToolBarButton

Description: Returns the handle of the default bitmap.

Get:
```
IBitmapHandle bitmap () const;
```

Set:
```
virtual IToolBarButton& setBitmap
(const IBitmapHandle& bitmap);
```

bitmapCount (unsigned long) — IAnimatedButton

Description: Returns the number of bitmaps specified for the button.

Get:
```
virtual unsigned long bitmapCount () const;
```

bitmapOnly (Boolean) — IFont

Description: If the `IFont` object uses only bit-map fonts, `true` is returned.

Get:
```
Boolean isBitmapOnly () const;
```

bitmapSize (ISize) — IToolBarButton

Description: Returns the size of the current bitmap.

Get:
```
virtual ISize bitmapSize () const;
```

bitmapVisible (Boolean) — IToolBarButton

Description: Returns `true` if the bitmap is visible.

Get:
```
Boolean isBitmapVisible () const;
```

bitsPerSample (unsigned long) — IMMConfigurableAudio — vbmm

Description: Returns the number of bits-per-sample.

Get:
```
unsigned long bitsPerSample (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMConfigurableAudio& setBitsPerSample
(unsigned long bitsPerSample, CallType call=IMMDevice::wait);
```

blockAlignment (unsigned long) — IMMConfigurableAudio — vbmm

Description: Returns the block alignment of data in bytes.

Get:
```
unsigned long blockAlignment (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMConfigurableAudio& setBlockAlignment
(unsigned long blockAlignment, CallType call=IMMDevice::wait);
```

blueMix (unsigned char) — IColor

Description: Returns the value of the blue component.

Get:
```
unsigned char blueMix () const;
```

Set:
```
IColor& setBlue (unsigned char blueMix);
```

bold (Boolean) — IFont

Description: If the `IFont` object uses a font that is bold, `true` is returned.

Get:
```
Boolean isBold () const;
```

Set:
```
virtual IFont& setBold (Boolean bold=true);
```

border (Boolean) — IBaseSpinButton

Description: If the spin button has a border, `true` is returned.

Get:
```
Boolean hasBorder () const;
```

border (Boolean) — IPushButton

Description: If the push button has a border, returns `true`.

Get:
```
Boolean hasBorder () const;
```

Set:
```
virtual IPushButton& addBorder (Boolean border=true);
```

borderColor (IColor) — IWindow

Description: Returns the border color value of the window area.

Get:
```
virtual IColor borderColor () const;
```

Set:
```
virtual IWindow& setBorderColor (const IColor& borderColor);
```

ID:
```
borderColorId
```

borderHeight (unsigned long) — IFrameWindow

Description: Returns the height of the frame window's top and bottom borders.

Get:
```
unsigned long borderHeight () const;
```

Set:
```
IFrameWindow& setBorderHeight (unsigned long borderHeight);
```

borderSize (ISize) — IFrameWindow

Description: Returns the width of the frame window's left and right borders and the height of the frame window's top and bottom borders.

Get:
```
ISize borderSize () const;
```

Set:
```
IFrameWindow& setBorderSize (const ISize& borderSize);
```

borderWidth (unsigned long) — IFrameWindow

Description: Returns the width of the frame window's left and right borders.

Get:
```
unsigned long borderWidth () const;
```

Set:
```
IFrameWindow& setBorderWidth (unsigned long borderWidth);
```

bottom (IRectangle::Coord) — IRectangle

Description: Returns the Y-coordinate of the horizontal line that forms the bottom of the rectangle.

Get:
```
Coord bottom () const;
```

bottomCenter (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the bottom-center point of the rectangle.

Get:
```
IPoint bottomCenter () const;
```

bottomLeft (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the bottom-left corner of the rectangle.

Get:
```
IPoint bottomLeft () const;
```

Set:
```
IRectangle& moveTo (const IPoint& bottomLeft);
```

bottomRight (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the bottom-right corner of the rectangle.

Get:
```
IPoint bottomRight () const;
```

bounded

bounded (IBoolean) — IRBag — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRDeque — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IREqualitySequence — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRHeap — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRQueue — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRSequence

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRSet — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRSortedBag — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRSortedSet — vbcc

Description: Returns `True` if the collection is bounded.
Get: `IBoolean isBounded () const;`

bounded (IBoolean) — IRStack — vbcc

Description: Returns `True` if the collection is bounded.

Get:
```
IBoolean isBounded () const;
```

button2CnrItems (IVSequence<CnrElement*>*) — IVBContainerControl

Description: Returns a collection of the container objects that mouse button2 was clicked over.

Get:
```
virtual IVSequence<CnrElement*>* button2CnrObjects ();
```

ID: `button2PointId`

button2CollectionPosition (unsigned long) — IVBContainerControl

Description: Returns the collection position corresponding to the first mouse button2ed element.

Get:
```
virtual unsigned long button2CollectionPosition ();
```

ID: `button2PointId`

button2CollectionPositions (IVSequence<unsigned long>*) — IVBContainerControl

Description: Returns a collection of the indexes of all button2ed elements.

Get:
```
virtual IVSequence<unsigned long>* button2CollectionPositions
();
```

ID: `button2PointId`

button2Item (Element) — IVBContainerControl

Description: Returns the collection element corresponding to the first button2ed item in the container.

Get:
```
virtual Element button2Element ();
```

ID: `button2PointId`

button2Items (IVSequence<Element>*) — IVBContainerControl

Description: Returns a collection of the items that mouse button2 was clicked over.

Get:
```
virtual IVSequence<Element>* button2Elements ();
```

ID: `button2PointId`

button2Point (IPoint) — IVBContainerControl

Description: Returns the point where mouse button2 was clicked.

Get:
```
virtual IPoint button2Point () const;
```

ID:
```
button2PointId
```

buttonPressedId (long) — IVBFileDialog

Description: Returns the ID of the push button that was used to dismiss the dialog.

Get:
```
long buttonPressedId () const;
```

buttonPressedId (unsigned long) — IVBFontDialog

Description: Returns the ID of the push button that was used to dismiss the dialog.

Get:
```
unsigned long buttonPressedId () const;
```

buttonsPosition (ISlider::ButtonsPosition) — ISlider

Description: Returns the current position of the slider's buttons for this slider object.

Get:
```
ButtonsPosition buttonsPosition () const;
```

buttonView (IToolBarButton::View) — IToolBar

Description: Returns the current view of `IToolBarButton` objects in the tool bar.

Get:
```
IToolBarButton::View buttonView () const;
```

Set:
```
virtual IToolBar& setButtonView
(IToolBarButton::View buttonView);
```

bytesPerSecond (unsigned long) — IMMConfigurableAudio — vbmm

Description: Returns the average number of bytes-per-second played or recorded.

Get:
```
unsigned long bytesPerSecond (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMConfigurableAudio& setBytesPerSecond
(unsigned long bytesPerSecond, CallType call=IMMDevice::wait);
```

cachingEnabled (Boolean) — IContainerControl

Description: Queries whether a delta value is set and caching is enabled.

Get:
```
Boolean isCachingEnabled () const;
```

Set:
```
virtual IContainerControl& enableCaching
(unsigned long cachingEnabled=30);
```

center (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the center point of the rectangle.

Get:
```
IPoint center () const;
```

centerXCenterY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the center point of the rectangle.

Get:
```
IPoint centerXCenterY () const;
```

centerXMaxY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the center point of the horizontal line opposite the origin of the rectangle.

Get:
```
IPoint centerXMaxY () const;
```

centerXMinY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the center point of the horizontal line passing through the origin of the rectangle.

Get:
```
IPoint centerXMinY () const;
```

changed (Boolean) — IEntryField

Description: Queries whether the entry field text has changed since the last query.

Get:
```
Boolean hasChanged () const;
```

changed (Boolean) — IMultiLineEdit

Description: Queries whether any changes have been made to the MLE since the last time the changed flag was reset.

Get:
```
Boolean isChanged () const;
```

channels (unsigned long) — IMMConfigurableAudio — vbmm

Description: Returns the number of audio channels set.

Get:
```
unsigned long channels (CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMConfigurableAudio& setChannels
(unsigned long channels, CallType call=IMMDevice::wait);
```

character (IString) — IAcceleratorKey

Description: Returns the data key defined for this accelerator key.

Get:
```
IString character () const;
```

characterSize (ISize) — IWindow

Description: Calculates and returns the average character width and maximum character height for the currently set font.

Get:
```
ISize characterSize () const;
```

charType (IEntryField::CharType) — IEntryField

Description: Returns the current type of character that this entry field accepts.

Get:
```
CharType charType () const;
```

Set:
```
virtual IEntryField& setCharType (CharType charType);
```

checked (Boolean) — IMenuItem

Description: If the attribute checked is set, `true` is returned.

Get:
```
Boolean isChecked () const;
```

Set:
```
IMenuItem& setChecked (Boolean checked=true);
```

city (IString) — IAddress — vbsample

Description: Query the city (`IString`) attribute.

Get:
```
virtual IString city () const;
```

Set:
```
virtual IAddress& setCity (const IString& city);
```

ID:
```
cityId
```

client (IWindow*) — IFrameWindow

Description: Returns a pointer to the `IWindow` corresponding to the client window.

Get:
```
virtual IWindow* client () const;
```

Set:
```
virtual IFrameWindow& setClient (IWindow* client);
```

clientHandle (IWindowHandle) — IFrameWindow

Description: Returns the IWindowHandle of the client window. If no client window exists, a 0 handle is returned.

Get:
```
virtual IWindowHandle clientHandle () const;
```

clipboardHasTextFormat (Boolean) — ITextControl

Description: Returns `true` if the clipboard is in text format.

Get:
```
static Boolean clipboardHasTextFormat ();
```

closeOnDestroy (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device closes itself when the device object is deleted.

Get:
```
Boolean isCloseOnDestroy () const;
```

Set:
```
IMMDevice& setCloseOnDestroy (Boolean closeOnDestroy=false);
```

columnCount (unsigned long) — IContainerControl

Description: Returns the number of columns in the container.

Get:
```
unsigned long columnCount () const;
```

command (Boolean) — IEntryField

Description: Queries whether the style command is set on an entry field control.

Get:
```
Boolean isCommand () const;
```

Set:
```
virtual IEntryField& enableCommand (Boolean command=true);
```

ID: `commandId`

commandId (ICommand::CommandId) — IAcceleratorKey

Description: If the accelerator key runs an application or system command, this function returns the identifier of the command.

Get:
```
ICommand::CommandId commandId () const;
```

Set:
```
IAcceleratorKey& setCommand (ICommand::CommandId commandId);
```

commandType (IMenuItem::CommandType) — IMenuItem

Description: Returns the event the menu item generates when selected.

Get:
```
CommandType commandType () const;
```

Set:
```
IMenuItem& setCommand (CommandType commandType);
```

communicationWindow (IWindowHandle) — IHelpWindow

Description: Returns the handle of the active communication window.

Get:
```
IWindowHandle communicationWindow () const;
```

consistent (IBoolean) — IRBag — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRDeque — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IREqualitySequence — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRHeap — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRQueue — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRSequence

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRSet — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRSortedBag — vbcc

Get:
```
IBoolean isConsistent () const;
```

consistent (IBoolean) — IRSortedSet — vbcc

Get: `IBoolean isConsistent () const;`

consistent (IBoolean) — IRStack — vbcc

Get: `IBoolean isConsistent () const;`

contents (IMMAudioCDContents) — IMMAudioCD — vbmm

Description: Returns a copy of the CD's table of contents.
Get: `IMMAudioCDContents contents () const;`

contents (IString) — IString

Description: Returns the string's contents.
Get: `IString asString () const;`
Set: `IString& operator= (const IString& aString);`

contentsWindow (IWindowHandle) — IHelpWindow

Description: Returns the handle of the Table of Contents window.
Get: `IWindowHandle contentsWindow () const;`

continuousPlayEnabled (Boolean) — IMMAudioCD — vbmm

Description: Returns `true`, if continuous play is turned on.
Get: `Boolean isContinuousPlayEnabled () const;`
Set:
```
IMMAudioCD& enableContinuousPlay
(Boolean continuousPlayEnabled=true);
```

control (Boolean) — IString

Description: If all the characters are in {0x00-0x1F,0x7F}, `true` is returned.
Get: `Boolean isControl () const;`

coord1 (IPair::Coord) — IPair

Description: Obtains the value of the first coordinate.
Get: `Coord coord1 () const;`
Set: `IPair& setCoord1 (Coord coord1);`

coord2 •currentEditObject

coord2 (IPair::Coord) — IPair

Description: Obtains the value of the second coordinate.
Get:
```
Coord coord2 () const;
```
Set:
```
IPair& setCoord2 (Coord coord2);
```

count (unsigned long) — IBaseComboBox

Description: Returns the number of items in the list box.
Get:
```
virtual unsigned long count () const;
```

count (unsigned long) — IBaseListBox

Description: Returns the count of the number of items in the list box.
Get:
```
virtual unsigned long count () const;
```

coverPageWindow (IWindowHandle) — IHelpWindow

Description: Returns the handle of the Information Presentation Facility (IPF) multiple document interface parent window.
Get:
```
IWindowHandle coverPageWindow () const;
```

currentBitmapIndex (unsigned long) — IAnimatedButton

Description: Returns the index of the bitmap that is currently displayed for the button.
Get:
```
virtual unsigned long currentBitmapIndex () const;
```
Set:
```
virtual IAnimatedButton& setCurrentBitmapIndex
(unsigned long index=0);
```

currentEditColumn (IContainerColumn*) — IContainerControl

Description: Retrieves the column being edited.
Get:
```
IContainerColumn* currentEditColumn () const;
```

currentEditMLE (IMultiLineEdit*) — IContainerControl

Description: Retrieves the multiple-line edit (MLE) field being used for editing.
Get:
```
IMultiLineEdit* currentEditMLE () const;
```

currentEditObject (IContainerObject*) — IContainerControl

Description: Retrieves the object being edited.
Get:
```
IContainerObject* currentEditObject () const;
```

currentGraphicType (IGraphicPushButton::GraphicType) — IGraphicPushButton

Description: Returns the current type of graphic set onto the graphic push button.

Get:
```
GraphicType currentGraphicType () const;
```

cursoredCollectionPosition (unsigned long) — IVBContainerControl

Description: Returns the collection position corresponding to the cursored element.

Get:
```
virtual unsigned long cursoredCollectionPosition ();
```

cursoredItem (Element) — IVBContainerControl

Description: Returns the item corresponding to the cursored item in the container.

Get:
```
virtual Element button2Element ();
```

cursoredObject (IContainerObject*) — IContainerControl

Description: Returns the object on which the cursor is located.

Get:
```
virtual IContainerObject* cursoredObject () const;
```

cursorLinePosition (unsigned long) — IMultiLineEdit

Description: Returns the line number of the line that contains the cursor.

Get:
```
unsigned long cursorLinePosition () const;
```

Set:
```
virtual IMultiLineEdit& setCursorLinePosition
(unsigned long cursorLinePosition);
```

cursorPosition (unsigned long) — IEntryField

Description: Returns the character position from the start of the entry field to the current cursor location.

Get:
```
unsigned long cursorPosition () const;
```

Set:
```
virtual IEntryField& setCursorPosition
(unsigned long cursorPosition);
```

cursorPosition (unsigned long) — IMultiLineEdit

Description: Returns the character position from the start of the MLE to the current cursor location.

Get:
```
unsigned long cursorPosition () const;
```

Set:
```
virtual IMultiLineEdit& setCursorPosition
(unsigned long cursorPosition);
```

cursorSelect (Boolean) — IRadioButton

Description: If the radio button is cursor-selectable, `true` is returned.

Get:
```
Boolean isCursorSelect () const;
```

Set:
```
virtual IRadioButton& enableCursorSelect
(Boolean cursorSelect=true);
```

customerList (IVSequence<ICustomer*>*) — ICompany — vbsample

Description: Query the customerList (`IVSequence <ICustomer*> *`) attribute.

Get:
```
virtual IVSequence<ICustomer*>* customerList () const;
```

Set:
```
virtual ICompany& setCustomerList
(const IVSequence<ICustomer*>& customerList);
```

ID:
```
customerListId
```

dataIString (Boolean) — ICnrEditHandler

Description: If the user data is type `IString`, `true` is returned.

Get:
```
Boolean isDataIString () const;
```

date (Boolean) — IContainerColumn

Description: If the data in the column is a date, `true` is returned.

Get:
```
Boolean isDate () const;
```

date (IDate) — ICustomer — vbsample

Description: Query the date (IDate) attribute.
Get:
```
virtual IDate date () const;
```
Set:
```
virtual ICustomer& setDate (const IDate& date);
```
ID:
```
dateId
```

dayOfMonth (int) — IDate

Description: Returns the day in the receiver's month as an integer from 1 to 31.
Get:
```
int dayOfMonth () const;
```

dayOfWeek (IDate::DayOfWeek) — IDate

Description: Returns the index of the receiver's day of the week: Monday through Sunday.
Get:
```
DayOfWeek dayOfWeek () const;
```

dayOfYear (int) — IDate

Description: Returns the day in the receiver's year as an integer from 1 to 366.
Get:
```
int dayOfYear () const;
```

deckCount (unsigned long) — ISetCanvas

Description: Returns the maximum number of decks used by the canvas.
Get:
```
unsigned long deckCount () const;
```
Set:
```
virtual ISetCanvas& setDeckCount (unsigned long deckCount);
```

deckOrientation (ISetCanvas::DeckOrientation) — ISetCanvas

Description: Returns an enumerator for deck direction.
Get:
```
DeckOrientation deckOrientation () const;
```
Set:
```
virtual ISetCanvas& setDeckOrientation
(DeckOrientation deckOrientation);
```
ID:
```
deckOrientationId
```

default (Boolean) — IPushButton

Description: If the style defaultButton is set, returns true.

Get:
```
Boolean isDefault () const;
```

Set:
```
virtual IPushButton& enableDefault (Boolean default=true);
```

defaultApplicationName (IString) — IProfile

Description: Returns the default application name.

Get:
```
virtual IString defaultApplicationName () const;
```

Set:
```
virtual IProfile& setDefaultApplicationName
(const char* defaultApplicationName);
```

defaultCell (ISize) — IMultiCellCanvas

Description: Queries the current default cell size.

Get:
```
static ISize defaultCell ();
```

Set:
```
static void setDefaultCell (const ISize& defaultCell);
```

defaultGroupPad (unsigned long) — IToolBar

Description: Returns the current number of pels of padding added between groups in tool bars.

Get:
```
static unsigned long defaultGroupPad ();
```

Set:
```
static void setDefaultGroupPad
(unsigned long defaultGroupPad=8);
```

defaultInput (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the defaultInput (Boolean) attribute.

Get:
```
virtual Boolean defaultInput () const;
```

defaultInput (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the defaultInput (Boolean) attribute.

Get:
```
virtual Boolean defaultInput () const;
```

defaultMargin (ISize) — IToolBar

Description: Returns the current number of pels of margin added around windows in tool bars.

Get:
```
static ISize defaultMargin ();
```

Set:
```
static void setDefaultMargin (const ISize& defaultMargin=ISize
( 7 , 4 ));
```

defaultMisfitWidth (unsigned long) — IToolBar

Description: Returns the current width used to filter oversized windows from tool bars.

Get:
```
static unsigned long defaultMisfitWidth ();
```

Set:
```
static void setDefaultMisfitWidth
(unsigned long defaultMisfitWidth=100);
```

defaultOrdering (IWindow::SiblingOrder) — IFrameWindow

Description: Returns the order in which new windows are created relative to their sibling windows.

Get:
```
static IWindow::SiblingOrder defaultOrdering ();
```

Set:
```
static void setDefaultOrdering (SiblingOrder defaultOrdering);
```

defaultPad (long) — IToolBar

Description: Returns the current default number of pels use to separate windows in tool bars.

Get:
```
static long defaultPad ();
```

Set:
```
static void setDefaultPad (long defaultPad=-1);
```

defaultPushButton (IWindowHandle) — IWindow

Description: Returns the first child window that is a default push button, if one exists.

Get:
```
virtual IWindowHandle defaultPushButton () const;
```

defaultText (IString) — IFlyOverHelpHandler

Description: Returns the text to display when an information string is not found for the `ITextControl`.

Get:
```
virtual IString defaultText () const;
```

Set:
```
virtual IFlyOverHelpHandler& setDefaultText
(unsigned long defaultText);
```

defaultText (IString) — IVBStringPart — vbsample

Description: Query the defaultText (IString) attribute.

Get:
```
virtual IString defaultText () const;
```

Set:
```
virtual IVBStringPart& setDefaultText
(const IString& defaultText=false);
```

ID:
```
defaultTextId
```

defaultTransparentColor (IColor) — IToolBarButton

Description: Returns the current default transparent color.

Get:
```
static IColor defaultTransparentColor ();
```

Set:
```
static void setDefaultTransparentColor
(const IColor& defaultTransparentColor=IColor ( IColor::pink
));
```

defaultTransparentColorSet (Boolean) — IToolBarButton

Description: Returns true if a default transparent color is set.

Get:
```
static Boolean isDefaultTransparentColorSet ();
```

defaultValue (Boolean) — IVBBooleanPart — vbsample

Description: Query the defaultValue (Boolean) attribute.

Get:
```
virtual Boolean defaultValue () const;
```

Set:
```
virtual IVBBooleanPart& setDefaultValue
(Boolean defaultValue=false);
```

ID:
```
defaultValueId
```

defaultValue (double) — IVBDoublePart — vbsample

Description: Query the defaultValue (double) attribute.

Get:
```
virtual double defaultValue () const;
```

Set:
```
virtual IVBDoublePart& setDefaultValue
(double defaultValue=0);
```

ID:
```
defaultValueId
```

defaultValue (long) — IVBLongPart — vbsample

Description: Query the defaultValue (long) attribute.

Get:
```
virtual long defaultValue () const;
```

Set:
```
virtual IVBLongPart& setDefaultValue (long defaultValue=0);
```

ID:
```
defaultValueId
```

defaultValue (short) — IVBShortPart — vbsample

Description: Query the defaultValue (short) attribute.

Get:
```
virtual short defaultValue () const;
```

Set:
```
virtual IVBShortPart& setDefaultValue (short defaultValue=0);
```

ID:
```
defaultValueId
```

defaultValue (unsigned long) — IVBUnsignedLongPart — vbsample

Description: Query the defaultValue (unsigned long) attribute.

Get:
```
virtual unsigned long defaultValue () const;
```

Set:
```
virtual IVBUnsignedLongPart& setDefaultValue
(unsigned long defaultValue=0);
```

ID:
```
defaultValueId
```

defaultValue (unsigned short) — IVBUnsignedShortPart — vbsample

Description: Query the defaultValue (unsigned short) attribute.

Get:
```
virtual unsigned short defaultValue () const;
```

Set:
```
virtual IVBUnsignedShortPart& setDefaultValue
(unsigned short defaultValue=0);
```

ID:
```
defaultValueId
```

delayTime (unsigned long) — IFlyOverHelpHandler

Description: Returns the minimum time, in milliseconds, the mouse must remain in the same position before help text is displayed.

Get:
```
virtual unsigned long delayTime () const;
```

Set:
```
virtual IFlyOverHelpHandler& setDelayTime
(unsigned long delayTime=100);
```

description (IString) — IMMDevice — vbmm

Description: Returns the description of the hardware associated with this device.

Get:
```
IString description (CallType call=IMMDevice::wait) const;
```

destinationRectangle (IRectangle) — IMMDigitalVideo — vbmm

Description: Returns the subrectangle within the video window where video is played.

Get:
```
IRectangle destinationRectangle
(CallType call=IMMDevice::wait) const;
```

Set:
```
IMMDigitalVideo& setDestination (const IRectangle& rectangle,
CallType call=IMMDevice::wait);
```

detailsTitleRectangle (IRectangle) — IContainerControl

Description: Returns the details view, column-title rectangle in container window coordinates.

Get:
```
IRectangle detailsTitleRectangle (Boolean rightSide=false)
const;
```

detailsView (Boolean) — IContainerControl

Description: Queries whether the container is currently in the details view.

Get:
```
Boolean isDetailsView () const;
```

detailsViewPortOnWindow (IRectangle) — IContainerControl

Description: Retrieves either the right or left work area of a split details view, in window coordinates.

Get:
```
IRectangle detailsViewPortOnWindow (Boolean rightSide=false)
const;
```

detailsViewPortOnWorkspace (IRectangle) — IContainerControl

Description: Retrieves either the right or left work area of a split details view, in workspace coordinates.

Get:
```
IRectangle detailsViewPortOnWorkspace
(Boolean rightSide=false) const;
```

detailsViewSplit (IContainerColumn*) — IContainerControl

Description: Returns the last column before the split bar.

Get:
```
IContainerColumn* detailsViewSplit () const;
```

Set:
```
virtual IContainerControl& setDetailsViewSplit
(const IContainerColumn* detailsViewSplit,
unsigned long pixelsFromLeft=50);
```

deviceId (unsigned long) — IMMDevice — vbmm

Description: Returns the ID of the current device.
Get:
```
unsigned long deviceId () const;
```

deviceName (IString) — IMMDevice — vbmm

Description: Returns the name of the device depending on the class of the object.
Get:
```
IString deviceName () const;
```

deviceType (unsigned long) — IMMDevice — vbmm

Description: Returns the type of the device.
Get:
```
unsigned long deviceType (CallType call=IMMDevice::wait)
const;
```

deviceType (unsigned long) — IMMPlayerPanel — vbmm

Description: Returns the type of device this player panel was created for.
Get:
```
unsigned long deviceType ();
```

digits (Boolean) — IString

Description: If all the characters are in {'0'-'9'}, `true` is returned.
Get:
```
Boolean isDigits () const;
```

digits (Boolean) — IVBStringPart — vbsample

Description: Query the digits (`Boolean`) attribute.
Get:
```
virtual Boolean isDigits () const;
```
ID:
```
textId
```

disabled (Boolean) — IMenuItem

Description: If the attribute disabled is set, `true` is returned.
Get:
```
Boolean isDisabled () const;
```
Set:
```
IMenuItem& setDisabled (Boolean disabled=true);
```

disabledBackgroundColor (IColor) — IWindow

Description: Returns the disabled background color value of the window area.

Get:
```
virtual IColor disabledBackgroundColor () const;
```

Set:
```
virtual IWindow& setDisabledBackgroundColor
(const IColor& disabledBackgroundColor);
```

ID:
```
disabledBackgroundColorId
```

disabledForegroundColor (IColor) — IWindow

Description: Returns the disabled foreground color value of the window area.

Get:
```
virtual IColor disabledForegroundColor () const;
```

Set:
```
virtual IWindow& setDisabledForegroundColor
(const IColor& disabledForegroundColor);
```

ID:
```
disabledForegroundColorId
```

disabledText (IString) — IInfoArea

Description: Returns the text displayed when the menu item at the selection cursor is disabled.

Get:
```
virtual IString disabledText () const;
```

Set:
```
virtual IInfoArea& setDisabledText
(const IString& disabledText);
```

ID:
```
disabledTextId
```

discId (IString) — IMMAudioCD — vbmm

Description: Returns the identifier of the current disc.

Get:
```
IString discId (CallType call=IMMDevice::wait) const;
```

discId (IString) — IMMAudioCDContents — vbmm

Description: Returns the compact disc identifier.

Get:
```
IString discId () const;
```

discTitle (IString) — IMMAudioCD — vbmm

Description: Returns the disc title.

Get:
```
IString discTitle () const;
```

Set:
```
IMMAudioCD& setDiscTitle (const IString& discTitle);
```

displaySize (ISize) — ITextControl

Description: Returns the width and height of the rectangle enclosing the string.

Get:
```
virtual ISize displaySize (const char* text=0) const;
```

displayWidth (unsigned long) — IContainerColumn

Description: Returns the displayable width of the column, in pixels.

Get:
```
virtual unsigned long displayWidth ();
```

Set:
```
virtual IContainerColumn& setDisplayWidth
(unsigned long displayWidth);
```

dragLines (Boolean) — IMultiCellCanvas

Description: Queries whether a multiple-cell canvas has the style dragLines set.

Get:
```
Boolean hasDragLines () const;
```

Set:
```
virtual IMultiCellCanvas& enableDragLines
(Boolean dragLines=true);
```

drawBackgroundEnabled (Boolean) — IContainerControl

Description: Queries whether owner draw of the container background is enabled.

Get:
```
Boolean isDrawBackgroundEnabled () const;
```

Set:
```
virtual IContainerControl& enableDrawBackground
(Boolean drawBackgroundEnabled=true);
```

drawItem (Boolean) — IBaseListBox

Description: If the style drawItem is set, `true` is returned.

Get:
```
Boolean isDrawItem () const;
```

Set:
```
virtual IBaseListBox& enableDrawItem (Boolean drawItem=true);
```

drawItem (Boolean) — IMenuItem

Description: If the style drawItem is set, returns `true`.

Get:
```
Boolean isDrawItem () const;
```

Set:
```
IMenuItem& setDrawItem (Boolean drawItem=true);
```

drawItemEnabled (Boolean) — IContainerControl

Description: Queries whether owner draw of container objects is enabled.

Get:
```
Boolean isDrawItemEnabled () const;
```

Set:
```
virtual IContainerControl& enableDrawItem
(Boolean drawItemEnabled=true);
```

drawItemEnabled (Boolean) — IProgressIndicator

Description: If the handleDrawItem style is set, `true` is returned.

Get:
```
Boolean isDrawItemEnabled () const;
```

Set:
```
virtual IProgressIndicator& enableDrawItem
(Boolean drawItemEnabled=true);
```

dropOnAble (Boolean) — IContainerObject

Description: If this object can accept a drop, `true` is returned.

Get:
```
virtual Boolean isDropOnAble (IContainerControl* container=0)
const;
```

Set:
```
virtual IContainerObject& enableDrop (Boolean dropOnAble=true,
IContainerControl* container=0);
```

editRegionHeight (unsigned long) — IMultiLineEdit

Description: Returns the height of the edit region.

Get:
```
unsigned long editRegionHeight () const;
```

Set:
```
virtual IMultiLineEdit& setEditRegionHeight
(long editRegionHeight);
```

editRegionWidth (unsigned long) — IMultiLineEdit

Description: Returns the width of the edit region.

Get:
```
unsigned long editRegionWidth () const;
```

Set:
```
virtual IMultiLineEdit& setEditRegionWidth
(long editRegionWidth);
```

empty (Boolean) — IBaseListBox

Description: If the list box is empty, `true` is returned.

Get:
```
virtual Boolean isEmpty () const;
```

empty (Boolean) — IEntryField

Description: Queries whether the entry field is empty.

Get:
```
Boolean isEmpty () const;
```

empty (Boolean) — INotebook

Description: Queries whether the notebook has any pages, and returns `true` if it is empty.

Get:
```
virtual Boolean isEmpty () const;
```

empty (IBoolean) — IRBag — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRDeque — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IREqualitySequence — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRHeap — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRQueue — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRSequence

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRSet — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRSortedBag — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRSortedSet — vbcc

Description: Returns `True` if the collection is empty.
Get: `IBoolean isEmpty () const;`

empty (IBoolean) — IRStack — vbcc

Description: Returns `True` if the collection is empty.
Get:
```
IBoolean isEmpty () const;
```

enabled (Boolean) — IHandler

Description: Returns whether the handler is currently enabled.
Get:
```
Boolean isEnabled () const;
```
Set:
```
virtual IHandler& enable (Boolean enabled=true);
```

enabled (Boolean) — IInfoArea

Description: If the window is sent mouse and keyboard input, `true` is returned.
Get:
```
Boolean IWindow::isEnabled () const;
```
Set:
```
virtual IWindow& IWindow::enable (Boolean enabled=true);
```
ID:
```
enableId
```

enabled (Boolean) — IWindow

Description: If the window is sent mouse and keyboard input, `true` is returned.
Get:
```
Boolean isEnabled () const;
```
Set:
```
virtual IWindow& enable (Boolean enabled=true);
```
ID:
```
enableId
```

enabledForNotification (Boolean) — IRBag — vbcc

Description: Returns `true` if an object is sending notifications to its observers.
Get:
```
virtual Boolean isEnabledForNotification () const;
```
Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRDeque — vbcc

Description: Returns `true` if an object is sending notifications to its observers.
Get:
```
virtual Boolean isEnabledForNotification () const;
```
Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IREqualitySequence — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRHeap — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRQueue — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRSequence

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRSet — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRSortedBag — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRSortedSet — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IRStack — vbcc

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IStandardNotifier

Description: Returns `true` if an object is sending notifications to its observers.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IStandardNotifier& enableNotification
(Boolean enabledForNotification=true);
```

enabledForNotification (Boolean) — IWindow

Description: Returns `true` if the window is sending notifications to observer objects.

Get:
```
virtual Boolean isEnabledForNotification () const;
```

Set:
```
virtual IWindow& enableNotification
(Boolean enabledForNotification=true);
```

errorId (unsigned long) — ICLibErrorInfo

Description: Returns the value of errno, which you can use to obtain the errno information.

Get:
```
virtual unsigned long errorId () const;
```

errorId (unsigned long) — IGUIErrorInfo

Description: Returns the error ID.

Get:
```
virtual unsigned long errorId () const;
```

errorId (unsigned long) — IMMErrorInfo — vbmm

Description: Returns the error ID.

Get:
```
virtual unsigned long errorId () const;
```

errorId (unsigned long) — ISystemErrorInfo

Description: Returns the error ID.
Get:
```
virtual unsigned long errorId () const;
```

expanded (Boolean) — IToolBar

Description: Returns `true` if the entire tool bar is displayed when it is floating.
Get:
```
Boolean isExpanded () const;
```

extendedSelect (Boolean) — IBaseListBox

Description: If the style extendedSelect is set, `true` is returned.
Get:
```
Boolean isExtendedSelect () const;
```
Set:
```
virtual IBaseListBox& enableExtendedSelect
(Boolean extendedSelect=true);
```

extendedSelection (Boolean) — IContainerControl

Description: Queries whether the container is in extended selection mode.
Get:
```
Boolean isExtendedSelection () const;
```

externalLeading (unsigned long) — IFont

Description: Returns the amount of white space that should appear between adjacent rows of text.
Get:
```
unsigned long externalLeading () const;
```

fastForwardButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the fast forward animated button.
Get:
```
IAnimatedButton* fastForwardButton () const;
```

fastSpinEnabled (Boolean) — IBaseSpinButton

Description: If the spin speed is doubled every two seconds, `true` is returned.
Get:
```
Boolean isFastSpinEnabled () const;
```
Set:
```
virtual IBaseSpinButton& enableFastSpin
(Boolean fastSpinEnabled=true);
```

fileDialog (IFileDialog*) — IVBFileDialog

Description: Returns a pointer to the `IFileDialog` instance.
Get:
```
IFileDialog* fileDialog ();
```

fileDialogSettings (IFileDialog::Settings*) — IVBFileDialog

Description: Returns a pointer to the IFileDialog::Settings instance.

Get:
```
IFileDialog::Settings* fileDialogSettings ();
```

filename (IString) — IMMFileMedia — vbmm

Description: Returns the currently loaded filename.

Get:
```
IString filename (CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMFileMedia& load (const IString& filename,
Boolean readOnly=false, CallType call=IMMDevice::wait);
```

fileName (IString) — IResourceLibrary

Description: Returns the fully qualified file name of the resource library.

Get:
```
virtual IString fileName () const;
```

fileName (IString) — IVBFileDialog

Description: Returns the fully qualified file name selected by the user.

Get:
```
IString fileName (unsigned long fileNumber=0) const;
```

Set:
```
virtual IVBFileDialog& setFileName (const char* fileName);
```

ID:
```
pressedOkId
```

fileNormalSpeed (IMMSpeed) — IMMDigitalVideo — vbmm

Description: Returns the normal play rate of the currently loaded video file, in the current speed format, either as a percentage or in frames-per-second.

Get:
```
IMMSpeed fileNormalSpeed (CallType call=IMMDevice::wait)
const;
```

fillBackground (Boolean) — IStaticText

Description: If the background is erased before any text is drawn, `true` is returned.

Get:
```
Boolean hasFillBackground () const;
```

Set:
```
virtual IStaticText& enableFillBackground
(Boolean fillBackground=true);
```

ID: `fillBackgroundId`

fillColor (IColor) — IStaticText

Description: Returns the fill color value of the static text.

Get:
```
virtual IColor fillColor () const;
```

Set:
```
virtual IStaticText& setFillColor (const IColor& fillColor);
```

ID: `fillColorId`

firstElement (Element const) — IRDeque — vbcc

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstElement (Element const) — IREqualitySequence — vbcc

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstElement (Element const) — IRQueue — vbcc

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstElement (Element const) — IRSequence

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstElement (Element const) — IRSortedBag — vbcc

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstElement (Element const) — IRSortedSet — vbcc

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstElement (Element const) — IRStack — vbcc

Description: Returns a reference to the first element of the collection.

Get:
```
Element const& firstElement () const;
```

firstPage (IPageHandle) — INotebook

Description: Returns an IPageHandle object that references the first page in the notebook.

Get:
```
virtual IPageHandle firstPage () const;
```

fixed (Boolean) — IFont

Description: If the `IFont` object uses a fixed-size (that is, non-proportional) font, `true` is returned.

Get:
```
Boolean isFixed () const;
```

flashing (Boolean) — IFrameWindow

Description: If the frame window is flashing, `true` is returned.

Get:
```
Boolean isFlashing () const;
```

floatingFrame (IToolBarFrameWindow*) — IToolBar

Description: Returns the IToolBarFrameWindow object that contains this tool bar.

Get:
```
virtual IToolBarFrameWindow* floatingFrame ();
```

floatingPosition (IPoint) — IToolBar

Description: Returns the current position of the frame window that encloses the floating tool bar (relative to the owning frame window used to construct the tool bar).

Get:
```
IPoint floatingPosition () const;
```

Set:
```
virtual IToolBar& setFloatingPosition
(const IPoint& floatingPosition);
```

floatingTitle (IString) — IToolBar

Description: Returns the title text of the floating frame window.
Get:
```
IString floatingTitle () const;
```
Set:
```
virtual IToolBar& setFloatingTitle
(const char* floatingTitle);
```

flowed (Boolean) — IContainerControl

Description: Queries whether the current container view is flowed.
Get:
```
Boolean isFlowed () const;
```

flowedNameView (Boolean) — IContainerControl

Description: Returns `true` if the container is currently in flowed name view.
Get:
```
Boolean isFlowedNameView () const;
```

flowedTextView (Boolean) — IContainerControl

Description: Returns `true` if the container is currently in flowed text view.
Get:
```
Boolean isFlowedTextView () const;
```

flyOverHelp (Boolean) — IVBFlyText

Description: If the `IFlyOverHelpHandler` is enabled, `true` is returned.
Get:
```
Boolean isFlyOverHelp () const;
```
Set:
```
IVBFlyText& enableFlyOverHelp (Boolean flyOverHelp=true);
```

flyOverHelpHandler (IFlyOverHelpHandler*) — IVBFlyText

Description: Returns the `IFlyOverHelpHandler`.
Get:
```
IFlyOverHelpHandler* flyOverHelpHandler ();
```

flyTextControl (IFlyText*) — IFlyOverHelpHandler

Description: Returns a pointer to the fly text control that is used to display fly help text.
Get:
```
virtual IFlyText* flyTextControl () const;
```
Set:
```
virtual IFlyOverHelpHandler& setFlyTextControl
(IFlyText* flyTextControl);
```

flyTextStringTableOffset (long) — IFlyOverHelpHandler

Description: Returns the offset currently in use for the fly text control.

Get:
```
virtual long flyTextStringTableOffset () const;
```

Set:
```
virtual IFlyOverHelpHandler& setFlyTextStringTableOffset
(long flyTextStringTableOffset=0);
```

focus (Boolean) — IWindow

Description: If the window has the input focus, `true` is returned.

Get:
```
Boolean hasFocus () const;
```

ID:
```
focusId
```

font (IFont) — IVBFontDialog

Description: Returns the font attribute.

Get:
```
IFont font () const;
```

Set:
```
virtual IVBFontDialog& setFont (const IFont& font);
```

ID:
```
pressedOkId
```

font (IFont) — IWindow

Description: Returns the font used by the window.

Get:
```
virtual IFont font () const;
```

Set:
```
virtual IWindow& setFont (const IFont& font);
```

ID:
```
fontId
```

fontDialog (IFontDialog*) — IVBFontDialog

Description: Returns a pointer to the `IFontDialog` object.

Get:
```
IFontDialog* fontDialog ();
```

fontDialogSettings (IFontDialog::Settings*) — IVBFontDialog

Description: Returns a pointer to the `IFontDialog::Settings` object.

Get:
```
IFontDialog::Settings* fontDialogSettings ();
```

foregroundColor (IColor) — IWindow

Description: Returns the foreground color value of the window area, or the default if no color for the area has been set.

Get:
```
virtual IColor foregroundColor () const;
```

Set:
```
virtual IWindow& setForegroundColor
(const IColor& foregroundColor);
```

ID: `foregroundColorId`

format (IMMAudioBuffer::Format) — IMMConfigurableAudio — vbmm

Description: Returns the interpretation of the audio format.

Get:
```
IMMAudioBuffer::Format format (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMConfigurableAudio& setFormat
(IMMAudioBuffer::Format format=IMMAudioBuffer::pcm,
CallType call=IMMDevice::wait);
```

format (IMMSpeed::Format) — IMMSpeed — vbmm

Description: Returns the current speed format.

Get:
```
Format format () const;
```

formatCount (unsigned long) — IClipboard

Description: Returns the number of clipboard formats in the clipboard.

Get:
```
unsigned long formatCount () const;
```

framed (Boolean) — IMenuItem

Description: If the attribute framed is set, `true` is returned.

Get:
```
Boolean isFramed () const;
```

Set:
```
IMenuItem& setFramed (Boolean framed=true);
```

frames (unsigned long) — IMMHourMinSecFrameTime — vbmm

Description: Returns the frames component of the time.

Get:
```
virtual unsigned long frames () const;
```

frames (unsigned long) — IMMMinSecFrameTime — vbmm

Description: Returns the frames component of the time.

Get:
```
unsigned long frames () const;
```

frames (unsigned long) — IMMTrackMinSecFrameTime — vbmm

Description: Returns the frames component of the time.
Get: `virtual unsigned long frames () const;`

framesPerSecond (unsigned long) — IMMHourMinSecFrameTime — vbmm

Description: Returns the number of frames per second.
Get: `unsigned long framesPerSecond () const;`

frameWindow (Boolean) — IWindow

Description: If this object represents a frame window, `true` is returned.
Get: `virtual Boolean isFrameWindow () const;`

full (IBoolean) — IRBag — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.
Get: `IBoolean isFull () const;`

full (IBoolean) — IRDeque — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.
Get: `IBoolean isFull () const;`

full (IBoolean) — IREqualitySequence — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.
Get: `IBoolean isFull () const;`

full (IBoolean) — IRHeap — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.
Get: `IBoolean isFull () const;`

full (IBoolean) — IRQueue — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.
Get: `IBoolean isFull () const;`

full (IBoolean) — IRSequence

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.

Get:
```
IBoolean isFull () const;
```

full (IBoolean) — IRSet — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.

Get:
```
IBoolean isFull () const;
```

full (IBoolean) — IRSortedBag — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.

Get:
```
IBoolean isFull () const;
```

full (IBoolean) — IRSortedSet — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.

Get:
```
IBoolean isFull () const;
```

full (IBoolean) — IRStack — vbcc

Description: Returns `True` if the collection is bounded and contains the maximum number of elements.

Get:
```
IBoolean isFull () const;
```

gain (unsigned long) — IMMAmpMixer — vbmm

Description: Returns the gain, where 0% is the least amount of gain and 100% is the most amount of gain..

Get:
```
unsigned long gain (CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMAmpMixer& setGain (unsigned long gain,
CallType call=IMMDevice::wait);
```

graphicContext (IGraphicContext*) — IDrawingCanvas

Description: Returns a pointer to the graphic context that the drawing canvas uses for paint events.

Get:
```
virtual IGraphicContext* graphicContext () const;
```

Set:
```
virtual IDrawingCanvas& setGraphicContext
(IGraphicContext* graphicContext);
```

graphicList (IGList*) — IDrawingCanvas

Description: Returns a pointer to the IGList object set for the drawing canvas.

Get:
```
virtual IGList* graphicList () const;
```

Set:
```
virtual IDrawingCanvas& setGraphicList (IGList* graphicList);
```

graphics (Boolean) — IString

Description: If all the characters are in {0x21-0x7E}, `true` is returned.

Get:
```
Boolean isGraphics () const;
```

graphicWindow (IIconControl) — IGraphicPushButton

Description: Returns the `IIconControl` used for drawing the graphic.

Get:
```
IIconControl& graphicWindow () const;
```

greenMix (unsigned char) — IColor

Description: Returns the value of the green component.

Get:
```
unsigned char greenMix () const;
```

Set:
```
IColor& setGreen (unsigned char greenMix);
```

gridLines (Boolean) — IMultiCellCanvas

Description: Queries whether a multiple-cell canvas has the style gridLines set.

Get:
```
Boolean hasGridLines () const;
```

Set:
```
virtual IMultiCellCanvas& enableGridLines
(Boolean gridLines=true);
```

group (Boolean) — IControl

Description: If the control has the group style set, `true` is returned.

Get:
```
virtual Boolean isGroup () const;
```

Set:
```
virtual IControl& enableGroup (Boolean group=true);
```

group (Boolean) — IWindow

Description: If the control has the group style set, `true` is returned.

Get:
```
virtual Boolean isGroup () const;
```

groupPad (unsigned long) — ISetCanvas

Description: Returns the pad width or height, which is the space before a child window in a deck with a style of `IControl::group`.

Get:
```
unsigned long groupPad () const;
```

Set:
```
virtual ISetCanvas& setGroupPad (unsigned long groupPad);
```

halftone (Boolean) — I3StateCheckBox

Description: If the three-state check box control has been selected as halftone, `true` is returned.

Get:
```
Boolean isHalftone () const;
```

halftone (Boolean) — IStaticText

Description: Queries whether the text has the style halftone.

Get:
```
Boolean isHalftone () const;
```

Set:
```
virtual IStaticText& enableHalftone (Boolean halftone=true);
```

ID:
```
halftoneId
```

handle (IAccelTblHandle) — IAccelerator

Description: Returns the lAccelTblHandle of the accelerator in effect.

Get:
```
IAccelTblHandle handle () const;
```

handle (IWindowHandle) — IMMDigitalVideo — vbmm

Description: Returns the window handle for the current video playback window.

Get:
```
IWindowHandle handle (CallType call=IMMDevice::wait) const;
```

handle (IModuleHandle) — IResourceLibrary

Description: Returns the handle used to load resources.

Get:
```
virtual IModuleHandle handle () const;
```

handle (IWindowHandle) — IWindow

Description: Returns the window handle.

Get:
```
virtual IWindowHandle handle () const;
```

hasBitmap (Boolean) — IClipboard

Description: If the clipboard has data with the format `IClipboard::bitmapFormat`, returns `true`.

Get:
```
virtual Boolean hasBitmap () const;
```

hasSelectedText (Boolean) — IEntryField

Description: Queries whether any of the entry field text is selected.

Get:
```
Boolean hasSelectedText () const;
```

hasSelectedText (Boolean) — IMultiLineEdit

Description: Queries whether any of the MLE's text is selected.

Get:
```
virtual Boolean hasSelectedText () const;
```

hasText (Boolean) — IClipboard

Description: If the clipboard has data with the format `IClipboard::textFormat`, returns `true`.

Get:
```
virtual Boolean hasText () const;
```

hasTransparentColor (Boolean) — IToolBarButton

Description: Returns `true` if a transparent color has been set.

Get:
```
virtual Boolean hasTransparentColor () const;
```

headingIcon (IPointerHandle) — IContainerColumn

Description: If the style is icon, the icon in the heading is returned.

Get:
```
virtual IPointerHandle headingIcon () const;
```

Set:
```
virtual IContainerColumn& setHeadingIcon
(unsigned long headingIcon);
```

headingIconHandle (Boolean) — IContainerColumn

Description: If the data in the column heading is an icon, `true` is returned.

Get:
```
Boolean isHeadingIconHandle () const;
```

headingString (Boolean) — IContainerColumn

Description: If the data in the column heading is a character string, `true` is returned.

Get:
```
Boolean isHeadingString () const;
```

headingText (IString) — IContainerColumn

Description: If the style is string, the text in the heading is returned.

Get:
```
virtual IString headingText () const;
```

Set:
```
virtual IContainerColumn& setHeadingText
(const char* headingText);
```

headingWriteable (Boolean) — IContainerColumn

Description: If the data in the column heading can be edited, `true` is returned.

Get:
```
Boolean isHeadingWriteable () const;
```

Set:
```
virtual IContainerColumn& enableHeadingUpdate
(Boolean headingWriteable=true);
```

height (IRectangle::Coord) — IRectangle

Description: Returns the height of the rectangle.

Get:
```
Coord height () const;
```

height (Coord) — ISize

Description: Returns the height represented by the `ISize` object.

Get:
```
Coord height () const;
```

Set:
```
ISize& setHeight (Coord height);
```

help (Boolean) — IPushButton

Description: If the style help is set, returns `true`.

Get:
```
Boolean isHelp () const;
```

Set:
```
virtual IPushButton& enableHelp (Boolean help=true);
```

helpId (unsigned long) — IContainerColumn

Description: Retrieves the help panel ID.

Get:
```
unsigned long helpId () const;
```

Set:
```
virtual IContainerColumn& setHelpId (unsigned long helpId);
```

helpId (unsigned long) — IContainerObject

Description: Called when an ICnrHelpEvent is dispatched to ICnrHandler.

Get:
```
virtual unsigned long helpId () const;
```

helpId (unsigned long) — IWindow

Description: Returns the help topic id for this window.

Get:
```
unsigned long helpId () const;
```

Set:
```
IWindow& setHelpId (unsigned long helpId);
```

hexDigits (Boolean) — IString

Description: If all the characters are in {'0'-'9','A'-'F','a'-'f'}, `true` is returned.

Get:
```
Boolean isHexDigits () const;
```

highlighted (Boolean) — IButton

Description: If the button's highlight state is set, `true` is returned.

Get:
```
Boolean isHighlighted () const;
```

highlighted (Boolean) — IMenuItem

Description: If the attribute highlighted is set, `true` is returned.

Get:
```
Boolean isHighlighted () const;
```

Set:
```
IMenuItem& setHighlighted (Boolean highlighted=true);
```

highLimit (double) — IVBDoublePart — vbsample

Description: Query the highLimit (`double`) attribute.

Get:
```
virtual double highLimit () const;
```

Set:
```
virtual IVBDoublePart& setHighLimit
(double highLimit=DBL_MAX);
```

ID:
```
highLimitId
```

highLimit (long) — IVBLongPart — vbsample

Description: Query the highLimit (`long`) attribute.

Get:
```
virtual long highLimit () const;
```

Set:
```
virtual IVBLongPart& setHighLimit (long highLimit=LONG_MAX);
```

ID:
```
highLimitId
```

highLimit (short) — IVBShortPart — vbsample

Description: Query the highLimit (short) attribute.

Get:
```
virtual short highLimit () const;
```

Set:
```
virtual IVBShortPart& setHighLimit (short highLimit=SHRT_MAX);
```

ID:
```
highLimitId
```

highLimit (unsigned long) — IVBUnsignedLongPart — vbsample

Description: Query the highLimit (unsigned long) attribute.

Get:
```
virtual unsigned long highLimit () const;
```

Set:
```
virtual IVBUnsignedLongPart& setHighLimit
(unsigned long highLimit=ULONG_MAX);
```

ID:
```
highLimitId
```

highLimit (unsigned short) — IVBUnsignedShortPart — vbsample

Description: Query the highLimit (unsigned short) attribute.

Get:
```
virtual unsigned short highLimit () const;
```

Set:
```
virtual IVBUnsignedShortPart& setHighLimit
(unsigned short highLimit=USHRT_MAX);
```

ID:
```
highLimitId
```

hiliteBackgroundColor (IColor) — IWindow

Description: Returns the highlight background color value of the window area, or the default if no color for the area has been set.

Get:
```
virtual IColor hiliteBackgroundColor () const;
```

Set:
```
virtual IWindow& setHiliteBackgroundColor
(const IColor& hiliteBackgroundColor);
```

ID:
```
hiliteBackgroundColorId
```

hiliteForegroundColor (IColor) — IWindow

Description: Returns the highlight foreground color value of the window area, or the default if no color for the area has been set.

Get:
```
virtual IColor hiliteForegroundColor () const;
```

Set:
```
virtual IWindow& setHiliteForegroundColor
(const IColor& hiliteForegroundColor);
```

ID:
```
hiliteForegroundColorId
```

homePhone (IString) — ICustomer — vbsample

Description: Query the homePhone (IString) attribute.

Get:
```
virtual IString homePhone () const;
```

Set:
```
virtual ICustomer& setHomePhone (const IString& homePhone);
```

ID:
```
homePhoneId
```

homePosition (IProgressIndicator::HomePosition) — IProgressIndicator

Description: Returns the current home position for this progress indicator object.

Get:
```
HomePosition homePosition () const;
```

Set:
```
virtual IProgressIndicator& setHomePosition
(HomePosition homePosition=IProgressIndicator::homeBottomLeft);
```

horizontal (Boolean) — IScrollBar

Description: If the scroll bar is horizontal, true is returned.

Get:
```
Boolean isHorizontal () const;
```

horizontalDataAlignment (IContainerColumn::HorizontalAlignment) — IContainerColumn

Description: Returns the current horizontal alignment of the column data in a container details view.

Get:
```
HorizontalAlignment horizontalDataAlignment () const;
```

horizontalHeadingAlignment (IContainerColumn::HorizontalAlignment) — IContainerColumn

Description: Returns the current horizontal alignment of the column heading data in a container details view.

Get:
```
HorizontalAlignment horizontalHeadingAlignment () const;
```

horizontalScroll (Boolean) — IBaseComboBox

Description: If the list box part of the combination box has a horizontal scrollbar, true is returned.

Get:
```
Boolean isHorizontalScroll () const;
```

horizontalScroll (Boolean) — IBaseListBox

Description: If the style horizontalScroll is set, `true` is returned.
Get:
```
Boolean isHorizontalScroll () const;
```

horizontalScrollBar (IScrollBar*) — IViewPort

Description: Returns the address of the horizontal scroll bar.
Get:
```
IScrollBar* horizontalScrollBar () const;
```

horizontalSeparator (Boolean) — IContainerColumn

Description: If the column heading has a separator drawn under it, `true` is returned.
Get:
```
Boolean hasHorizontalSeparator () const;
```

hours (unsigned long) — IMMTime — vbmm

Description: Returns the hours component of the time.
Get:
```
virtual unsigned long hours () const;
```

hours (unsigned) — ITime

Description: Returns the number of hours past midnight.
Get:
```
unsigned hours () const;
```

hundredths (unsigned long) — IMMTime — vbmm

Description: Returns the millisecond component of the time rounded to the nearest hundredth.
Get:
```
virtual unsigned long hundredths () const;
```

icon (IPointerHandle) — IContainerObject

Description: Returns the icon handle of an object.
Get:
```
virtual IPointerHandle icon () const;
```
Set:
```
virtual IContainerObject& setIcon (const IResourceId& icon);
```

icon (IPointerHandle) — IFrameWindow

Description: Returns the pointer handle of the window icon.

Get:
```
virtual IPointerHandle icon () const;
```

Set:
```
virtual IFrameWindow& setIcon (const IResourceId& icon);
```

icon (IPointerHandle) — IGraphicPushButton

Description: Returns the handle of the currently set icon.

Get:
```
IPointerHandle icon () const;
```

Set:
```
virtual IGraphicPushButton& setGraphic (unsigned long icon);
```

icon (IPointerHandle) — IIconControl

Description: Returns the handle of the currently set icon.

Get:
```
IPointerHandle icon () const;
```

Set:
```
virtual IIconControl& setIcon (unsigned long icon);
```

iconHandle (Boolean) — IContainerColumn

Description: If the data in the column is an icon, `true` is returned.

Get:
```
Boolean isIconHandle () const;
```

iconSize (ISize) — IContainerControl

Description: Returns the icon or bit-map size of all objects.

Get:
```
ISize iconSize () const;
```

Set:
```
virtual IContainerControl& setIconSize
(const ISize& iconSize);
```

iconText (IString) — IContainerObject

Description: Returns the text of an object.

Get:
```
virtual IString iconText () const;
```

Set:
```
virtual IContainerObject& setIconText
(const IString& iconText);
```

iconView (Boolean) — IContainerControl

Description: Queries whether the container is currently in an icon view.

Get:
```
Boolean isIconView (Boolean iconOnly=false) const;
```

id (unsigned long) — IMenuItem

Description: Returns the item identifier associated with the menu item.

Get:
```
unsigned long id () const;
```

id (unsigned long) — IResourceId

Description: Returns the identifier used for the resource.

Get:
```
unsigned long id () const;
```

id (unsigned long) — IWindow

Description: Returns the window ID.

Get:
```
unsigned long id () const;
```

Set:
```
virtual IWindow& setId (unsigned long id);
```

inactiveColor (IColor) — IWindow

Description: Returns the inactive color value of the window area, or the default if no color for the area has been set.

Get:
```
virtual IColor inactiveColor () const;
```

Set:
```
virtual IWindow& setInactiveColor
(const IColor& inactiveColor);
```

ID:
```
inactiveColorId
```

inactiveText (IString) — IInfoArea

Description: Returns the text displayed when the menu is inactive.

Get:
```
virtual IString inactiveText () const;
```

Set:
```
virtual IInfoArea& setInactiveText
(const IString& inactiveText);
```

ID:
```
inactiveTextId
```

inactiveTextBackgroundColor (IColor) — ITitle

Description: Returns the inactive text background color value of the title area.

Get:
```
virtual IColor inactiveTextBackgroundColor () const;
```

Set:
```
virtual ITitle& setInactiveTextBackgroundColor
(const IColor& inactiveTextBackgroundColor);
```

ID:
```
inactiveTextBackgroundColorId
```

inactiveTextForegroundColor (IColor) — ITitle

Description: Returns the inactive text foreground color value of the title area.

Get:
```
virtual IColor inactiveTextForegroundColor () const;
```

Set:
```
virtual ITitle& setInactiveTextForegroundColor
(const IColor& inactiveTextForegroundColor);
```

ID:
```
inactiveTextForegroundColorId
```

includesDBCS (Boolean) — IString

Description: If any characters are DBCS (double-byte character set), true is returned.

Get:
```
Boolean includesDBCS () const;
```

includesMBCS (Boolean) — IString

Description: If any characters are MBCS (multiple-byte character set), true is returned.

Get:
```
Boolean includesMBCS () const;
```

includesSBCS (Boolean) — IString

Description: If any characters are SBCS (single-byte character set), true is returned.

Get:
```
Boolean includesSBCS () const;
```

index (long) — IColor

Description: Returns the logical color table index closest to the red, green, and blue values.

Get:
```
virtual long index () const;
```

index (unsigned long) — IMenuItem

Description: Returns the ordinal position of the menu item in its submenu.

Get:
```
unsigned long index () const;
```

Set:
```
IMenuItem& setIndex (unsigned long index=IMenuItem::atEnd);
```

indexWindow (IWindowHandle) — IHelpWindow

Description: Returns the handle of the Index window.

Get:
```
IWindowHandle indexWindow () const;
```

initialize () — IVBContainerControl

Description: The initialize feature is not supported.

input1 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input1 (Boolean) attribute.
Get: `virtual Boolean input1 () const;`
Set: `virtual IVBLogicalAndPart& setInput1 (Boolean input1);`
ID: `input1Id`

input1 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input1 (Boolean) attribute.
Get: `virtual Boolean input1 () const;`
Set: `virtual IVBLogicalOrPart& setInput1 (Boolean input1);`
ID: `input1Id`

input10 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input10 (Boolean) attribute.
Get: `virtual Boolean input10 () const;`
Set: `virtual IVBLogicalAndPart& setInput10 (Boolean input10);`
ID: `input10Id`

input10 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input10 (Boolean) attribute.
Get: `virtual Boolean input10 () const;`
Set: `virtual IVBLogicalOrPart& setInput10 (Boolean input10);`
ID: `input10Id`

input2 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input2 (Boolean) attribute.
Get: `virtual Boolean input2 () const;`
Set: `virtual IVBLogicalAndPart& setInput2 (Boolean input2);`
ID: `input2Id`

input2 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input2 (Boolean) attribute.
Get: `virtual Boolean input2 () const;`
Set: `virtual IVBLogicalOrPart& setInput2 (Boolean input2);`
ID: `input2Id`

input3 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input3 (Boolean) attribute.
Get: `virtual Boolean input3 () const;`
Set: `virtual IVBLogicalAndPart& setInput3 (Boolean input3);`
ID: `input3Id`

input3 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input3 (Boolean) attribute.
Get: `virtual Boolean input3 () const;`
Set: `virtual IVBLogicalOrPart& setInput3 (Boolean input3);`
ID: `input3Id`

input4 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input4 (Boolean) attribute.
Get: `virtual Boolean input4 () const;`
Set: `virtual IVBLogicalAndPart& setInput4 (Boolean input4);`
ID: `input4Id`

input4 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input4 (Boolean) attribute.
Get: `virtual Boolean input4 () const;`
Set: `virtual IVBLogicalOrPart& setInput4 (Boolean input4);`
ID: `input4Id`

input5 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input5 (Boolean) attribute.
Get: `virtual Boolean input5 () const;`
Set: `virtual IVBLogicalAndPart& setInput5 (Boolean input5);`
ID: `input5Id`

input5 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input5 (Boolean) attribute.
Get: `virtual Boolean input5 () const;`
Set: `virtual IVBLogicalOrPart& setInput5 (Boolean input5);`
ID: `input5Id`

input6 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input6 (Boolean) attribute.
Get: `virtual Boolean input6 () const;`
Set: `virtual IVBLogicalAndPart& setInput6 (Boolean input6);`
ID: `input6Id`

input6 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input6 (Boolean) attribute.
Get: `virtual Boolean input6 () const;`
Set: `virtual IVBLogicalOrPart& setInput6 (Boolean input6);`
ID: `input6Id`

input7 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input7 (Boolean) attribute.
Get: `virtual Boolean input7 () const;`
Set: `virtual IVBLogicalAndPart& setInput7 (Boolean input7);`
ID: `input7Id`

input7 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input7 (Boolean) attribute.
Get: `virtual Boolean input7 () const;`
Set: `virtual IVBLogicalOrPart& setInput7 (Boolean input7);`
ID: `input7Id`

input8 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input8 (Boolean) attribute.
Get: `virtual Boolean input8 () const;`
Set: `virtual IVBLogicalAndPart& setInput8 (Boolean input8);`
ID: `input8Id`

input8 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input8 (Boolean) attribute.
Get: `virtual Boolean input8 () const;`
Set: `virtual IVBLogicalOrPart& setInput8 (Boolean input8);`
ID: `input8Id`

input9 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the input9 (Boolean) attribute.
Get: `virtual Boolean input9 () const;`
Set: `virtual IVBLogicalAndPart& setInput9 (Boolean input9);`
ID: `input9Id`

input9 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the input9 (Boolean) attribute.
Get: `virtual Boolean input9 () const;`
Set: `virtual IVBLogicalOrPart& setInput9 (Boolean input9);`
ID: `input9Id`

insertMode (Boolean) — IEntryField

Description: Queries whether the entry field is in insert mode.

Get:
```
Boolean isInsertMode () const;
```

Set:
```
virtual IEntryField& enableInsertMode
(Boolean insertMode=true);
```

ID:
```
insertModeId
```

internalLeading (unsigned long) — IFont

Description: Returns the approximate position of the top of a row of characters.

Get:
```
unsigned long internalLeading () const;
```

inUse (Boolean) — IContainerObject

Description: If this object has in-use emphasis, `true` is returned.

Get:
```
virtual Boolean isInUse (IContainerControl* container=0)
const;
```

Set:
```
virtual IContainerObject& setInUse (Boolean inUse=true,
IContainerControl* container=0);
```

isASCII (Boolean) — IString

Description: If all the characters are in {0x00-0x7F}, `true` is returned.

Get:
```
Boolean isASCII () const;
```

isBitmap (Boolean) — IMenuItem

Description: Returns `true` if this is a bitmap menu item.

Get:
```
Boolean isBitmap () const;
```

isDBCS (Boolean) — IString

Description: If all the characters are DBCS, `true` is returned.

Get:
```
Boolean isDBCS () const;
```

isDigits (Boolean) — ITextControl

Description: Return isDigits from the `IString` text attribute

Get:
```
VBISDIGITSFROMTEXT ();
```

ID:
```
textId
```

isDigits (Boolean) — ITextSpinButton

Description: Return isDigits from the `IString` text attribute
Get: `VBISDIGITSFROMTEXT ();`
ID: `textId`

isLowerCase (Boolean) — IString

Description: If all the characters are in {'a'-'z'}, `true` is returned.
Get: `Boolean isLowerCase () const;`

isMBCS (Boolean) — IString

Description: If all the characters are MBCS, `true` is returned.
Get: `Boolean isMBCS () const;`

isOpen (Boolean) — IClipboard

Description: Returns `true` if this `IClipboard` object has opened the clipboard.
Get: `virtual Boolean isOpen () const;`

isOpen (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device is opened.
Get: `Boolean isOpen () const;`

isSBCS (Boolean) — IString

Description: If all the characters are SBCS, `true` is returned.
Get: `Boolean isSBCS () const;`

isSet (Boolean) — IAccelerator

Description: If the accelerator is in effect, `true` is returned.
Get: `Boolean isSet () const;`

isText (Boolean) — IMenuItem

Description: Returns `true` if this is a menu item containing text.
Get: `Boolean isText () const;`

isUpperCase (Boolean) — IString

Description: If all the characters are in {'A'-'Z'}, `true` is returned.
Get: `Boolean isUpperCase () const;`

italic (Boolean) — IFont

Description: If the IFont object uses an italicized font, true is returned.

Get:
```
Boolean isItalic () const;
```

Set:
```
virtual IFont& setItalic (Boolean italic=true);
```

itemProvider (IDMItemProvider*) — IWindow

Description: Returns a pointer to the drag item provider (IDMItemProvider) for this window.

Get:
```
IDMItemProvider* itemProvider () const;
```

Set:
```
IWindow& setItemProvider (IDMItemProvider* itemProvider);
```

items (Collection*) — ICollectionViewComboBox

Description: Returns the collection currently being viewed.

Get:
```
virtual Collection* items () const;
```

Set:
```
virtual ICollectionViewComboBox<Element,Collection>& setItems
(Collection* items);
```

ID: itemsId

items (Collection*) — ICollectionViewListBox

Description: Returns the collection currently being viewed.

Get:
```
virtual Collection* items () const;
```

Set:
```
virtual ICollectionViewListBox<Element,Collection>& setItems
(Collection* items);
```

ID: itemsId

items (Collection*) — IVBContainerControl

Description: Returns the collection currently being viewed.

Get:
```
virtual Collection* items () const;
```

Set:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
setItems (Collection* items);
```

ID: itemsId

julianDate (unsigned long) — IDate

Description: Returns the Julian day number of the receiver IDate.

Get:
```
unsigned long julianDate () const;
```

keyCount (unsigned long) — IAcceleratorTable

Description: Returns the number of accelerator keys contained in the accelerator table.

Get:
```
unsigned long keyCount () const;
```

keyModifier (IKey::KeyModifier) — IAcceleratorKey

Description: Returns an `IAcceleratorKey::KeyModifier` value.

Get:
```
KeyModifier keyModifier () const;
```

lastElement (Element const) — IRDeque — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastElement (Element const) — IREqualitySequence — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastElement (Element const) — IRQueue — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastElement (Element const) — IRSequence

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastElement (Element const) — IRSortedBag — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastElement (Element const) — IRSortedSet — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastElement (Element const) — IRStack — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& lastElement () const;
```

lastPage (IPageHandle) — INotebook

Description: Returns an IPageHandle object that references the last page in the notebook.

Get:
```
virtual IPageHandle lastPage () const;
```

latched (Boolean) — ICustomButton

Description: Returns `true` if the button is in a latched state.

Get:
```
virtual Boolean isLatched () const;
```

Set:
```
virtual ICustomButton& latch (Boolean latched=true,
Boolean refresh=true);
```

ID: `latchId`

latchedBackgroundColor (IColor) — ICustomButton

Description: Returns the color used to paint the background of the custom button when it is in the latched state.

Get:
```
virtual IColor latchedBackgroundColor () const;
```

Set:
```
virtual ICustomButton& setLatchedBackgroundColor
(const IColor& latchedBackgroundColor, Boolean halftone=true);
```

latchedBackgroundColorHalftone (Boolean) — ICustomButton

Description: Returns `true` if the background is a halftone when the custom button is in the latched state.

Get:
```
virtual Boolean isLatchedBackgroundColorHalftone () const;
```

latchedBitmap (IBitmapHandle) — IToolBarButton

Description: Returns the handle of the bitmap that is displayed when the button is in the latched state.

Get:
```
IBitmapHandle latchedBitmap () const;
```

Set:
```
virtual IToolBarButton& setLatchedBitmap
(const IResourceId& latchedBitmap);
```

latchedForegroundColor (IColor) — ICustomButton

Description: Returns the color used to paint the foreground of the custom button when it is in the latched state.

Get:
```
virtual IColor latchedForegroundColor () const;
```

Set:
```
virtual ICustomButton& setLatchedForegroundColor
(const IColor& latchedForegroundColor);
```

latchingEnabled (Boolean) — ICustomButton

Description: Returns `true` if the latchable style is set for the button.

Get:
```
virtual Boolean isLatchingEnabled () const;
```

Set:
```
virtual ICustomButton& enableLatching
(Boolean latchingEnabled=true);
```

layoutAdjustment (IRectangle) — IWindow

Description: The layout adjustment is the dimensions that a window is moved and/or sized after a canvas runs its layout routines.

Get:
```
virtual IRectangle layoutAdjustment () const;
```

layoutType (IMenuItem::LayoutType) — IMenuItem

Description: Returns the menu item placement and layout.

Get:
```
LayoutType layoutType () const;
```

Set:
```
IMenuItem& setLayout (LayoutType layoutType);
```

left (IRectangle::Coord) — IRectangle

Description: Returns the X-coordinate of the vertical line that forms the left side of the rectangle.

Get:
```
Coord left () const;
```

leftCenter (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the left-center point of the rectangle.

Get:
```
IPoint leftCenter () const;
```

leftIndex (unsigned long) — IEntryField

Description: Returns the index of the first character displayed at the left edge of the entry field control.

Get:
```
unsigned long leftIndex () const;
```

Set:
```
virtual IEntryField& setLeftIndex (unsigned long leftIndex);
```

length (unsigned long) — IMMPlayableDevice — vbmm

Description: Returns the total length in the current time format as an `unsigned long`.

Get:
```
unsigned long length (CallType call=IMMDevice::wait) const;
```

length (unsigned) — IString

Description: Returns the length of the string, not counting the terminating NULL character.

Get:
```
unsigned length () const;
```

limit (unsigned long) — IBaseSpinButton

Description: Returns the number of characters permitted in the spin field.

Get:
```
unsigned long limit () const;
```

Set:
```
virtual IBaseSpinButton& setLimit (unsigned long limit=255);
```

limit (unsigned long) — IEntryField

Description: Returns the length, in bytes, of the longest text the entry field can hold.

Get:
```
unsigned long limit () const;
```

Set:
```
virtual IEntryField& setLimit (unsigned long limit);
```

ID:
```
limitId
```

limit (unsigned long) — IMultiLineEdit

Description: Returns the currently set number of bytes the MLE can hold.

Get:
```
unsigned long limit () const;
```

Set:
```
virtual IMultiLineEdit& setLimit (unsigned long limit);
```

ID:
```
limitId
```

limit (unsigned long) — IStaticText

Description: Returns the number of characters set by setLimit.

Get:
```
unsigned long limit () const;
```

Set:
```
virtual IStaticText& setLimit (unsigned long limit=0);
```

ID:
```
limitId
```

lineCount (unsigned long) — IInfoArea

Description: Returns the number of text lines that the information area uses.

Get:
```
virtual unsigned long lineCount () const;
```

Set:
```
virtual IInfoArea& setLineCount (unsigned long lineCount=1);
```

lineSpacing (long) — IContainerControl

Description: Returns the space between lines, in pixels.

Get:
```
long lineSpacing () const;
```

Set:
```
virtual IContainerControl& setLineSpacing (long lineSpacing);
```

listShowing (Boolean) — IBaseComboBox

Description: If the list box is visible, `true` is returned.

Get:
```
Boolean isListShowing () const;
```

location (IToolBar::Location) — IToolBar

Description: Returns the current location of the tool bar.

Get:
```
Location location () const;
```

Set:
```
virtual IToolBar& setLocation (Location location);
```

longStringTableOffset (long) — IFlyOverHelpHandler

Description: Returns the offset currently in use for the `long` text control.

Get:
```
virtual long longStringTableOffset () const;
```

Set:
```
virtual IFlyOverHelpHandler& setLongStringTableOffset
(long longStringTableOffset=0);
```

longTextControl (ITextControl*) — IFlyOverHelpHandler

Description: Returns a pointer to the text control that is used to display `long` help text.

Get:
```
virtual ITextControl* longTextControl () const;
```

Set:
```
virtual IFlyOverHelpHandler& setLongTextControl
(ITextControl* longTextControl);
```

longTextControl (ITextControl*) — IVBFlyText

Description: Text control used to display the `long` version of the flyover text

Get:
```
ITextControl* longTextControl () const;
```

Set:
```
IVBFlyText& setLongTextControl
(ITextControl* longTextControl);
```

lowerBound (Coord) — IRange

Description: Returns the lower bound of the range.
Get:
```
Coord lowerBound () const;
```
Set:
```
IRange& setLowerBound (Coord lowerBound);
```

lowerCase (Boolean) — IVBStringPart — vbsample

Description: Query the lowerCase (Boolean) attribute.
Get:
```
virtual Boolean isLowerCase () const;
```
ID:
```
textId
```

lowLimit (double) — IVBDoublePart — vbsample

Description: Query the lowLimit (double) attribute.
Get:
```
virtual double lowLimit () const;
```
Set:
```
virtual IVBDoublePart& setLowLimit (double lowLimit=DBL_MIN);
```
ID:
```
lowLimitId
```

lowLimit (long) — IVBLongPart — vbsample

Description: Query the lowLimit (long) attribute.
Get:
```
virtual long lowLimit () const;
```
Set:
```
virtual IVBLongPart& setLowLimit (long lowLimit=LONG_MIN);
```
ID:
```
lowLimitId
```

lowLimit (short) — IVBShortPart — vbsample

Description: Query the lowLimit (short) attribute.
Get:
```
virtual short lowLimit () const;
```
Set:
```
virtual IVBShortPart& setLowLimit (short lowLimit=SHRT_MIN);
```
ID:
```
lowLimitId
```

lowLimit (unsigned long) — IVBUnsignedLongPart — vbsample

Description: Query the lowLimit (unsigned long) attribute.

Get:
```
virtual unsigned long lowLimit () const;
```

Set:
```
virtual IVBUnsignedLongPart& setLowLimit
(unsigned long lowLimit=0);
```

ID:
```
lowLimitId
```

lowLimit (unsigned short) — IVBUnsignedShortPart — vbsample

Description: Query the lowLimit (unsigned short) attribute.

Get:
```
virtual unsigned short lowLimit () const;
```

Set:
```
virtual IVBUnsignedShortPart& setLowLimit
(unsigned short lowLimit=0);
```

ID:
```
lowLimitId
```

majorTabBackgroundColor (IColor) — INotebook

Description: Returns the major tab background color of the notebook or the default if you have not set it.

Get:
```
virtual IColor majorTabBackgroundColor () const;
```

Set:
```
virtual INotebook& setMajorTabBackgroundColor
(const IColor& majorTabBackgroundColor);
```

ID:
```
majorTabBackgroundColorId
```

majorTabForegroundColor (IColor) — INotebook

Description: Returns the major tab foreground color of the notebook or the default if you have not set it.

Get:
```
virtual IColor majorTabForegroundColor () const;
```

Set:
```
virtual INotebook& setMajorTabForegroundColor
(const IColor& majorTabForegroundColor);
```

ID:
```
majorTabForegroundColorId
```

margin (Boolean) — IEntryField

Description: Returns a Boolean depending on whether the control margin has a surrounding border.

Get:
```
Boolean isMargin () const;
```

Set:
```
virtual IEntryField& enableMargin (Boolean margin=true);
```

margin (ISize) — ISetCanvas

Description: Returns the margin width and height, which is the space between the edge of the canvas and the outermost child windows.

Get:
```
ISize margin () const;
```

Set:
```
virtual ISetCanvas& setMargin (const ISize& margin);
```

marginSize (ISize) — IGraphicPushButton

Description: Returns the currently set margin size.

Get:
```
ISize marginSize () const;
```

Set:
```
IGraphicPushButton& setMarginSize (const ISize& marginSize);
```

master (Boolean) — IBaseSpinButton

Description: If the spin button is a master, `true` is returned.

Get:
```
Boolean isMaster () const;
```

maxAscender (unsigned long) — IFont

Description: Returns the height of the largest ascender above the baseline.

Get:
```
unsigned long maxAscender () const;
```

maxCharHeight (unsigned long) — IFont

Description: Returns the height portion of `IFont::maxSize`.

Get:
```
unsigned long maxCharHeight () const;
```

maxDescender (unsigned long) — IFont

Description: Returns the height of the largest descender below the baseline.

Get:
```
unsigned long maxDescender () const;
```

maximized (Boolean) — IFrameWindow

Description: If the frame window is maximized, `true` is returned.

Get:
```
Boolean isMaximized () const;
```

maximizeRect (IRectangle) — IFrameWindow

Description: Returns a rectangle indicating the size and position the frame window has when it is maximized.

Get:
```
virtual IRectangle maximizeRect () const;
```

maximumSpeed (IMMSpeed) — IMMDigitalVideo — vbmm

Description: Returns the maximum play rate in the current speed format, either as percentage or in frames-per-second.

Get:
```
IMMSpeed maximumSpeed (CallType call=IMMDevice::wait) const;
```

maximumWindows (unsigned long) — IMMDigitalVideo — vbmm

Description: Returns the maximum number of video windows allowed on this machine.

Get:
```
unsigned long maximumWindows (CallType call=IMMDevice::wait)
const;
```

maxNumberOfElements (INumber) — IRBag — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRDeque — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IREqualitySequence — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRHeap — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRQueue — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRSequence

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRSet — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRSortedBag — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRSortedSet — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxNumberOfElements (INumber) — IRStack — vbcc

Description: Returns the maximum number of elements the collection can contain.

Get:
```
INumber maxNumberOfElements () const;
```

maxSize (ISize) — IFont

Description: Returns the maximum width and height.

Get:
```
ISize maxSize () const;
```

maxUppercaseSize (ISize) — IFont

Description: Returns the maximum size for an uppercase character.

Get:
```
ISize maxUppercaseSize () const;
```

maxX (IRectangle::Coord) — IRectangle

Description: Returns the X-coordinate of the vertical line that is opposite the origin of the rectangle.

Get:
```
Coord maxX () const;
```

maxXCenterY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the center point of the vertical line opposite the origin of the rectangle.

Get:
```
IPoint maxXCenterY () const;
```

maxXMaxY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the corner of the rectangle opposite the origin.

Get:
```
IPoint maxXMaxY () const;
```

maxXMinY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the corner of the rectangle at the other end of the horizontal line passing though the origin.

Get:
```
IPoint maxXMinY () const;
```

maxY (IRectangle::Coord) — IRectangle

Description: Returns the Y-coordinate of the horizontal line opposite the origin.

Get:
```
Coord maxY () const;
```

mediaPresent (Boolean) — IMMRemovableMedia — vbmm

Description: Returns `true` if media is inserted in the device; otherwise, it returns `false`.

Get:
```
Boolean isMediaPresent (CallType call=IMMDevice::wait) const;
```

menu (IMenuBar) — IFrameWindow

Description: Menu bar link.

menu (IMenu) — IMenuCascade

Description: Pulldown menu.

menu (IPopupMenu) — IWindow

Description: Popup menu link.

minimized (Boolean) — IFrameWindow

Description: If the frame window is minimized, `true` is returned.

Get:
```
Boolean isMinimized () const;
```

minimizeRect (IRectangle) — IFrameWindow

Description: Returns a rectangle indicating the size and position the frame window has when it is minimized.

Get:
```
virtual IRectangle minimizeRect () const;
```

minimumCharacters (unsigned long) — IBaseListBox

Description: Returns the maximum number of characters in an item of a minimum size list box.

Get:
```
unsigned long minimumCharacters () const;
```

Set:
```
virtual IBaseListBox& setMinimumCharacters
(unsigned long minimumCharacters);
```

minimumRows (unsigned long) — IBaseComboBox

Description: Returns the number of visible rows in the list box portion of a minimum size combination-box window.

Get:
```
unsigned long minimumRows () const;
```

Set:
```
virtual IBaseComboBox& setMinimumRows
(unsigned long minimumRows);
```

minimumRows (unsigned long) — IBaseListBox

Description: Returns the number of visible rows of a minimum size list box.

Get:
```
unsigned long minimumRows () const;
```

Set:
```
virtual IBaseListBox& setMinimumRows
(unsigned long minimumRows);
```

minimumSize (ISize) — IWindow

Description: Returns the minimum allowable size set by a derived class.

Get:
```
ISize minimumSize (Boolean windowCalculatedSize=false) const;
```

Set:
```
IWindow& setMinimumSize (const ISize& minimumSize);
```

minimumSpeed (IMMSpeed) — IMMDigitalVideo — vbmm

Description: Returns the minimum play rate in the current speed format, either as percentage or in frames-per-second.

Get:
```
IMMSpeed minimumSpeed (CallType call=IMMDevice::wait) const;
```

minorTabBackgroundColor (IColor) — INotebook

Description: Returns the minor tab background color of the notebook or the default if you have not set it.

Get:
```
virtual IColor minorTabBackgroundColor () const;
```

Set:
```
virtual INotebook& setMinorTabBackgroundColor
(const IColor& minorTabBackgroundColor);
```

ID:
```
minorTabBackgroundColorId
```

minorTabForegroundColor (IColor) — INotebook

Description: Returns the minor tab foreground color of the notebook or the default if you have not set it.

Get:
```
virtual IColor minorTabForegroundColor () const;
```

Set:
```
virtual INotebook& setMinorTabForegroundColor
(const IColor& minorTabForegroundColor);
```

ID:
```
minorTabForegroundColorId
```

minScrollIncrement (unsigned long) — IScrollBar

Description: Returns the amount of the scrollable range that is scrolled by selecting the scroll buttons.

Get:
```
virtual unsigned long minScrollIncrement () const;
```

Set:
```
virtual IScrollBar& setMinScrollIncrement
(unsigned long minScrollIncrement=1);
```

minutes (unsigned long) — IMMTime — vbmm

Description: Returns the minutes component of the time.

Get:
```
virtual unsigned long minutes () const;
```

minutes (unsigned) — ITime

Description: Returns the number of minutes past the hour.

Get:
```
unsigned minutes () const;
```

minX (IRectangle::Coord) — IRectangle

Description: Returns the X-coordinate of the vertical line that passes though the origin.

Get:
```
Coord minX () const;
```

minXCenterY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the center of the vertical line that passes through the origin.

Get:
```
IPoint minXCenterY () const;
```

minXMaxY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the corner of the rectangle at the other end of the vertical line passing though the origin.

Get:
```
IPoint minXMaxY () const;
```

minXMinY (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the origin corner of the rectangle.

Get:
```
IPoint minXMinY () const;
```

minY (IRectangle::Coord) — IRectangle

Description: Returns the Y-coordinate of the horizontal line that passes though the origin of the rectangle.

Get:
```
Coord minY () const;
```

misfitFilteringEnabled (Boolean) — IToolBar

Description: Returns `true` if the filterMisfits style is set for the tool bar.

Get:
```
Boolean isMisfitFilteringEnabled () const;
```

Set:
```
IToolBar& enableMisfitFiltering
(Boolean misfitFilteringEnabled=true);
```

missingText (IString) — IInfoArea

Description: Returns the text that displays when the ICLUI cannot obtain the required information string from the resource library.

Get:
```
virtual IString missingText () const;
```

Set:
```
virtual IInfoArea& setMissingText (unsigned long missingText);
```

ID:
```
missingTextId
```

mixedTargetEmphasis (Boolean) — IContainerControl

Description: Queries if the mixed target emphasis mode is set for drag operations over this container.

Get:
```
Boolean hasMixedTargetEmphasis () const;
```

mleHandler (IHandler) — ICnrEditHandler

Description: Called to return the handler for a multi-line entry field.

Get:
```
IHandler* mleHandler () const;
```

modal (Boolean) — IFrameWindow

Description: If the window is displayed in application-modal mode, `true` is returned.

Get:
```
Boolean isModal () const;
```

mode (IMMDevice::Mode) — IMMDevice — vbmm

Description: Returns the current mode of the device.

Get:
```
virtual Mode mode (CallType call=IMMDevice::wait) const;
```

modeless (Boolean) — IVBFileDialog

Description: Returns `true` if this is a modeless dialog.

Get:
```
Boolean isModeless () const;
```

modeless (Boolean) — IVBFontDialog

Description: Returns `true` if the dismissed dialog was modeless.

Get:
```
Boolean isModeless () const;
```

monitoringEnabled (Boolean) — IMMAmpMixer — vbmm

Description: Returns `true` if the monitor is turned on.

Get:
```
Boolean isMonitoringEnabled (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMAmpMixer& enableMonitoring
(Boolean monitoringEnabled=true,
CallType call=IMMDevice::wait);
```

monthOfYear (IDate::Month) — IDate

Description: Returns the index of the receiver's month of the year.

Get:
```
Month monthOfYear () const;
```

mousePointer (IPointerHandle) — IFrameWindow

Description: Returns the handle of the mouse pointer used when the mouse is over the frame window.

Get:
```
virtual IPointerHandle mousePointer () const;
```

Set:
```
virtual IFrameWindow& setMousePointer
(const IPointerHandle& mousePointer);
```

multipleSelect (Boolean) — IBaseListBox

Description: If the style multipleSelect is set, `true` is returned.

Get:
```
Boolean isMultipleSelect () const;
```

Set:
```
virtual IBaseListBox& enableMultipleSelect
(Boolean multipleSelect=true);
```

multipleSelection (Boolean) — IContainerControl

Description: Queries whether the container is in multiple selection mode.

Get:
```
Boolean isMultipleSelection () const;
```

name (IString) — ICompany — vbsample

Description: Query the name (`IString`) attribute.

Get:
```
virtual IString name () const;
```

Set:
```
virtual ICompany& setName (const IString& name);
```

ID:
```
nameId
```

name (IString) — ICustomer — vbsample

Description: Query the name (`IString`) attribute.

Get:
```
virtual IString name () const;
```

Set:
```
virtual ICustomer& setName (const IString& name);
```

ID:
```
nameId
```

name (IString) — IFont

Description: Returns the face name of the font.

Get:
```
IString name () const;
```

Set:
```
virtual IFont& setName (const char* name,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

name (IString) — IProfile

Description: Returns the profile's file name.

Get:
```
virtual IString name () const;
```

nameView (Boolean) — IContainerControl

Description: Queries whether the container is currently in a name view.

Get:
```
Boolean isNameView (Boolean nameOnly=false) const;
```

nativeRect (IRectangle) — IWindow

Description: Returns a rectangle representing the position and size of the window.

Get:
```
virtual IRectangle nativeRect () const;
```

nextShellRect (IRectangle) — IFrameWindow

Description: Returns the system-recommended coordinates of the next main window.

Get:
```
static IRectangle nextShellRect ();
```

noAdjustPosition (Boolean) — IBaseListBox

Description: If the style noAdjustPosition is set, `true` is returned.

Get:
```
Boolean isNoAdjustPosition () const;
```

Set:
```
virtual IBaseListBox& enableNoAdjustPosition
(Boolean noAdjustPosition=true);
```

noDismiss (Boolean) — IMenuItem

Description: If the attribute noDismiss is set, `true` is returned.

Get:
```
Boolean isNoDismiss () const;
```

Set:
```
IMenuItem& setNoDismiss (Boolean noDismiss=true);
```

nonPropOnly (Boolean) — IFont

Description: If the `IFont` object uses only non-proportional fonts, `true` is returned.

Get:
```
Boolean isNonPropOnly () const;
```

normalSpeed (IMMSpeed) — IMMDigitalVideo — vbmm

Description: Returns the device's normal play rate in the current speed format, either as a percentage or in frames-per-second.

Get:
```
IMMSpeed normalSpeed (CallType call=IMMDevice::wait) const;
```

normalTargetEmphasis (Boolean) — IContainerControl

Description: Queries if the regular target emphasis mode is set for drag operations over this container.

Get:
```
Boolean hasNormalTargetEmphasis () const;
```

notInput1 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput1 (Boolean) attribute.
Get: `virtual Boolean notInput1 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput1 (Boolean notInput1);`

notInput1 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput1 (Boolean) attribute.
Get: `virtual Boolean notInput1 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput1 (Boolean notInput1);`

notInput10 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput10 (Boolean) attribute.
Get: `virtual Boolean notInput10 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput10 (Boolean notInput10);`

notInput10 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput10 (Boolean) attribute.
Get: `virtual Boolean notInput10 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput10 (Boolean notInput10);`

notInput2 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput2 (Boolean) attribute.
Get: `virtual Boolean notInput2 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput2 (Boolean notInput2);`

notInput2 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput2 (Boolean) attribute.
Get: `virtual Boolean notInput2 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput2 (Boolean notInput2);`

notInput3 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput3 (Boolean) attribute.
Get: `virtual Boolean notInput3 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput3 (Boolean notInput3);`

notInput3 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput3 (Boolean) attribute.
Get: `virtual Boolean notInput3 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput3 (Boolean notInput3);`

notInput4 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput4 (Boolean) attribute.
Get: `virtual Boolean notInput4 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput4 (Boolean notInput4);`

notInput4 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput4 (Boolean) attribute.
Get: `virtual Boolean notInput4 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput4 (Boolean notInput4);`

notInput5 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput5 (Boolean) attribute.
Get: `virtual Boolean notInput5 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput5 (Boolean notInput5);`

notInput5 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput5 (Boolean) attribute.
Get: `virtual Boolean notInput5 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput5 (Boolean notInput5);`

notInput6 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput6 (Boolean) attribute.
Get: `virtual Boolean notInput6 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput6 (Boolean notInput6);`

notInput6 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput6 (Boolean) attribute.
Get: `virtual Boolean notInput6 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput6 (Boolean notInput6);`

notInput7 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput7 (Boolean) attribute.
Get: `virtual Boolean notInput7 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput7 (Boolean notInput7);`

notInput7 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput7 (Boolean) attribute.
Get: `virtual Boolean notInput7 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput7 (Boolean notInput7);`

notInput8 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput8 (Boolean) attribute.
Get: `virtual Boolean notInput8 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput8 (Boolean notInput8);`

notInput8 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput8 (Boolean) attribute.
Get: `virtual Boolean notInput8 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput8 (Boolean notInput8);`

notInput9 (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notInput9 (Boolean) attribute.
Get: `virtual Boolean notInput9 () const;`
Set: `virtual IVBLogicalAndPart& setNotInput9 (Boolean notInput9);`

notInput9 (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notInput9 (Boolean) attribute.
Get: `virtual Boolean notInput9 () const;`
Set: `virtual IVBLogicalOrPart& setNotInput9 (Boolean notInput9);`

notValue (Boolean) — IVBBooleanPart — vbsample

Description: Query the notValue (Boolean) attribute.
Get:
```
virtual Boolean notValue () const;
```
Set:
```
virtual IVBBooleanPart& setNotValue (Boolean notValue);
```
ID:
```
valueId
```

notValue (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the notValue (Boolean) attribute.
Get:
```
virtual Boolean notValue () const;
```
ID:
```
valueId
```

notValue (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the notValue (Boolean) attribute.
Get:
```
virtual Boolean notValue () const;
```
ID:
```
valueId
```

notValueAsText (IString) — IVBBooleanPart — vbsample

Description: Query the notValueAsText (IString) attribute.
Get:
```
virtual IString notValueAsText () const;
```
Set:
```
virtual IVBBooleanPart& setNotValueAsText
(const IString& notValueAsText);
```
ID:
```
valueId
```

now (ITime) — ITime

Description: Returns the current time.
Get:
```
static ITime now ();
```

number (Boolean) — IContainerColumn

Description: If the data in the column is a number, true is returned.
Get:
```
Boolean isNumber () const;
```

numberOfApplications (unsigned long) — IProfile

Description: Returns the number of application names in the profile.
Get:
```
virtual unsigned long numberOfApplications () const;
```

numberOfButton2Items (unsigned long) — IVBContainerControl

Description: Returns the number of button2ed items in the container.
Get: `unsigned long numberOfButton2Objects ();`
ID: `button2PointId`

numberOfColumnChanges (unsigned long) — IContainerControl

Description: Retrieves the container's count of column additions and removals.
Get: `unsigned long numberOfColumnChanges () const;`

numberOfDifferentElements (INumber) — IRBag — vbcc

Description: Returns the number of different elements in the collection.
Get: `INumber numberOfDifferentElements () const;`

numberOfDifferentElements (INumber) — IRSortedBag — vbcc

Description: Returns the number of different elements in the collection.
Get: `INumber numberOfDifferentElements () const;`

numberOfElements (INumber) — IRBag — vbcc

Description: Returns the number of elements the collection contains.
Get: `INumber numberOfElements () const;`

numberOfElements (INumber) — IRDeque — vbcc

Description: Returns the number of elements the collection contains.
Get: `INumber numberOfElements () const;`

numberOfElements (INumber) — IREqualitySequence — vbcc

Description: Returns the number of elements the collection contains.
Get: `INumber numberOfElements () const;`

numberOfElements (INumber) — IRHeap — vbcc

Description: Returns the number of elements the collection contains.
Get: `INumber numberOfElements () const;`

numberOfElements (INumber) — IRQueue — vbcc

Description: Returns the number of elements the collection contains.
Get: `INumber numberOfElements () const;`

numberOfElements (INumber) — IRSequence

Description: Returns the number of elements the collection contains.
Get:
```
INumber numberOfElements () const;
```

numberOfElements (INumber) — IRSet — vbcc

Description: Returns the number of elements the collection contains.
Get:
```
INumber numberOfElements () const;
```

numberOfElements (INumber) — IRSortedBag — vbcc

Description: Returns the number of elements the collection contains.
Get:
```
INumber numberOfElements () const;
```

numberOfElements (INumber) — IRSortedSet — vbcc

Description: Returns the number of elements the collection contains.
Get:
```
INumber numberOfElements () const;
```

numberOfElements (INumber) — IRStack — vbcc

Description: Returns the number of elements the collection contains.
Get:
```
INumber numberOfElements () const;
```

numberOfEntries (unsigned long) — IMMAudioCDContents — vbmm

Description: Returns the number of tracks in the playback list.
Get:
```
unsigned long numberOfEntries () const;
```

numberOfKeys (unsigned long) — IProfile

Description: Returns the number of keys for the application name.
Get:
```
virtual unsigned long numberOfKeys (const char* applName=0)
const;
```

numberOfLines (unsigned long) — IMultiLineEdit

Description: Returns the number of displayed lines, including word breaks.
Get:
```
unsigned long numberOfLines () const;
```

numberOfObjectChanges (unsigned long) — IContainerControl

Description: Retrieves the container's count of object additions and removals.
Get:
```
unsigned long numberOfObjectChanges () const;
```

numberOfSelections (unsigned long) — IBaseComboBox

Description: If no item is selected, 0 is returned.
Get: `virtual unsigned long numberOfSelections () const;`
ID: `selectId`

numberOfSelections (unsigned long) — IBaseListBox

Description: For single-selection list boxes, if no item is selected, 0 is returned.
Get: `virtual unsigned long numberOfSelections () const;`
ID: `selectId`

numberOfSelections (unsigned long) — IVBContainerControl

Description: Returns the number of selected items in the container.
Get: `unsigned long numberOfSelections ();`
ID: `selectId`

numberOfTracks (unsigned long) — IMMAudioCD — vbmm

Description: Returns the number of tracks.
Get: `unsigned long numberOfTracks (CallType call=IMMDevice::wait) const;`

numberOfTracks (unsigned long) — IMMAudioCDContents — vbmm

Description: Returns the number of tracks.
Get: `unsigned long numberOfTracks () const;`

numWords (unsigned) — IString

Description: Returns the number of words in the receiver.
Get: `unsigned numWords () const;`

objectCopy (IContainerObject*) — IContainerObject

Description: Called when it is necessary to make a copy of an object.
Get: `virtual IContainerObject* objectCopy ();`

objectCount (unsigned long) — IContainerControl

Description: Returns the number of objects in the container.
Get: `unsigned long objectCount () const;`

objectList •ordinal

objectList (ICnrObjectSet) — IContainerControl

Description: Returns the set of all objects in the container.
Get:
```
ICnrObjectSet objectList () const;
```

objectText (IString) — ITitle

Description: Returns the object text.
Get:
```
virtual IString objectText () const;
```
Set:
```
virtual ITitle& setObjectText (const char* objectText);
```
ID:
```
objectTextId
```

open (Boolean) — IContainerObject

Description: If this object is open, `true` is returned.
Get:
```
virtual Boolean isOpen () const;
```
Set:
```
virtual IContainerObject& setOpen (Boolean open=true);
```

open (Boolean) — IResourceLibrary

Description: Returns the open state of the resource file.
Get:
```
virtual Boolean isOpen () const;
```

ordered (IBoolean) — IRecord — vbsample

Description: Returns `false`.
Get:
```
virtual IBoolean isOrdered () const;
```

orderedTargetEmphasis (Boolean) — IContainerControl

Description: Queries if the ordered target emphasis mode is set for drag operations over this container.
Get:
```
Boolean hasOrderedTargetEmphasis () const;
```

ordinal (unsigned long) — IMMTime — vbmm

Description: Returns an ordinal number in milliseconds.
Get:
```
virtual unsigned long ordinal () const;
```
Set:
```
virtual IMMTime& setTimeToOrdinal (unsigned long ordinal);
```

orientation (INotebook::Orientation) — INotebook

Description: Returns the current orientation of the notebook, which is the location of the major tabs in relation to the back pages' corner.

Get:
```
virtual Orientation orientation () const;
```

Set:
```
virtual INotebook& setOrientation (Orientation orientation);
```

ID:
```
orientationId
```

orientation (ISplitCanvas::Orientation) — ISplitCanvas

Description: Returns an enumerator Orientation for the orientation of the canvas's split bars.

Get:
```
Orientation orientation () const;
```

Set:
```
virtual ISplitCanvas& setOrientation
(Orientation orientation);
```

ID:
```
orientationId
```

origDefaultButtonHandle (IWindowHandle) — ICanvas

Description: Returns the push button originally identified as the default.

Get:
```
IWindowHandle origDefaultButtonHandle () const;
```

outline (Boolean) — IFont

Description: If the `IFont` object uses a font that appears hollow, `true` is returned.

Get:
```
Boolean isOutline () const;
```

Set:
```
virtual IFont& setOutline (Boolean outline=true);
```

outlineType (IOutlineBox::OutlineType) — IOutlineBox

Description: Returns the current type of outline for this outline box object.

Get:
```
OutlineType outlineType () const;
```

Set:
```
IOutlineBox& setOutlineType (const OutlineType& outlineType);
```

owner (IWindow*) — IAccelerator

Description: Returns the `IWindow` to which the accelerator applies.

Get:
```
IWindow* owner () const;
```

owner (IWindowHandle) — IClipboard

Description: Returns the current clipboard owner window.

Get:
```
IWindowHandle owner () const;
```

owner (IWindow*) — IWindow

Description: Returns the window's owner.

Get:
```
IWindow* owner () const;
```

Set:
```
virtual IWindow& setOwner (const IWindow* owner);
```

packType (ISetCanvas::PackType) — ISetCanvas

Description: Returns an enumerator for child window spacing in decks.

Get:
```
PackType packType () const;
```

Set:
```
virtual ISetCanvas& setPackType (PackType packType);
```

pad (ISize) — ISetCanvas

Description: Returns the pad width and height, which is the space between child windows in a deck and between multiple decks.

Get:
```
ISize pad () const;
```

Set:
```
virtual ISetCanvas& setPad (const ISize& pad);
```

pageBackgroundColor (IColor) — INotebook

Description: Returns the page background color of the notebook or the default if you have not set it.

Get:
```
virtual IColor pageBackgroundColor () const;
```

Set:
```
virtual INotebook& setPageBackgroundColor
(const IColor& pageBackgroundColor);
```

ID:
```
pageBackgroundColorId
```

pageScrollIncrement (unsigned long) — IScrollBar

Description: Returns the amount of the scrollable range that is scrolled by selecting the scroll shaft.

Get:
```
virtual unsigned long pageScrollIncrement () const;
```

Set:
```
virtual IScrollBar& setPageScrollIncrement
(unsigned long pageScrollIncrement=0);
```

pageSize (ISize) — INotebook

Description: Returns the size of the notebook's pages.

Get:
```
virtual ISize pageSize () const;
```

parent (IWindow*) — IWindow

Description: Returns the window's parent.
Get:
```
IWindow* parent () const;
```
Set:
```
virtual IWindow& setParent (const IWindow* parent);
```

parentSize (ISize) — IWindow

Description: Returns an ISize object representing the size of the client rectangle in the parent window.
Get:
```
virtual ISize parentSize () const;
```

pauseButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the pause animated button.
Get:
```
IAnimatedButton* pauseButton () const;
```

phone (IString) — ICompany — vbsample

Description: Query the phone (IString) attribute.
Get:
```
virtual IString phone () const;
```
Set:
```
virtual ICompany& setPhone (const IString& phone);
```
ID:
```
phoneId
```

pitch (unsigned long) — IMMAmpMixer — vbmm

Description: Returns the pitch, where 0 is the lowest pitch and 100 the highest pitch.
Get:
```
unsigned long pitch (CallType call=IMMDevice::wait) const;
```
Set:
```
virtual IMMAmpMixer& setPitch (unsigned long pitch,
CallType call=IMMDevice::wait);
```

playableDevice (IMMPlayableDevice*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the IMMPlayableDevice the player is controlling.
Get:
```
IMMPlayableDevice* playableDevice () const;
```
Set:
```
IMMPlayerPanel& setPlayableDevice
(IMMPlayableDevice* playableDevice);
```

playButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the play animated button.
Get:
```
IAnimatedButton* playButton () const;
```

playingForward (Boolean) — IMMDigitalVideo — vbmm

Description: Returns `true` if the play direction is forward or if the device is not playing.

Get:
```
Boolean isPlayingForward (CallType call=IMMDevice::wait)
const;
```

pointerCaptured (Boolean) — IWindow

Description: This function returns `true` if this window is currently capturing pointer events.

Get:
```
virtual Boolean hasPointerCaptured () const;
```

pointSize (unsigned long) — IFont

Description: Returns the point size of the font.

Get:
```
unsigned long pointSize () const;
```

Set:
```
virtual IFont& setPointSize (unsigned long pointSize,
const IPresSpaceHandle& presSpaceHandle=IPresSpaceHandle ());
```

pointSize (unsigned long) — IVBFontDialog

Description: Returns the font's point size.

Get:
```
unsigned long pointSize () const;
```

position (unsigned long) — IMMPlayableDevice — vbmm

Description: Returns the current position in the current time format as an `unsigned long`.

Get:
```
unsigned long position (CallType call=IMMDevice::wait) const;
```

position (IPoint) — IWindow

Description: Returns the position of the window.

Get:
```
virtual IPoint position () const;
```

Set:
```
virtual IWindow& moveTo (const IPoint& position);
```

ID:
```
positionId
```

pressedOK (Boolean) — IVBFileDialog

Description: Returns `true` if the user ended the dialog by pressing the OK push button.

Get:
```
Boolean pressedOK () const;
```

pressedOK (Boolean) — IVBFontDialog

Description: Returns `true` if the user ended the dialog by pressing the OK push button.

Get:
```
Boolean pressedOK () const;
```

presSpace (IPresSpaceHandle) — IWindow

Description: Acquires and returns the presentation space handle (also known as graphics context, or GC) for the window.

Get:
```
virtual IPresSpaceHandle presSpace () const;
```

primaryFormat (IString) — IClipboard

Description: Returns the primary format of the data on the clipboard.

Get:
```
static IString primaryFormat ();
```

primaryScale (IProgressIndicator::Scale) — IProgressIndicator

Description: Returns the current primary scale of this progress indicator object.

Get:
```
Scale primaryScale () const;
```

Set:
```
virtual IProgressIndicator& setPrimaryScale
(Scale primaryScale=IProgressIndicator::scale1);
```

ID:
```
scaleId
```

printable (Boolean) — IString

Description: If all the characters are in {0x20-0x7E}, `true` is returned.

Get:
```
Boolean isPrintable () const;
```

profile (IProfile) — IMMAudioCD — vbmm

Description: Returns the current profile.

Get:
```
IProfile profile () const;
```

Set:
```
IMMAudioCD& setProfile (IProfile& profile);
```

punctuation (Boolean) — IString

Description: If none of the characters is white space, a control character, or an alphanumeric character, `true` is returned.

Get:
```
Boolean isPunctuation () const;
```

radius (unsigned long) — ICircularSlider

Description: Returns the radius of the dial in pixels.

Get:
```
unsigned long radius () const;
```

range (IRange) — INumericSpinButton

Description: Returns the number range of the spin button.

Get:
```
virtual IRange range () const;
```

Set:
```
virtual INumericSpinButton& setRange (const IRange& range,
Boolean override=false);
```

readOnly (Boolean) — IMMFileMedia — vbmm

Description: Returns `true` if the file was loaded as readonly; otherwise `false` is returned.

Get:
```
virtual Boolean isReadOnly () const;
```

rect (IRectangle) — IWindow

Description: Returns a rectangle representing the position and size of the window.

Get:
```
virtual IRectangle rect () const;
```

Set:
```
virtual IWindow& moveSizeTo (const IRectangle& rect);
```

redMix (unsigned char) — IColor

Description: Returns the value of the red component.

Get:
```
unsigned char redMix () const;
```

Set:
```
IColor& setRed (unsigned char redMix);
```

refreshOn (Boolean) — IContainerControl

Description: Queries the refresh state of the container.

Get:
```
Boolean isRefreshOn () const;
```

Set:
```
virtual IContainerControl& setRefreshOn
(Boolean refreshOn=true);
```

refreshOn (Boolean) — IContainerObject

Description: If this object is in a refreshable state, `true` is returned.

Get:
```
virtual Boolean isRefreshOn () const;
```

Set:
```
virtual IContainerObject& setRefreshOn
(Boolean refreshOn=true);
```

relativeWindowRect (IRectangle) — IFlyText

Description: This function returns the rectangle of the control which the IFlyText control is positioned relative to.

Get:
```
virtual IRectangle relativeWindowRect () const;
```

Set:
```
virtual IFlyText& setRelativeWindowRect
(const IRectangle& relativeWindowRect);
```

requiresFiles (Boolean) — IMMDevice — vbmm

Description: Returns true if the device requires the use of files.

Get:
```
Boolean requiresFiles (CallType call=IMMDevice::wait) const;
```

resourceLibrary (IResourceLibrary) — IFlyOverHelpHandler

Description: Returns a reference to the library being used.

Get:
```
virtual IResourceLibrary& resourceLibrary () const;
```

Set:
```
virtual IFlyOverHelpHandler& setResourceLibrary
(const char* resourceLibrary);
```

resourceLibrary (IResourceLibrary) — IInfoArea

Description: Returns a reference to the library being used.

Get:
```
virtual IResourceLibrary& resourceLibrary () const;
```

Set:
```
virtual IInfoArea& setResourceLibrary
(const char* resourceLibrary);
```

ID:
```
resourceLibraryId
```

resourceLibrary (IResourceLibrary) — IResourceId

Description: Returns a const reference to the resource library.

Get:
```
const IResourceLibrary& resourceLibrary () const;
```

restoreRect (IRectangle) — IFrameWindow

Description: Returns the window rectangle coordinates to which window will be restored.

Get:
```
virtual IRectangle restoreRect () const;
```

Set:
```
virtual IFrameWindow& setRestoreRect
(const IRectangle& restoreRect);
```

result •rightCenter

result (unsigned long) — IFrameWindow

Description: Returns the frame result value.
Get:
```
virtual unsigned long result () const;
```
Set:
```
virtual IFrameWindow& setResult (unsigned long result);
```
ID:
```
closeId
```

returnValue (long) — IVBFileDialog

Description: Returns the return code, if an error occurred.
Get:
```
long returnValue () const;
```

returnValue (long) — IVBFontDialog

Description: Returns the return code, if an error occurred.
Get:
```
long returnValue () const;
```

reversed (IPointArray) — IPointArray

Description: Returns a copy of the point array with its elements reversed.
Get:
```
IPointArray reversed () const;
```

rewindButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the rewind animated button.
Get:
```
IAnimatedButton* rewindButton () const;
```

ribbonStripEnabled (Boolean) — IProgressIndicator

Description: If the ribbonStrip style is set, `true` is returned.
Get:
```
Boolean isRibbonStripEnabled () const;
```
Set:
```
virtual IProgressIndicator& enableRibbonStrip
(Boolean ribbonStripEnabled=true);
```

right (IRectangle::Coord) — IRectangle

Description: Returns the X-coordinate of the vertical line that forms the right side of the rectangle.
Get:
```
Coord right () const;
```

rightCenter (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the right-center point of the rectangle.
Get:
```
IPoint rightCenter () const;
```

rotationIncrement (unsigned long) — ICircularSlider

Description: Returns the increment that the arm is rotated.

Get:
```
unsigned long rotationIncrement () const;
```

Set:
```
ICircularSlider& setRotationIncrement
(unsigned long rotationIncrement);
```

samplesPerSecond (unsigned long) — IMMConfigurableAudio — vbmm

Description: Returns the number of samples-per-second played or recorded.

Get:
```
unsigned long samplesPerSecond (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMConfigurableAudio& setSamplesPerSecond
(unsigned long samplesPerSecond,
CallType call=IMMDevice::wait);
```

scrollableRange (IRange) — IScrollBar

Description: Returns the range that the scroll bar represents.

Get:
```
virtual IRange scrollableRange () const;
```

Set:
```
virtual IScrollBar& setScrollableRange
(const IRange& scrollableRange);
```

scrollBoxPosition (unsigned long) — IScrollBar

Description: Returns the current position of the scroll box.

Get:
```
virtual unsigned long scrollBoxPosition () const;
```

Set:
```
virtual IScrollBar& moveScrollBoxTo
(unsigned long scrollBoxPosition=1);
```

ID:
```
scrollBoxPositionId
```

scrollBoxRange (IRange) — IScrollBar

Description: Returns the range in which the scroll box can be positioned.

Get:
```
virtual IRange scrollBoxRange () const;
```

searchListWindow (IWindowHandle) — IHelpWindow

Description: Returns the handle of the Search window.

Get:
```
IWindowHandle searchListWindow () const;
```

seconds (unsigned long) — IMMTime — vbmm

Description: Returns the seconds component of the time.

Get:
```
virtual unsigned long seconds () const;
```

seconds (unsigned) — ITime

Description: Returns the number of seconds past the minute.
Get:
```
unsigned seconds () const;
```

selectable (Boolean) — IMenuItem

Description: If the menu item is selectable, returns `true`.
Get:
```
Boolean isSelectable () const;
```
Set:
```
IMenuItem& setSelectable (Boolean selectable=true);
```

selected (Boolean) — ISettingButton

Description: If the button is selected, `true` is returned.
Get:
```
Boolean isSelected () const;
```
Set:
```
virtual ISettingButton& select (Boolean selected=true);
```
ID:
```
selectId
```

selectedCnrElement (CnrElement*) — IVBContainerControl

Description: Use selectedCnrItem instead of this attribute. Provided for compatibility with V3.
Get:
```
virtual CnrElement* selectedCnrObject ();
```
ID:
```
selectId
```

selectedCnrItem (CnrElement*) — IVBContainerControl

Description: Returns the item corresponding to the first selected container object in the container.
Get:
```
virtual CnrElement* selectedCnrObject ();
```
ID:
```
selectId
```

selectedCnrItems (IVSequence<CnrElement*>*) — IVBContainerControl

Description: Returns a collection of the selected container objects.
Get:
```
virtual IVSequence<CnrElement*>* selectedCnrObjects ();
```
ID:
```
selectId
```

selectedCollectionPosition (unsigned long) — ICollectionViewComboBox

Description: Returns the collection position corresponding to the selected item in the combination box.

Get:
```
virtual unsigned long selectedCollectionPosition ();
```

Set:
```
virtual ICollectionViewComboBox<Element,Collection>& select
(unsigned long collectionPosition, Boolean select=true);
```

ID:
```
selectId
```

selectedCollectionPosition (unsigned long) — ICollectionViewListBox

Description: Returns the collection position corresponding to the first selected item in the list box.

Get:
```
virtual unsigned long selectedCollectionPosition ();
```

Set:
```
virtual ICollectionViewListBox<Element,Collection>& select
(unsigned long collectionPosition, Boolean select=true);
```

ID:
```
selectId
```

selectedCollectionPosition (unsigned long) — IVBContainerControl

Description: Returns the collection position corresponding to the first selected item in the container.

Get:
```
virtual unsigned long selectedCollectionPosition ();
```

Set:
```
virtual IVBContainerControl<Element,Collection,CnrElement>&
select (unsigned long collectionPosition,
Boolean select=true);
```

ID:
```
selectId
```

selectedCollectionPositions (IVSequence<unsigned long>*) — IVBContainerControl

Description: Returns a collection of the indexes of all selected elements.

Get:
```
virtual IVSequence<unsigned long>* selectedCollectionPositions
();
```

ID:
```
selectId
```

selectedElement (Element) — ICollectionViewComboBox

Description: Returns the collection element corresponding to the selected item in the combination box.

Get:
```
virtual Element selectedElement () const;
```

ID: `selectId`

selectedElement (Element) — ICollectionViewListBox

Description: Returns the collection element corresponding to the first selected item in the list box.

Get:
```
virtual Element selectedElement () const;
```

ID: `selectId`

selectedElement (Element) — IVBContainerControl

Description: Use selectedItem instead of this attribute. Provided for compatibility with V3.

Get:
```
virtual Element selectedElement ();
```

ID: `selectId`

selectedFileCount (unsigned long) — IVBFileDialog

Description: Returns the number of files selected by the user.

Get:
```
unsigned long selectedFileCount () const;
```

selectedIndex (unsigned long) — IRadioButton

Description: Returns the 0-based index of the selected radio button in a group.

Get:
```
unsigned long selectedIndex () const;
```

selectedItem (Element) — IVBContainerControl

Description: Returns the item corresponding to the first selected item in the container.

Get:
```
virtual Element selectedElement ();
```

ID: `selectId`

selectedItems (IVSequence<Element>*) — IVBContainerControl

Description: Returns a collection of the selected items.
Get: `virtual IVSequence<Element>* selectedElements ();`
ID: `selectId`

selectedRange (IRange) — IEntryField

Description: Returns the range of the selected text.
Get: `IRange selectedRange () const;`

selectedRange (IRange) — IMultiLineEdit

Description: Returns the range of the selected text.
Get: `virtual IRange selectedRange () const;`

selectedText (IString) — IEntryField

Description: Returns the selected text string.
Get: `IString selectedText () const;`

selectedText (IString) — IMultiLineEdit

Description: Returns the selected text string.
Get: `virtual IString selectedText () const;`

selectedTextLength (unsigned long) — IEntryField

Description: Returns the size of the selected area, in bytes.
Get: `virtual unsigned long selectedTextLength () const;`

selectedTextLength (unsigned long) — IMultiLineEdit

Description: Returns the size of the selected area, in bytes.
Get: `virtual unsigned long selectedTextLength () const;`

selection (long) — IBaseComboBox

Description: Returns the 0-based index of the selected item.

Get:
```
virtual long selection () const;
```

Set:
```
virtual IBaseComboBox& select (unsigned long selection,
Boolean select=true);
```

ID:
```
selectId
```

selection (long) — IBaseListBox

Description: Returns the 0-based index of the selected item.

Get:
```
virtual long selection () const;
```

Set:
```
virtual IBaseListBox& select (unsigned long selection,
Boolean select=true);
```

ID:
```
selectId
```

selection (long) — ICollectionViewComboBox

Description: Returns the 0-based index of the selected item.

Get:
```
virtual long selection () const;
```

ID:
```
selectId
```

selection (long) — ICollectionViewListBox

Description: Read only attribute returning the 0-based index of the selected item.

Get:
```
virtual long selection () const;
```

ID:
```
selectId
```

selectionText (IString) — IBaseListBox

Description: Returns the itemText using selection as the index.

Get:
```
VBSELECTIONTEXT ();
```

ID:
```
selectId
```

separator (Boolean) — IMenuItem

Description: If the style separator is set, returns `true`.

Get:
```
Boolean isSeparator () const;
```

servant (Boolean) — IBaseSpinButton

Description: If the spin button is a servant, `true` is returned.

Get:
```
Boolean isServant () const;
```

shadowColor (IColor) — IWindow

Description: Returns the shadow color value of the window area, or the default if no color for the area has been set.

Get:
```
virtual IColor shadowColor () const;
```

Set:
```
virtual IWindow& setShadowColor (const IColor& shadowColor);
```

ID:
```
shadowColorId
```

shaftPosition (IPoint) — IProgressIndicator

Description: Returns the origin point of the shaft relative to the origin corner of the progress indicator window.

Get:
```
virtual IPoint shaftPosition () const;
```

Set:
```
virtual IProgressIndicator& setShaftPosition
(const IPoint& shaftPosition);
```

shaftSize (ISize) — IProgressIndicator

Description: Returns the size of the shaft in pixels.

Get:
```
virtual ISize shaftSize () const;
```

showing (Boolean) — IWindow

Description: If any part of the window is showing, `true` is returned.

Get:
```
Boolean isShowing () const;
```

showingMiniIcons (Boolean) — IContainerControl

Description: Queries whether the container mode is set for displaying mini icons.

Get:
```
Boolean isShowingMiniIcons () const;
```

singleSelection (Boolean) — IContainerControl

Description: Queries whether the container is in the single selection mode.

Get:
```
Boolean isSingleSelection () const;
```

size (unsigned long) — IPointArray

Description: Returns the dimension of the array.

Get:
```
unsigned long size () const;
```

size (unsigned long) — IRecord — vbsample

Description: Return the size attribute.

Get:
```
unsigned long size () const;
```

size (ISize) — IRectangle

Description: Returns the ISize(width, height).

Get:
```
ISize size () const;
```

Set:
```
IRectangle& sizeTo (const IPair& size);
```

size (unsigned) — IString

Description: Returns the length of the string, not counting the terminating NULL character.

Get:
```
unsigned size () const;
```

size (ISize) — IWindow

Description: Returns the size of the window in window coordinates.

Get:
```
virtual ISize size () const;
```

Set:
```
virtual IWindow& sizeTo (const ISize& size);
```

ID:
```
sizeId
```

sizeToGraphic (Boolean) — IGraphicPushButton

Description: If the style sizeToGraphic is set, true is returned.

Get:
```
Boolean isSizeToGraphic () const;
```

Set:
```
virtual IGraphicPushButton& enableSizeToGraphic
(Boolean sizeToGraphic=true);
```

snapToTickEnabled (Boolean) — IProgressIndicator

Description: If the snapToTickMark style is set, true is returned.

Get:
```
Boolean isSnapToTickEnabled () const;
```

Set:
```
virtual IProgressIndicator& enableSnapToTick
(Boolean snapToTickEnabled=true);
```

sourceRectangle (IRectangle) — IMMDigitalVideo — vbmm

Description: Returns the subrectangle of the video from the currently loaded video file.

Get:
```
IRectangle sourceRectangle (CallType call=IMMDevice::wait)
const;
```

speed (IMMSpeed) — IMMDigitalVideo — vbmm

Description: Returns the currently set speed for the device.

Get:
```
IMMSpeed speed (CallType call=IMMDevice::wait) const;
```

speed (unsigned long) — IMMSpeed — vbmm

Description: Returns the speed value as either a percentage or in frames-per-second.

Get:
```
virtual unsigned long speed () const;
```

speedFormat (IMMSpeed::Format) — IMMDevice — vbmm

Description: Returns the currently set speed format for the device.

Get:
```
IMMSpeed::Format speedFormat (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMDevice& setSpeedFormat
(IMMSpeed::Format speedFormat, CallType call=IMMDevice::wait);
```

spinFieldValid (Boolean) — INumericSpinButton

Description: If the contents of the spin field are within the number range, `true` is returned.

Get:
```
virtual Boolean isSpinFieldValid (Boolean caseSensitive=false)
const;
```

spinFieldValid (Boolean) — ITextSpinButton

Description: If the contents of the spin field matches one of the text values in the text array, `true` is returned.

Get:
```
virtual Boolean isSpinFieldValid (Boolean caseSensitive=false)
const;
```

splitBarEdgeColor (IColor) — ISplitCanvas

Description: Returns the color of the edges of the canvas's split bars.

Get:
```
virtual IColor splitBarEdgeColor () const;
```

Set:
```
virtual ISplitCanvas& setSplitBarEdgeColor
(const IColor& splitBarEdgeColor);
```

splitBarMiddleColor (IColor) — ISplitCanvas

Description: Returns the color of the interior of the canvas's split bars.

Get:
```
virtual IColor splitBarMiddleColor () const;
```

Set:
```
virtual ISplitCanvas& setSplitBarMiddleColor
(const IColor& splitBarMiddleColor);
```

splitBarOffset (unsigned long) — IContainerControl

Description: Returns the offset of the split bar, in pixels, from the left side of the container window.

Get:
```
unsigned long splitBarOffset () const;
```

standardBitmapSize (ISize) — IToolBarButton

Description: Returns the size of the area reserved for a bitmap for a standard tool bar button when the button's bitmap is visible.

Get:
```
static ISize standardBitmapSize ();
```

Set:
```
static void setStandardBitmapSize
(const ISize& standardBitmapSize=ISize ( 20 , 17 ));
```

standardFormat (Boolean) — IToolBarButton

Description: Returns `true` if the button has a style of standardFormat.

Get:
```
Boolean isStandardFormat () const;
```

standardTextLines (unsigned long) — IToolBarButton

Description: Returns the number of lines of text displayed for a standard tool bar button when the button's text is visible.

Get:
```
static unsigned long standardTextLines ();
```

Set:
```
static void setStandardTextLines
(unsigned long standardTextLines=1);
```

standardTextWidth (unsigned long) — IToolBarButton

Description: Returns the width of a standard tool bar button when the text is visible.

Get:
```
static unsigned long standardTextWidth ();
```

Set:
```
static void setStandardTextWidth
(unsigned long standardTextWidth=50);
```

state (IString) — IAddress — vbsample

Description: Query the state (`IString`) attribute.

Get:
```
virtual IString state () const;
```

Set:
```
virtual IAddress& setState (const IString& state);
```

ID:
```
stateId
```

statusTextAlignment (INotebook::TextAlignment) — INotebook

Description: Returns the alignment of the text in the notebook's status line.

Get:
```
virtual TextAlignment statusTextAlignment () const;
```

Set:
```
virtual INotebook& setStatusTextAlignment
(TextAlignment statusTextAlignment);
```

stepBackwardButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the step backward animated button.
Get:
```
IAnimatedButton* stepBackwardButton () const;
```

stepForwardButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the step forward animated button.
Get:
```
IAnimatedButton* stepForwardButton () const;
```

stopButton (IAnimatedButton*) — IMMPlayerPanel — vbmm

Description: Returns a pointer to the stop animated button.
Get:
```
IAnimatedButton* stopButton () const;
```

street (IString) — IAddress — vbsample

Description: Query the street (IString) attribute.
Get:
```
virtual IString street () const;
```
Set:
```
virtual IAddress& setStreet (const IString& street);
```
ID:
```
streetId
```

strikeout (Boolean) — IFont

Description: If the IFont object uses a font with a line drawn through the characters, true is returned.
Get:
```
Boolean isStrikeout () const;
```
Set:
```
virtual IFont& setStrikeout (Boolean strikeout=true);
```

strikeout (Boolean) — IStaticText

Description: Queries whether the text has the style strikeout.
Get:
```
Boolean isStrikeout () const;
```
Set:
```
virtual IStaticText& enableStrikeout (Boolean strikeout=true);
```
ID:
```
strikeoutId
```

string (Boolean) — IContainerColumn

Description: If the data in the column is a pointer to a character string or an IString, true is returned.
Get:
```
Boolean isString () const;
```

stringGenerator (IStringGenerator<Element>) — ICollectionViewComboBox

Description: Retrieves the string generator associated with the collection view combination box.

Get:
```
virtual IStringGenerator<Element>& stringGenerator ();
```

Set:
```
virtual IStringGenerator<Element>& setStringGenerator
(const IStringGenerator<Element>& stringGenerator);
```

stringGenerator (IStringGenerator<Element>) — ICollectionViewListBox

Description: Retrieves the string generator associated with the collection view list box.

Get:
```
virtual IStringGenerator<Element>& stringGenerator ();
```

Set:
```
virtual IStringGenerator<Element>& setStringGenerator
(const IStringGenerator<Element>& stringGenerator);
```

stringTableOffset (long) — IInfoArea

Description: Returns the offset currently in use.

Get:
```
virtual long stringTableOffset () const;
```

Set:
```
virtual IInfoArea& setStringTableOffset
(long stringTableOffset=0);
```

submenu (Boolean) — IMenuItem

Description: Returns true if this item is a submenu.

Get:
```
Boolean isSubmenu () const;
```

submenuHandle (IWindowHandle) — IMenuItem

Description: Returns the submenu window handle for the item.

Get:
```
IWindowHandle submenuHandle () const;
```

Set:
```
IMenuItem& setSubmenuHandle
(const IWindowHandle& submenuHandle);
```

supportsAudio (Boolean) — IMMDevice — vbmm

Description: Returns true if the device has support for audio playback.

Get:
```
Boolean supportsAudio (CallType call=IMMDevice::wait) const;
```

supportsDigitalTransfer (Boolean) — IMMDevice — vbmm

Description: Returns true if the device can internally process digital data, for example a digital-to-analog converter on a CD player.

Get:
```
Boolean supportsDigitalTransfer
(CallType call=IMMDevice::wait) const;
```

supportsDisableEject (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device can disable the manual ejection of the media.

Get:
```
Boolean supportsDisableEject (CallType call=IMMDevice::wait)
const;
```

supportsEject (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device can eject the media.

Get:
```
Boolean supportsEject (CallType call=IMMDevice::wait) const;
```

supportsOverlayGraphics (Boolean) — IMMDigitalVideo — vbmm

Description: Returns `true` if the device supports the display by an application of overlay graphics in a video window.

Get:
```
Boolean supportsOverlayGraphics
(CallType call=IMMDevice::wait) const;
```

supportsPlay (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device can play.

Get:
```
Boolean supportsPlay (CallType call=IMMDevice::wait) const;
```

supportsRecord (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device supports recording.

Get:
```
Boolean supportsRecord (CallType call=IMMDevice::wait) const;
```

supportsRecordInsertion (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device supports insertion of data while recording.

Get:
```
Boolean supportsRecordInsertion
(CallType call=IMMDevice::wait) const;
```

supportsReverse (Boolean) — IMMDigitalVideo — vbmm

Description: Returns `true` if the device can play in reverse.

Get:
```
Boolean supportsReverse (CallType call=IMMDevice::wait) const;
```

supportsSave (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device can save data to some type of media (for example a file).

Get:
```
Boolean supportsSave (CallType call=IMMDevice::wait) const;
```

supportsSizing (Boolean) — IMMDigitalVideo — vbmm

Description: Returns `true` if the device can independently stretch the horizontal and vertical dimensions of the image.

Get:
```
Boolean supportsSizing (CallType call=IMMDevice::wait) const;
```

supportsStreaming (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device can continuously transfer digital data to or from another device.

Get:
```
Boolean supportsStreaming (CallType call=IMMDevice::wait)
const;
```

supportsStretchToFit (Boolean) — IMMDigitalVideo — vbmm

Description: Returns `true` if the device can stretch the video frames to fill the display rectangle.

Get:
```
Boolean supportsStretchToFit (CallType call=IMMDevice::wait)
const;
```

supportsVideo (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device has support for video playback.

Get:
```
Boolean supportsVideo (CallType call=IMMDevice::wait) const;
```

supportsVolumeAdjustment (Boolean) — IMMDevice — vbmm

Description: Returns `true` if the device supports software control of the volume.

Get:
```
Boolean supportsVolumeAdjustment
(CallType call=IMMDevice::wait) const;
```

systemCommand (ICommand::CommandId) — IAcceleratorKey

Description: Returns the system command that this accelerator key runs.

Get:
```
ICommand::CommandId commandId () const;
```

Set:
```
IAcceleratorKey& setSystemCommand
(ICommand::CommandId commandId);
```

systemCommand (Boolean) — IPushButton

Description: If the style systemCommand is set, `true` is returned.

Get:
```
Boolean isSystemCommand () const;
```

Set:
```
virtual IPushButton& enableSystemCommand
(Boolean systemCommand=true);
```

ID: `systemCommandId`

systemProfile (IProfile::IProfile) — IProfile

Description: Returns the system profile.

Get:
```
static IProfile systemProfile ();
```

systemScrollBarWidth (unsigned long) — IScrollBar

Description: Returns the system width of a vertical or horizontal scroll bar.

Get:
```
static unsigned long systemScrollBarWidth
(Boolean verticalScrollBar=true);
```

systemScrollBoxLength (unsigned long) — IScrollBar

Description: Returns the system length of a scroll box for a vertical or horizontal scroll bar.

Get:
```
static unsigned long systemScrollBoxLength
(Boolean verticalScrollBar=true);
```

systemScrollButtonLength (unsigned long) — IScrollBar

Description: Returns the system length of a scroll button for a vertical or horizontal scroll bar.

Get:
```
static unsigned long systemScrollButtonLength
(Boolean verticalScrollBar=true);
```

tabShape (INotebook::TabShape) — INotebook

Description: Returns the shape of the notebook's tabs.

Get:
```
virtual TabShape tabShape () const;
```

Set:
```
virtual INotebook& setTabShape (TabShape tabShape);
```

tabStop •text

tabStop (Boolean) — IControl

Description: If the control has the tabStop style set, `true` is returned.
Get:
```
virtual Boolean isTabStop () const;
```
Set:
```
virtual IControl& enableTabStop (Boolean tabStop=true);
```

tabStop (Boolean) — IWindow

Description: If the user can tab to the window, `true` is returned.
Get:
```
virtual Boolean isTabStop () const;
```

tabTextAlignment (INotebook::TextAlignment) — INotebook

Description: Returns the alignment of the text in the notebook's tabs.
Get:
```
virtual TextAlignment tabTextAlignment () const;
```
Set:
```
virtual INotebook& setTabTextAlignment
(TextAlignment tabTextAlignment);
```

text (const char*) — ICLibErrorInfo

Description: Returns the error text.
Get:
```
virtual const char* text () const;
```

text (IString) — IClipboard

Description: Returns textFormat data as an `IString` object.
Get:
```
virtual IString text ();
```
Set:
```
virtual IClipboard& setText (const char* text);
```

text (const char*) — IGUIErrorInfo

Description: Returns the error text.
Get:
```
virtual const char* text () const;
```

text (IVBMnemonicString) — IMenuItem

Description: Returns an `IString` representing the text to be displayed.
Get:
```
IString text () const;
```
Set:
```
IMenuItem& setText (const char* text);
```

text (const char*) — IMMErrorInfo — vbmm

Description: Returns the error text.
Get:
```
virtual const char* text () const;
```

text (IVBMnemonicString) — ISetCanvas

Description: Returns the text of the group box.
Get: `virtual IString text () const;`
Set: `virtual ISetCanvas& setText (const char* text);`
ID: `textId`

text (const char*) — ISystemErrorInfo

Description: Returns the error text.
Get: `virtual const char* text () const;`

text (IVBMnemonicString) — ITextControl

Description: Returns the control window's text.
Get: `virtual IString text () const;`
Set: `virtual ITextControl& setText (const char* text);`
ID: `textId`

text (IString) — ITextSpinButton

Description: Returns the displayed contents of the spin field.
Get: `virtual IString text () const;`
Set: `virtual ITextSpinButton& setText (const char* text);`
ID: `textId`

text (IString) — IVBStringPart — vbsample

Description: Query the text (`IString`) attribute.
Get: `virtual IString text () const;`
Set: `virtual IVBStringPart& setText (const IString& text);`
ID: `textId`

textAsLowerCase (IString) — IVBStringPart — vbsample

Description: Query the textAsLowerCase (`IString`) attribute.
Get: `virtual IString textAsLowerCase () const;`
ID: `textId`

textAsUpperCase (IString) — IVBStringPart — vbsample

Description: Query the textAsUpperCase (IString) attribute.
Get: `virtual IString textAsUpperCase () const;`
ID: `textId`

textEqualDefault (Boolean) — IVBStringPart — vbsample

Description: Query the textEqualDefault (Boolean) attribute.
Get: `virtual Boolean isTextEqualDefault () const;`
ID: `textId`

textLength (unsigned long) — ITextControl

Description: Returns the current length of the control window's text in bytes.
Get: `virtual unsigned long textLength () const;`
ID: `textId`

textLength (unsigned) — ITextSpinButton

Description: Return length from the IString text attribute
Get: `VBLENGTHFROMTEXT ();`
ID: `textId`

textLength (unsigned long) — IVBStringPart — vbsample

Description: Query the textLength (unsigned long) attribute.
Get: `virtual unsigned long textLength () const;`
ID: `textLengthId`

textNotEqualDefault (Boolean) — IVBStringPart — vbsample

Description: Query the textNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isTextNotEqualDefault () const;`
ID: `VBANYEVENT`

textView (Boolean) — IContainerControl

Description: Queries whether the container is currently in a text view.
Get: `Boolean isTextView (Boolean textOnly=false) const;`

textVisible (Boolean) — IToolBarButton

Description: Returns true if the tool bar button text is visible.
Get: `Boolean isTextVisible () const;`

thousandths (unsigned long) — IMMTime — vbmm

Description: Returns the millisecond component of the time rounded to the nearest thousandth.

Get:
```
virtual unsigned long thousandths () const;
```

tickSpacing (unsigned long) — ICircularSlider

Description: Returns the increment used to draw the tick marks.

Get:
```
unsigned long tickSpacing () const;
```

Set:
```
ICircularSlider& setTickSpacing (unsigned long tickSpacing);
```

time (Boolean) — IContainerColumn

Description: If the data in the column is a time, `true` is returned.

Get:
```
Boolean isTime () const;
```

time (ITime) — ICustomer — vbsample

Description: Query the time (`ITime`) attribute.

Get:
```
virtual ITime time () const;
```

Set:
```
virtual ICustomer& setTime (const ITime& time);
```

ID: `timeId`

timeFormat (IMMTime::Format) — IMMDevice — vbmm

Description: Returns the currently set time format for the device.

Get:
```
IMMTime::Format timeFormat (CallType call=IMMDevice::wait)
const;
```

Set:
```
virtual IMMDevice& setTimeFormat (IMMTime::Format timeFormat,
CallType call=IMMDevice::wait);
```

title (IString) — IContainerControl

Description: Returns the title of the container.
Get:
```
virtual IString title () const;
```
Set:
```
virtual IContainerControl& setTitle (const char* title);
```
ID:
```
titleId
```

titleRectangle (IRectangle) — IContainerControl

Description: Returns the rectangle that contains the container's title.
Get:
```
IRectangle titleRectangle () const;
```

titleSeparatorVisible (Boolean) — IContainerControl

Description: Queries whether the title separator is currently displayed.
Get:
```
Boolean isTitleSeparatorVisible () const;
```
Set:
```
virtual IContainerControl& showTitleSeparator
(Boolean titleSeparatorVisible=true);
```

titleVisible (Boolean) — IContainerControl

Description: Queries whether the container title is currently displayed.
Get:
```
Boolean isTitleVisible () const;
```
Set:
```
virtual IContainerControl& showTitle
(Boolean titleVisible=true);
```
ID:
```
titleVisibleId
```

titleWriteable (Boolean) — IContainerControl

Description: Returns `true` if the data in the title can be edited.
Get:
```
Boolean isTitleWriteable () const;
```
Set:
```
virtual IContainerControl& enableTitleUpdate
(Boolean titleWriteable=true);
```

today (IDate) — IDate

Description: Returns the current date.
Get:
```
static IDate today ();
```

toolBarContainer (IToolBarContainer*) — IToolBar

Description: Returns a pointer to the IToolBarContainer object that contains this tool bar.
Get:
```
virtual IToolBarContainer* toolBarContainer () const;
```

toolBarList (IToolBarList*) — IFrameWindow

Description: Returns the address of the list of tool bars used in this frame window.

Get:
```
IToolBarList* toolBarList () const;
```

Set:
```
IFrameWindow& setToolBarList (IToolBarList* toolBarList);
```

top (unsigned long) — IBaseComboBox

Description: Returns the item number of the item currently at the top of the list box.

Get:
```
virtual unsigned long top () const;
```

Set:
```
virtual IBaseComboBox& setTop (unsigned long top);
```

top (unsigned long) — IBaseListBox

Description: Returns the item number of the item currently at the top of the list box.

Get:
```
virtual unsigned long top () const;
```

Set:
```
virtual IBaseListBox& setTop (unsigned long top);
```

top (unsigned long) — IMultiLineEdit

Description: Returns the line number of the line currently visible at the top of the screen.

Get:
```
unsigned long top () const;
```

Set:
```
virtual IMultiLineEdit& setTop (unsigned long top);
```

top (IRectangle::Coord) — IRectangle

Description: Returns the Y-coordinate of the horizontal line that forms the top of the rectangle.

Get:
```
Coord top () const;
```

top (Element const) — IRStack — vbcc

Description: Returns a reference to the last element of the collection.

Get:
```
Element const& top () const;
```

topCenter (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the top-center point of the rectangle.

Get:
```
IPoint topCenter () const;
```

topLeft (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the top-left corner of the rectangle.

Get:
```
IPoint topLeft () const;
```

topPage (IPageHandle) — INotebook

Description: Returns an IPageHandle object that references the current top page of the notebook.

Get:
```
virtual IPageHandle topPage () const;
```

topRight (IPoint) — IRectangle

Description: Returns the X- and Y-coordinates of the top-right corner of the rectangle.

Get:
```
IPoint topRight () const;
```

totalPages (unsigned long) — INotebook

Description: Returns the total number of pages in the notebook.

Get:
```
virtual unsigned long totalPages () const;
```

traceDestination (ITrace::Destination) — ITrace

Description: Returns the trace output destination for this trace object.

Get:
```
static ITrace::Destination traceDestination ();
```

traceEnabled (Boolean) — ITrace

Description: Determines whether tracing is currently enabled.

Get:
```
static Boolean isTraceEnabled ();
```

track (unsigned long) — IMMTrackMinSecFrameTime — vbmm

Description: Returns the track component of the time.

Get:
```
virtual unsigned long track () const;
```

transparentColor (IColor) — IToolBarButton

Description: Returns the current transparent color.

Get:
```
virtual IColor transparentColor () const;
```

Set:
```
virtual IToolBarButton& setTransparentColor
(const IColor& transparentColor=IColor::pink);
```

treble (unsigned long) — IMMAmpMixer — vbmm

Description: Returns the treble, where 0 is the least amount of treble and 100 is the most amount of treble.

Get:
```
unsigned long treble (CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMAmpMixer& setTreble (unsigned long treble,
CallType call=IMMDevice::wait);
```

treeIconView (Boolean) — IContainerControl

Description: Returns `true` only if the container is in tree-icon view.

Get:
```
Boolean isTreeIconView () const;
```

treeNameView (Boolean) — IContainerControl

Description: Returns `true` only if the container is currently in tree-name view.

Get:
```
Boolean isTreeNameView () const;
```

treeTextView (Boolean) — IContainerControl

Description: Returns `true` only if the container is currently in tree-text view.

Get:
```
Boolean isTreeTextView () const;
```

treeView (Boolean) — IContainerControl

Description: Queries whether the container is currently in a tree view.

Get:
```
Boolean isTreeView () const;
```

type (IBaseComboBox::ControlType) — IBaseComboBox

Description: Returns the ControlType enumerator for the type of combination box.

Get:
```
ControlType type () const;
```

underscore (Boolean) — IFont

Description: If the `IFont` object uses a font with a line drawn under the characters, `true` is returned.

Get:
```
Boolean isUnderscore () const;
```

Set:
```
virtual IFont& setUnderscore (Boolean underscore=true);
```

underscore (Boolean) — IStaticText

Description: Queries whether the text has the style underscore.

Get:
```
Boolean isUnderscore () const;
```

Set:
```
virtual IStaticText& enableUnderscore
(Boolean underscore=true);
```

ID: `underscoreId`

undoable (Boolean) — IMultiLineEdit

Description: Queries whether any undoable actions have been performed on the contents of the MLE.

Get:
```
Boolean isUndoable () const;
```

upc (IString) — IMMAudioCD — vbmm

Description: Returns the disc's universal product code (serial number).

Get:
```
IString upc (CallType call=IMMDevice::wait) const;
```

upperBound (Coord) — IRange

Description: Returns the upper bound of the range.

Get:
```
Coord upperBound () const;
```

Set:
```
IRange& setUpperBound (Coord upperBound);
```

upperCase (Boolean) — IVBStringPart — vbsample

Description: Query the upperCase (`Boolean`) attribute.

Get:
```
virtual Boolean isUpperCase () const;
```

ID: `textId`

userData (unsigned long) — ICustomButton

Description: Returns an `unsigned long` value that represents special data associated with the button object.

Get:
```
unsigned long userData () const;
```

Set:
```
ICustomButton& setUserData (unsigned long userData);
```

userProfile (IProfile::IProfile) — IProfile

Description: Returns the user profile.

Get:
```
static IProfile userProfile ();
```

usesDialogBackground (Boolean) — IFrameWindow

Description: Returns whether the frame window uses the system dialog background color rather than the system window background color.

Get:
```
Boolean usesDialogBackground () const;
```

valid (Boolean) — IMMAudioCDContents — vbmm

Description: Returns `true` if the table of contents is valid.

Get:
```
Boolean isValid () const;
```

valid (Boolean) — IMMTime — vbmm

Description: Returns `true` if the time value is valid.

Get:
```
virtual Boolean isValid () const;
```

valid (Boolean) — IWindow

Description: If this object represents a valid window in the window system, `true` is returned.

Get:
```
Boolean isValid () const;
```

validDBCS (Boolean) — IString

Description: If no DBCS characters have a 0 second byte, `true` is returned.

Get:
```
Boolean isValidDBCS () const;
```

validMBCS (Boolean) — IString

Description: If no MBCS characters have a 0 second byte, `true` is returned.

Get:
```
Boolean isValidMBCS () const;
```

value

value (long) — ICircularSlider

Description: Returns the current value of the circular slider.
Get: `long value () const;`
Set: `ICircularSlider& setValue (long value);`
ID: `valueId`

value (IColor::Color) — IColor

Description: Using a specified color index, determines the Color enumerator.
Get: `Color value () const;`

value (long) — INumericSpinButton

Description: Returns the current value displayed in the spin field.
Get: `virtual long value () const;`
Set: `virtual INumericSpinButton& setValue (long value);`
ID: `valueId`

value (Boolean) — IVBBooleanPart — vbsample

Description: Query the value (Boolean) attribute.
Get: `virtual Boolean value () const;`
Set: `virtual IVBBooleanPart& setValue (Boolean value);`
ID: `valueId`

value (double) — IVBDoublePart — vbsample

Description: Query the value (double) attribute.
Get: `virtual double value () const;`
Set: `virtual IVBDoublePart& setValue (double value);`
ID: `valueId`

value (Boolean) — IVBLogicalAndPart — vbsample

Description: Query the value (Boolean) attribute.
Get: `virtual Boolean value () const;`
ID: `valueId`

value (Boolean) — IVBLogicalOrPart — vbsample

Description: Query the value (Boolean) attribute.
Get: `virtual Boolean value () const;`
ID: `valueId`

value (long) — IVBLongPart — vbsample

Description: Query the value (long) attribute.
Get: `virtual long value () const;`
Set: `virtual IVBLongPart& setValue (long value);`
ID: `valueId`

value (short) — IVBShortPart — vbsample

Description: Query the value (short) attribute.
Get: `virtual short value () const;`
Set: `virtual IVBShortPart& setValue (short value);`
ID: `valueId`

value (unsigned long) — IVBUnsignedLongPart — vbsample

Description: Query the value (unsigned long) attribute.
Get: `virtual unsigned long value () const;`
Set: `virtual IVBUnsignedLongPart& setValue (unsigned long value);`
ID: `valueId`

value (unsigned short) — IVBUnsignedShortPart — vbsample

Description: Query the value (unsigned short) attribute.
Get: `virtual unsigned short value () const;`
Set: `virtual IVBUnsignedShortPart& setValue (unsigned short value);`
ID: `valueId`

valueAboveHighLimit (Boolean) — IVBDoublePart — vbsample

Description: Query the valueAboveHighLimit (Boolean) attribute.
Get: `virtual Boolean isValueAboveHighLimit () const;`
ID: `VBANYEVENT`

valueAboveHighLimit (Boolean) — IVBLongPart — vbsample

Description: Query the valueAboveHighLimit (Boolean) attribute.
Get:
```
virtual Boolean isValueAboveHighLimit () const;
```
ID: VBANYEVENT

valueAboveHighLimit (Boolean) — IVBShortPart — vbsample

Description: Query the valueAboveHighLimit (Boolean) attribute.
Get:
```
virtual Boolean isValueAboveHighLimit () const;
```
ID: VBANYEVENT

valueAboveHighLimit (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueAboveHighLimit (Boolean) attribute.
Get:
```
virtual Boolean isValueAboveHighLimit () const;
```
ID: VBANYEVENT

valueAboveHighLimit (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueAboveHighLimit (Boolean) attribute.
Get:
```
virtual Boolean isValueAboveHighLimit () const;
```
ID: VBANYEVENT

valueAs1Based (double) — IVBDoublePart — vbsample

Description: Query the valueAs1Based (double) attribute.
Get:
```
virtual double valueAs1Based () const;
```
Set:
```
virtual IVBDoublePart& setValueAs1Based
(double valueAs1Based);
```
ID: valueId

valueAs1Based (long) — IVBLongPart — vbsample

Description: Query the valueAs1Based (long) attribute.

Get:
```
virtual long valueAs1Based () const;
```

Set:
```
virtual IVBLongPart& setValueAs1Based (long valueAs1Based);
```

ID:
```
valueId
```

valueAs1Based (short) — IVBShortPart — vbsample

Description: Query the valueAs1Based (short) attribute.

Get:
```
virtual short valueAs1Based () const;
```

Set:
```
virtual IVBShortPart& setValueAs1Based (short valueAs1Based);
```

ID:
```
valueId
```

valueAs1Based (unsigned long) — IVBUnsignedLongPart — vbsample

Description: Query the valueAs1Based (unsigned long) attribute.

Get:
```
virtual unsigned long valueAs1Based () const;
```

Set:
```
virtual IVBUnsignedLongPart& setValueAs1Based
(unsigned long valueAs1Based);
```

ID:
```
valueId
```

valueAs1Based (unsigned short) — IVBUnsignedShortPart — vbsample

Description: Query the valueAs1Based (unsigned short) attribute.

Get:
```
virtual unsigned short valueAs1Based () const;
```

Set:
```
virtual IVBUnsignedShortPart& setValueAs1Based
(unsigned short valueAs1Based);
```

ID:
```
valueId
```

valueAsDouble (double) — ITextControl

Description: Return asDouble from the IString text attribute

Get:
```
VBDOUBLEVALUEFROMTEXT ();
```

Set:
```
VBSETTEXTFROMVALUE (double newValue);
```

ID:
```
textId
```

valueAsDouble (double) — ITextSpinButton

Description: Return asDouble from the IString text attribute
Get: `VBDOUBLEVALUEFROMTEXT ();`
Set: `VBSETTEXTFROMVALUE (double newValue);`
ID: `textId`

valueAsInt (int) — ITextControl

Description: Return asInt from the IString text attribute
Get: `VBINTVALUEFROMTEXT ();`
Set: `VBSETTEXTFROMVALUE (int newValue);`
ID: `textId`

valueAsInt (int) — ITextSpinButton

Description: Return asInt from the IString text attribute
Get: `VBINTVALUEFROMTEXT ();`
Set: `VBSETTEXTFROMVALUE (int newValue);`
ID: `textId`

valueAsText (IString) — IVBBooleanPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get: `virtual IString valueAsText () const;`
Set:
```
virtual IVBBooleanPart& setValueAsText
(const IString* valueAsText);
```
ID: `valueId`

valueAsText (IString) — IVBDoublePart — vbsample

Description: Query the valueAsText (IString) attribute.
Get: `virtual IString valueAsText () const;`
Set:
```
virtual IVBDoublePart& setValueAsText
(const IString& valueAsText);
```
ID: `valueId`

valueAsText (IString) — IVBLogicalAndPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get:
```
virtual IString valueAsText () const;
```
ID:
```
valueId
```

valueAsText (IString) — IVBLogicalOrPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get:
```
virtual IString valueAsText () const;
```
ID:
```
valueId
```

valueAsText (IString) — IVBLongPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get:
```
virtual IString valueAsText () const;
```
Set:
```
virtual IVBLongPart& setValueAsText
(const IString& valueAsText);
```
ID:
```
valueId
```

valueAsText (IString) — IVBShortPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get:
```
virtual IString valueAsText () const;
```
Set:
```
virtual IVBShortPart& setValueAsText
(const IString* valueAsText);
```
ID:
```
valueId
```

valueAsText (IString) — IVBUnsignedLongPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get:
```
virtual IString valueAsText () const;
```
Set:
```
virtual IVBUnsignedLongPart& setValueAsText
(const IString* valueAsText);
```
ID:
```
valueId
```

valueAsText (IString) — IVBUnsignedShortPart — vbsample

Description: Query the valueAsText (IString) attribute.
Get: `virtual IString valueAsText () const;`
Set: `virtual IVBUnsignedShortPart& setValueAsText (const IString* valueAsText);`
ID: `valueId`

valueAsUnsigned (unsigned int) — ITextControl

Description: Return asUnsigned from the IString text attribute
Get: `VBUNSIGNEDVALUEFROMTEXT ();`
Set: `VBSETTEXTFROMVALUE (unsigned newValue);`
ID: `textId`

valueAsUnsigned (unsigned int) — ITextSpinButton

Description: Return asUnsigned from the IString text attribute
Get: `VBUNSIGNEDVALUEFROMTEXT ();`
Set: `VBSETTEXTFROMVALUE (unsigned newValue);`
ID: `textId`

valueBelowLowLimit (Boolean) — IVBDoublePart — vbsample

Description: Query the valueBelowLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueBelowLowLimit () const;`
ID: `VBANYEVENT`

valueBelowLowLimit (Boolean) — IVBLongPart — vbsample

Description: Query the valueBelowLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueBelowLowLimit () const;`
ID: `VBANYEVENT`

valueBelowLowLimit (Boolean) — IVBShortPart — vbsample

Description: Query the valueBelowLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueBelowLowLimit () const;`
ID: `VBANYEVENT`

valueBelowLowLimit (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueBelowLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueBelowLowLimit () const;`
ID: VBANYEVENT

valueBelowLowLimit (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueBelowLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueBelowLowLimit () const;`
ID: VBANYEVENT

valueEqualDefault (Boolean) — IVBBooleanPart — vbsample

Description: Query the valueEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueEqualDefault () const;`
ID: valueId

valueEqualDefault (Boolean) — IVBDoublePart — vbsample

Description: Query the valueEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueEqualDefault () const;`
ID: VBANYEVENT

valueEqualDefault (Boolean) — IVBLongPart — vbsample

Description: Query the valueEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueEqualDefault () const;`
ID: VBANYEVENT

valueEqualDefault (Boolean) — IVBShortPart — vbsample

Description: Query the valueEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueEqualDefault () const;`
ID: VBANYEVENT

valueEqualDefault (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueEqualDefault () const;`
ID: VBANYEVENT

valueEqualDefault (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueEqualDefault () const;`
ID: VBANYEVENT

valueEqualHighLimit (Boolean) — IVBDoublePart — vbsample

Description: Query the valueEqualHighLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualHighLimit () const;`
ID: VBANYEVENT

valueEqualHighLimit (Boolean) — IVBLongPart — vbsample

Description: Query the valueEqualHighLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualHighLimit () const;`
ID: VBANYEVENT

valueEqualHighLimit (Boolean) — IVBShortPart — vbsample

Description: Query the valueEqualHighLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualHighLimit () const;`
ID: VBANYEVENT

valueEqualHighLimit (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueEqualHighLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualHighLimit () const;`
ID: VBANYEVENT

valueEqualHighLimit (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueEqualHighLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualHighLimit () const;`
ID: VBANYEVENT

valueEqualLowLimit (Boolean) — IVBDoublePart — vbsample

Description: Query the valueEqualLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualLowLimit () const;`
ID: VBANYEVENT

valueEqualLowLimit (Boolean) — IVBLongPart — vbsample

Description: Query the valueEqualLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualLowLimit () const;`
ID: VBANYEVENT

valueEqualLowLimit (Boolean) — IVBShortPart — vbsample

Description: Query the valueEqualLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualLowLimit () const;`
ID: VBANYEVENT

valueEqualLowLimit (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueEqualLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualLowLimit () const;`
ID: VBANYEVENT

valueEqualLowLimit (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueEqualLowLimit (Boolean) attribute.
Get: `virtual Boolean isValueEqualLowLimit () const;`
ID: VBANYEVENT

valueNegative (Boolean) — IVBDoublePart — vbsample

Description: Query the valueNegative (Boolean) attribute.
Get: `virtual Boolean isValueNegative () const;`
ID: valueId

valueNegative (Boolean) — IVBLongPart — vbsample

Description: Query the valueNegative (Boolean) attribute.
Get: `virtual Boolean isValueNegative () const;`
ID: valueId

valueNegative (Boolean) — IVBShortPart — vbsample

Description: Query the valueNegative (Boolean) attribute.
Get: `virtual Boolean isValueNegative () const;`
ID: valueId

valueNotEqualDefault (Boolean) — IVBBooleanPart — vbsample

Description: Query the valueNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueNotEqualDefault () const;`
ID: `valueId`

valueNotEqualDefault (Boolean) — IVBDoublePart — vbsample

Description: Query the valueNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueNotEqualDefault () const;`
ID: `VBANYEVENT`

valueNotEqualDefault (Boolean) — IVBLongPart — vbsample

Description: Query the valueNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueNotEqualDefault () const;`
ID: `VBANYEVENT`

valueNotEqualDefault (Boolean) — IVBShortPart — vbsample

Description: Query the valueNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueNotEqualDefault () const;`
ID: `VBANYEVENT`

valueNotEqualDefault (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueNotEqualDefault () const;`
ID: `VBANYEVENT`

valueNotEqualDefault (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueNotEqualDefault (Boolean) attribute.
Get: `virtual Boolean isValueNotEqualDefault () const;`
ID: `VBANYEVENT`

valueNotZero (Boolean) — IVBDoublePart — vbsample

Description: Query the valueNotZero (Boolean) attribute.
Get: `virtual Boolean isValueNotZero () const;`
ID: `valueId`

valueNotZero (Boolean) — IVBLongPart — vbsample

Description: Query the valueNotZero (Boolean) attribute.
Get: `virtual Boolean isValueNotZero () const;`
ID: `valueId`

valueNotZero (Boolean) — IVBShortPart — vbsample

Description: Query the valueNotZero (Boolean) attribute.
Get: `virtual Boolean isValueNotZero () const;`
ID: `valueId`

valueNotZero (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueNotZero (Boolean) attribute.
Get: `virtual Boolean isValueNotZero () const;`
ID: `valueId`

valueNotZero (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueNotZero (Boolean) attribute.
Get: `virtual Boolean isValueNotZero () const;`
ID: `valueId`

valueOutsideLimits (Boolean) — IVBDoublePart — vbsample

Description: Query the valueOutsideLimits (Boolean) attribute.
Get: `virtual Boolean isValueOutsideLimits () const;`
ID: `VBANYEVENT`

valueOutsideLimits (Boolean) — IVBLongPart — vbsample

Description: Query the valueOutsideLimits (Boolean) attribute.
Get: `virtual Boolean isValueOutsideLimits () const;`
ID: `VBANYEVENT`

valueOutsideLimits (Boolean) — IVBShortPart — vbsample

Description: Query the valueOutsideLimits (Boolean) attribute.
Get: `virtual Boolean isValueOutsideLimits () const;`
ID: `VBANYEVENT`

valueOutsideLimits (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueOutsideLimits (Boolean) attribute.
Get: `virtual Boolean isValueOutsideLimits () const;`
ID: `VBANYEVENT`

valueOutsideLimits (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueOutsideLimits (Boolean) attribute.
Get: `virtual Boolean isValueOutsideLimits () const;`
ID: `VBANYEVENT`

valuePositive (Boolean) — IVBDoublePart — vbsample

Description: Query the valuePositive (Boolean) attribute.
Get: `virtual Boolean isValuePositive () const;`
ID: `valueId`

valuePositive (Boolean) — IVBLongPart — vbsample

Description: Query the valuePositive (Boolean) attribute.
Get: `virtual Boolean isValuePositive () const;`
ID: `valueId`

valuePositive (Boolean) — IVBShortPart — vbsample

Description: Query the valuePositive (Boolean) attribute.
Get: `virtual Boolean isValuePositive () const;`
ID: `valueId`

valueWithinLimits (Boolean) — IVBDoublePart — vbsample

Description: Query the valueWithinLimits (Boolean) attribute.
Get: `virtual Boolean isValueWithinLimits () const;`
ID: `VBANYEVENT`

valueWithinLimits (Boolean) — IVBLongPart — vbsample

Description: Query the valueWithinLimits (Boolean) attribute.
Get: `virtual Boolean isValueWithinLimits () const;`
ID: `valueId`

valueWithinLimits (Boolean) — IVBShortPart — vbsample

Description: Query the valueWithinLimits (Boolean) attribute.
Get: `virtual Boolean isValueWithinLimits () const;`
ID: VBANYEVENT

valueWithinLimits (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueWithinLimits (Boolean) attribute.
Get: `virtual Boolean isValueWithinLimits () const;`
ID: VBANYEVENT

valueWithinLimits (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueWithinLimits (Boolean) attribute.
Get: `virtual Boolean isValueWithinLimits () const;`
ID: VBANYEVENT

valueZero (Boolean) — IVBDoublePart — vbsample

Description: Query the valueZero (Boolean) attribute.
Get: `virtual Boolean isValueZero () const;`
ID: valueId

valueZero (Boolean) — IVBLongPart — vbsample

Description: Query the valueZero (Boolean) attribute.
Get: `virtual Boolean isValueZero () const;`
ID: valueId

valueZero (Boolean) — IVBShortPart — vbsample

Description: Query the valueZero (Boolean) attribute.
Get: `virtual Boolean isValueZero () const;`
ID: valueId

valueZero (Boolean) — IVBUnsignedLongPart — vbsample

Description: Query the valueZero (Boolean) attribute.
Get: `virtual Boolean isValueZero () const;`
ID: valueId

valueZero (Boolean) — IVBUnsignedShortPart — vbsample

Description: Query the valueZero (Boolean) attribute.
Get:
```
virtual Boolean isValueZero () const;
```
ID: valueId

vectorOnly (Boolean) — IFont

Description: If the IFont object uses only vector fonts, true is returned.
Get:
```
Boolean isVectorOnly () const;
```

vertical (Boolean) — IProgressIndicator

Description: If the progress indicator is oriented vertically, true is returned.
Get:
```
Boolean isVertical () const;
```

vertical (Boolean) — IScrollBar

Description: If the scroll bar is vertical, true is returned.
Get:
```
Boolean isVertical () const;
```

verticalDataAlignment (IContainerColumn::VerticalAlignment) — IContainerColumn

Description: Returns the current alignment of column data in a container details view.
Get:
```
VerticalAlignment verticalDataAlignment () const;
```

verticalHeadingAlignment (IContainerColumn::VerticalAlignment) — IContainerColumn

Description: Returns the current alignment of column heading data in a container details view.
Get:
```
VerticalAlignment verticalHeadingAlignment () const;
```

verticalScrollBar (IScrollBar*) — IViewPort

Description: Returns the address of the vertical scroll bar.
Get:
```
IScrollBar* verticalScrollBar () const;
```

verticalSeparator (Boolean) — IContainerColumn

Description: If the column heading has a vertical separator drawn after the column, true is returned.
Get:
```
Boolean hasVerticalSeparator () const;
```

videoFileHeight (unsigned long) — IMMDigitalVideo — vbmm

Description: Returns the height of the video for the currently loaded video file.

Get:
```
unsigned long videoFileHeight (CallType call=IMMDevice::wait)
const;
```

videoFileName (IString) — IMMDigitalVideo — vbmm

Description: Returns the file name of the digital video file.

Get:
```
IString videoFileName () const;
```

videoFileWidth (unsigned long) — IMMDigitalVideo — vbmm

Description: Returns the width of the video for the currently loaded video file.

Get:
```
unsigned long videoFileWidth (CallType call=IMMDevice::wait)
const;
```

videoHeight (unsigned long) — IMMDigitalVideo — vbmm

Description: Returns the nomial (reasonable) height of the video for this machine.

Get:
```
unsigned long videoHeight (CallType call=IMMDevice::wait)
const;
```

videoWidth (unsigned long) — IMMDigitalVideo — vbmm

Description: Returns the nomial (reasonable) width of the video for this machine.

Get:
```
unsigned long videoWidth (CallType call=IMMDevice::wait)
const;
```

view (IToolBarButton::View) — IToolBarButton

Description: Returns the current view of the tool bar button.

Get:
```
View view () const;
```

Set:
```
virtual IToolBarButton& setView (View view);
```

viewedPagesWindow (IWindowHandle) — IHelpWindow

Description: Returns the handle of the Viewed Pages window.

Get:
```
IWindowHandle viewedPagesWindow () const;
```

viewNumber (unsigned long) — ITitle

Description: Returns the view number.
Get: `unsigned long viewNumber () const;`
Set: `virtual ITitle& setViewNumber (unsigned long viewNumber);`
ID: `viewNumberId`

viewPortOnWindow (IRectangle) — IContainerControl

Description: Retrieves the current work area in window coordinates.
Get: `IRectangle viewPortOnWindow () const;`

viewPortOnWorkspace (IRectangle) — IContainerControl

Description: Retrieves the current work area in workspace coordinates.
Get: `IRectangle viewPortOnWorkspace () const;`

viewText (IString) — ITitle

Description: Returns the view text.
Get: `virtual IString viewText () const;`
Set: `virtual ITitle& setViewText (const char* viewText);`
ID: `viewTextId`

viewWindow (IWindowHandle) — IViewPort

Description: Returns the handle of the window bounded by the view port.
Get: `virtual IWindowHandle viewWindow ();`

viewWindowDrawRectangle (IRectangle) — IViewPort

Description: Returns the area of the view window that is currently visible in the view port.
Get: `virtual IRectangle viewWindowDrawRectangle () const;`

viewWindowSize (ISize) — IViewPort

Description: Returns the logical size of the window being scrolled.

Get:
```
virtual ISize viewWindowSize () const;
```

Set:
```
virtual IViewPort& setViewWindowSize
(const ISize& viewWindowSize);
```

ID:
```
viewWindowSizeId
```

virtualKey (IKey::VirtualKey) — IAcceleratorKey

Description: Returns the platform-independent virtual key defined for this accelerator key.

Get:
```
VirtualKey virtualKey () const;
```

visible (Boolean) — IContainerColumn

Description: If the column is visible in the container, `true` is returned.

Get:
```
Boolean isVisible () const;
```

Set:
```
virtual IContainerColumn& show (Boolean visible=true);
```

visible (Boolean) — IContainerObject

Description: If this object is visible, `true` is returned.

Get:
```
virtual Boolean isVisible (IContainerControl* container=0)
const;
```

Set:
```
virtual IContainerObject& show (Boolean visible=true,
IContainerControl* container=0);
```

visible (Boolean) — IHelpWindow

Description: If the window's style is set to visible, `true` is returned.

Get:
```
Boolean isVisible () const;
```

ID:
```
visibleId
```

visible (Boolean) — IWindow

Description: If the window's style is set to visible, `true` is returned.

Get:
```
Boolean isVisible () const;
```

Set:
```
virtual IWindow& show (Boolean visible=true);
```

ID:
```
visibleId
```

visibleCount (unsigned long) — IScrollBar

Description: Returns the amount of the scrollable range that is displayed.

Get:
```
virtual unsigned long visibleCount () const;
```

Set:
```
virtual IScrollBar& setVisibleCount
(unsigned long visibleCount);
```

visibleLines (unsigned long) — IMultiLineEdit

Description: Returns the number of lines that completely fit inside the edit region of the MLE.

Get:
```
unsigned long visibleLines () const;
```

visibleRectangle (IRectangle) — IWindow

Description: For controls like the drop-down combination box where the painted rectangle of the control is different than its actual rectangle, this function returns the painted rectangle.

Get:
```
virtual IRectangle visibleRectangle () const;
```

volume (unsigned long) — IMMDevice — vbmm

Description: Returns the volume setting for the passed in audio channel.

Get:
```
virtual unsigned long volume
(AudioChannel channel=IMMDevice::left,
CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMDevice& setVolume (unsigned long volume,
AudioChannel channel=IMMDevice::all,
const IMMMillisecondTime& over=IMMMillisecondTime ( ),
CallType call=IMMDevice::wait);
```

volume (unsigned long) — IMMMasterAudio — vbmm

Description: Returns the master volume setting for either the current setting or the saved setting to a percent of the maximum audio level.

Get:
```
unsigned long volume
(SettingSource source=IMMMasterAudio::current,
IMMDevice::CallType call=IMMDevice::wait) const;
```

Set:
```
virtual IMMMasterAudio& setVolume (unsigned long volume,
IMMDevice::CallType call=IMMDevice::wait);
```

whiteSpace (Boolean) — IString

Description: If all the characters are in {0x09-0x0D,0x20}, `true` is returned.

Get:
```
Boolean isWhiteSpace () const;
```

width (IRectangle::Coord) — IRectangle

Description: Returns the width of the rectangle.

Get:
```
Coord width () const;
```

width (Coord) — ISize

Description: Returns the width represented by the `ISize` object.

Get:
```
Coord width () const;
```

Set:
```
ISize& setWidth (Coord width);
```

willDeleteColumnsOnClose (Boolean) — IContainerControl

Description: If the columns will be deleted when the container is deleted, `true` is returned.

Get:
```
Boolean willDeleteColumnsOnClose () const;
```

willDeleteObjectsOnClose (Boolean) — IContainerControl

Description: If container objects will be deleted when the container is deleted, `true` is returned.

Get:
```
Boolean willDeleteObjectsOnClose () const;
```

willDestroyOnClose (Boolean) — IFrameWindow

Description: Returns the state of the destroy window flag.

Get:
```
Boolean willDestroyOnClose () const;
```

wordWrap (Boolean) — IMultiLineEdit

Description: Queries whether the MLE is in word-wrap mode.

Get:
```
Boolean isWordWrap () const;
```

Set:
```
virtual IMultiLineEdit& enableWordWrap
(Boolean wordWrap=true);
```

workPhone (IString) — ICustomer — vbsample

Description: Query the workPhone (IString) attribute.

Get:
```
virtual IString workPhone () const;
```

Set:
```
virtual ICustomer& setWorkPhone (const IString& workPhone);
```

ID:
```
workPhoneId
```

writeable (Boolean) — IBaseSpinButton

Description: If the user cannot type in the spin field, true is returned.

Get:
```
Boolean isWriteable () const;
```

Set:
```
virtual IBaseSpinButton& enableDataUpdate
(Boolean writeable=true);
```

writeable (Boolean) — IContainerColumn

Description: If the user can edit the data in the column, true is returned.

Get:
```
Boolean isWriteable () const;
```

Set:
```
virtual IContainerColumn& enableDataUpdate
(Boolean writeable=true);
```

writeable (Boolean) — IContainerObject

Description: Returns true if the data in the object can be edited.

Get:
```
virtual Boolean isWriteable (IContainerControl* container=0)
const;
```

Set:
```
virtual IContainerObject& enableDataUpdate
(Boolean writeable=true, IContainerControl* container=0);
```

writeable (Boolean) — IEntryField

Description: Queries whether the contents of the entry field can be modified.

Get:
```
Boolean isWriteable () const;
```

Set:
```
virtual IEntryField& enableDataUpdate
(Boolean writeable=true);
```

ID: `dataUpdateId`

writeable (Boolean) — IMultiLineEdit

Description: Queries whether the user can modify the contents of the MLE.

Get:
```
Boolean isWriteable () const;
```

Set:
```
virtual IMultiLineEdit& enableDataUpdate
(Boolean writeable=true);
```

ID: `dataUpdateId`

writeLineNumberEnabled (Boolean) — ITrace

Description: Determines whether line numbers are currently being written.

Get:
```
static Boolean isWriteLineNumberEnabled ();
```

writePrefixEnabled (Boolean) — ITrace

Description: Determines whether the line count prefix is being written.

Get:
```
static Boolean isWritePrefixEnabled ();
```

x (Coord) — IPoint

Description: Returns the point's X-coordinate.

Get:
```
Coord x () const;
```

Set:
```
IPoint& setX (Coord x);
```

y (Coord) — IPoint

Description: Returns the point's Y-coordinate.

Get:
```
Coord y () const;
```

Set:
```
IPoint& setY (Coord y);
```

year (int) — IDate

Description: Returns the receiver's year.

Get:
```
int year () const;
```

zip

zip (IString) — IAddress — vbsample

Description: Query the zip (`IString`) attribute.

Get:
```
virtual IString zip () const;
```

Set:
```
virtual IAddress& setZip (const IString& zip);
```

ID:
```
zipId
```

Chapter 166. Events

activateEvent — IFrameWindow

Description: Notification identifier provided to observers when the frame window is activated.

ID: `activateId`

addedEvent — IRBag — vbcc

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

addedEvent — IRDeque — vbcc

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

addedEvent — IREqualitySequence — vbcc

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

addedEvent — IRHeap — vbcc

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

addedEvent — IRQueue — vbcc

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

addedEvent — IRSequence

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

addedEvent — IRSet — vbcc

Description: Notification provided when element is added.

ID: `IPartCollectionNotification::addedId`

© Copyright IBM Corp. 1995, 1996

addedEvent — IRSortedBag — vbcc

Description: Notification provided when element is added.
ID: `IPartCollectionNotification::addedId`

addedEvent — IRSortedSet — vbcc

Description: Notification provided when element is added.
ID: `IPartCollectionNotification::addedId`

addedEvent — IRStack — vbcc

Description: Notification provided when element is added.
ID: `IPartCollectionNotification::addedId`

addEvent — IContainerControl

Description: Notification identifier provided to observers when objects are added to a container.
ID: `addId`

anyEvent — INotifier

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRBag — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRDeque — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IREqualitySequence — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRHeap — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRQueue — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRSequence

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRSet — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRSortedBag — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRSortedSet — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

anyEvent — IRStack — vbcc

Description: Notification for any event.
ID: `VBANYEVENT`

armTrackEvent — ISlider

Description: Notification identifier provided to observers when tracking.
ID: `armTrackId`

buttonClickEvent — IButton

Description: Notification identifier provided to observers when the button control is clicked by the user.
ID: `buttonClickId`

characterTypeEvent — IEntryField

Description: Notification identifier provided to observers when the character type of the entry field control changes.
ID: `characterTypeId`

closeEvent — IFrameWindow

Description: Notification identifier provided to observers when the frame window is closed.
ID: `closeId`

commandEvent — IMenuItem

Description: Notification identifier provided to observers when a WM_COMMAND event occurs.

ID: `commandId`

commandEvent — IWindow

Description: Notification identifier provided to observers when an `IWindow::command` event occurs.

ID: `commandId`

commandNotifyEvent — IMMDevice — vbmm

Description: Provides a notification identifier to observers when one of the `IMMDevice` functions has been called with the notify flag.

ID: `commandNotifyId`

createdEvent — IVBFileDialog

Description: Notification identifier provided to observers when a new `IFileDialog` has been created.

ID: `createdId`

createdEvent — IVBFontDialog

Description: Notification identifier provided to observers when a new `IFontDialog` has been created.

ID: `createdId`

cuePointEvent — IMMDevice — vbmm

Description: Provides a notification identifier to observers when the device receives a cue point message.

ID: `cuePointId`

customerAddedEvent — ICompany — vbsample

Description: Notification event for when customer added to customer list.

ID: `customerAddedId`

deactivateEvent — IFrameWindow

Description: Notification identifier provided to observers when the frame window is deactivated.

ID: `deactivateId`

detailsViewTitlesEvent — IContainerControl

Description: Notification identifier provided to observers when the visibility of the container details view titles changes.

ID: `detailsViewTitlesId`

deviceEvent — IMMDevice — vbmm

Description: Provides a notification identifier to observers when the device receives a device message.

ID: `deviceEventId`

digitsEvent — IVBDataTypePart — vbsample

Description: Notification event for digits attribute.

ID: `digitsId`

dismissedEvent — IVBFileDialog

Description: Notification identifier provided to observers when the showing `IFileDialog` has been dismissed.

ID: `dismissedId`

dismissedEvent — IVBFontDialog

Description: Notification identifier provided to observers when the showing `IFontDialog` has been dismissed.

ID: `dismissedId`

enterEvent — IBaseComboBox

Description: Notification identifier provided to observers when an item in the list box portion of a combination box is `double` clicked or enter is pressed on an item with the cursor.

ID: `enterId`

enterEvent — IBaseListBox

Description: Notification identifier provided to observers when the user `double` clicks on an item or presses enter on an item with the cursor in a list box.

ID: `enterId`

enterEvent — IContainerControl

Description: Notification identifier provided to observers when an item in the container is `double` clicked, or the user presses the enter key.

ID: `enterId`

extendedSelectChangedEvent — ICollectionViewListBox

Description: Notification identifier provided to observers when the extended select state of the list box changes.
ID: `extendedSelectChangedId`

gotFocusEvent — IWindow

Description: Notification provided when focus attribute is `true`.
ID: `FOCUSTRUEID`
Parameter: focus (Boolean)

inputDisabledEvent — IWindow

Description: Notification provided when enable attribute is `false`.
ID: `ENABLEFALSEID`
Parameter: enable (Boolean)

inputEnabledEvent — IWindow

Description: Notification provided when enable attribute is `true`.
ID: `ENABLETRUEID`
Parameter: enable (Boolean)

inputStringIsValid — IVBDataTypePart — vbsample

Description: Notification event when input string is valid.
ID: `inputStringIsValidId`

inputStringNotValid — IVBDataTypePart — vbsample

Description: Notification event when input string not valid.
ID: `inputStringNotValidId`

itemChangedEvent — ICollectionViewComboBox

Description: Notification identifier provided to observers when a combination box item changes due to a change in the underlying collection element.
ID: `itemChangedId`

itemChangedEvent — ICollectionViewListBox

Description: Notification identifier provided to observers when a list box item changes due to a change in the underlying collection element.
ID: `itemChangedId`

itemChangedEvent — IVBContainerControl

Description: Notification identifier provided to observers when a container item changes due to a change in the underlying collection element.
ID: `itemChangedId`

lostFocusEvent — IWindow

Description: Notification provided when focus attribute is `false`.
ID: `FOCUSFALSEID`
Parameter: focus (Boolean)

lowerCaseEvent — IVBDataTypePart — vbsample

Description: Notification event when text is lower case.
ID: `lowerCaseId`

mediaLoadedEvent — IMMRemovableMedia — vbmm

Description: Notification identifier provided to observers when the current media-loaded state changes.
ID: `mediaLoadedId`

modifiedEvent — IRBag — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRDeque — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IREqualitySequence — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRHeap — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRQueue — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRSequence

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRSet — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRSortedBag — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRSortedSet — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

modifiedEvent — IRStack — vbcc

Description: Notification provided when collection is modified.
ID: `IPartCollectionNotification::modifiedId`

passDeviceEvent — IMMDevice — vbmm

Description: Provides a notification identifier to observers when the device receives a pass device message.
ID: `passDeviceId`

positionChangeEvent — IMMDevice — vbmm

Description: Provides a notification identifier to observers when the device receives a position change message.
ID: `positionChangeId`
Parameter: position (IMMTime*)

positionTimerEvent — IMMAudioCD — vbmm

Description: Notification identifier provided to observers as the position changes.
ID: `positionTimerId`
Parameter: position (IMMTrackMinSecFrameTime*)

pressedOkEvent — IVBFileDialog

Description: Notification identifier provided to observers when the showing `IFileDialog` has been dismissed with ok button.
ID: `pressedOkId`

pressedOkEvent — IVBFontDialog

Description: Notification identifier provided to observers when the showing `IFontDialog` has been dismissed with ok button.
ID: `pressedOkId`

removedEvent — IRBag — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRDeque — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IREqualitySequence — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRHeap — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRQueue — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRSequence

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRSet — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRSortedBag — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRSortedSet — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removedEvent — IRStack — vbcc

Description: Notification provided when element is removed.
ID: `IPartCollectionNotification::removedId`

removeEvent — IContainerControl

Description: Notification identifier provided to observers when objects are removed from a container.
ID: `removeId`

replacedEvent — IRBag — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRDeque — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IREqualitySequence — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRHeap — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRQueue — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRSequence

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRSet — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRSortedBag — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRSortedSet — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

replacedEvent — IRStack — vbcc

Description: Notification provided when element is replaced.
ID: `IPartCollectionNotification::replacedId`

scaleEvent — IProgressIndicator

Description: Notification identifier provided to observers when the scale style of a `IProgressIndicator` window changes.
ID: `scaleId`

selectEvent — IContainerControl

Description: Notification identifier provided to observers when a container item's selection state changes.
ID: `selectId`

systemCommandEvent — IWindow

Description: Notification identifier provided to observers when an `IWindow::systemCommand` event occurs.
ID: `systemCommandId`

textEqualDefaultEvent — IVBDataTypePart — vbsample

Description: Notification event when text equals default attribute.
ID: `textEqualDefaultId`

textLengthEvent — IVBDataTypePart — vbsample

Description: Notification event for textLength attribute.
ID: `textLengthId`

textNotEqualDefaultEvent — IVBDataTypePart — vbsample

Description: Notification event when text does not equal default attribute.
ID: `textNotEqualDefaultId`

trackStartedEvent — IMMAudioCD — vbmm

Description: Notification identifier provided to observers when the a new track is started.
ID: `trackStartedId`

upperCaseEvent — IVBDataTypePart — vbsample

Description: Notification event when text is upper case.
ID: `upperCaseId`

valueAboveHighLimitEvent — IVBDataTypePart — vbsample

Description: Notification event when value is above high limit attribute.
ID: `valueAboveHighLimitId`

valueBelowLowLimitEvent — IVBDataTypePart — vbsample

Description: Notification event when value is below low limit attribute.
ID: `valueBelowLowLimitId`

valueEqualDefaultEvent — IVBDataTypePart — vbsample

Description: Notification event for value equals default attribute.
ID: `valueEqualDefaultId`

valueEqualHighLimitEvent — IVBDataTypePart — vbsample

Description: Notification event when value equals high limit attribute.
ID: `valueEqualHighLimitId`

valueEqualLowLimitEvent — IVBDataTypePart — vbsample

Description: Notification event when value equals low limit attribute.
ID: `valueEqualLowLimitId`

valueFalseEvent — IVBBooleanPart — vbsample

Description: Notification event for value is `false`.
ID: `VALUEFALSEID`

valueNegativeEvent — IVBDataTypePart — vbsample

Description: Notification event when value is negative (value < 0).
ID: `valueNegativeId`

valueNotEqualDefaultEvent — IVBDataTypePart — vbsample

Description: Notification event when value does not equal default attribute.
ID: `valueNotEqualDefaultId`

valueNotZeroEvent — IVBDataTypePart — vbsample

Description: Notification event when value is not zero.
ID: `valueNotZeroId`

valueOutsideLimitsEvent — IVBDataTypePart — vbsample

Description: Notification when (value < low limit) or (value > high limit).
ID: `valueOutsideLimitsId`

valuePositiveEvent — IVBDataTypePart — vbsample

Description: Notification event when value is positive (value > 0).
ID: `valuePositiveId`

valueTrueEvent — IVBBooleanPart — vbsample

Description: Notification event for value is `true`.
ID: `VALUETRUEID`

valueWithinLimitsEvent — IVBDataTypePart — vbsample

Description: Notification when (high limit >= value >= low limit).
ID: `valueWithinLimitsId`

valueZeroEvent — IVBDataTypePart — vbsample

Description: Notification event for value is zero.
ID: `valueZeroId`

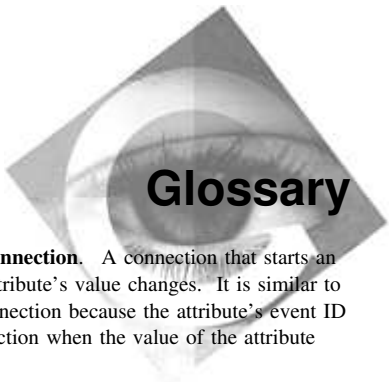
visibilityDisabledEvent — IWindow

Description: Notification provided when visible attribute is `false`.
ID: `VISIBLEFALSEID`
Parameter: visible (Boolean)

visibilityEnabledEvent — IWindow

Description: Notification provided when visible attribute is `true`.
ID: `VISIBLETRUEID`
Parameter: visible (Boolean)

visibilityEnabledEvent



Glossary

This glossary defines terms and abbreviations that are used in this book. If you do not find the term you are looking for, refer to the *IBM Dictionary of Computing*, New York:McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

A

abstract class. A class that provides common behavior across a set of subclasses but is not itself designed to have instances that work. An abstract class represents a concept; classes derived from it represent implementations of the concept. For example, IControl is the abstract base class for control view windows; the ICanvas and IListBox classes are controls derived from IControl. An abstract class must have at least one pure virtual function.

See also *base class*.

access. A property of a class that determines whether a class member is accessible in an expression or declaration.

action. A specification of a function that a part can perform. The visual builder uses action specifications to generate connections between parts. Actions are resolved to member function calls in the generated code.

Compare to *event* and *attribute*.

argument. A data element, or value, included as part of a member function call. Arguments provide additional information that the called member function can use to perform the requested operation.

attribute. A specification of a property of a part. For example, a customer part could have a name attribute and an address attribute. An attribute can itself be a part with its own behavior and attributes.

The visual builder uses attribute specifications to generate code to get and set part properties.

Compare to *event* and *action*.

attribute-to-action connection. A connection that starts an action whenever an attribute's value changes. It is similar to an event-to-action connection because the attribute's event ID is used to notify the action when the value of the attribute changes.

See also *connection*. Compare to *event-to-action connection*.

attribute-to-attribute connection. A connection from an attribute of one part to an attribute of another part. When one attribute is updated, the other attribute is updated automatically.

See also *connection*.

attribute-to-member function connection. A connection from an attribute of a part to a member function. The connected attribute receives its value from the member function, which can make calculations based on the values of other parts.

See also *connection*.

B

base class. A class from which other classes or parts are derived. A base class may itself be derived from another base class.

See also *abstract class*.

behavior. The set of external characteristics that an object exhibits.

C

caller. An object that sends a member function call to another object.

Contrast with *receiver*.

category. In the Composition Editor, a selectable grouping of parts represented by an icon in the left-most column. Selecting a category displays the parts belonging to that category in the next column.

See also *parts palette*.

class. An aggregate that can contain functions, types, and user-defined operators, in addition to data. Classes can be defined hierarchically, allowing one class to be an expansion of another, and can restrict access to its members.

© Copyright IBM Corp. 1995, 1996

Class Editor. The editor you use to specify the names of files that Visual Builder writes to when you generate default code. You can also use this editor to do the following:

- Enter a description of the part
- Specify a different .vbb file in which to store the part
- See the name of the part's base class
- Modify the part's default constructor
- Enter additional constructor and destructor code
- Specify a .lib file for the part
- Specify a resource DLL and ID to assign an icon to the part
- Specify other files that you want to include when you build your application

Compare to *Composition Editor* and *Part Interface Editor*.

class hierarchy. A tree-like structure showing relationships among object classes. It places one abstract class at the top (a base class) and one or more layers of less abstract classes below it.

class library. A collection of classes.

class member function. See *member function*.

client area object. An intermediate window between a frame window (IFrameWindow) and its controls and other child windows.

client object. An object that requests services from other objects.

collection. A set of features in which each feature is an object.

Common User Access (CUA). An IBM architecture for designing graphical user interfaces using a set of standard components and terminology.

composite part. A part that is composed of a part and one or more subparts. A composite part can contain visual parts, nonvisual parts, or both.

See also *nonvisual part, part, subpart,* and *visual part*.

Composition Editor. A view that is used to build a graphical user interface and to make connections between parts.

Compare to *Class Editor* and *Part Interface Editor*.

concrete class. A subclass of an abstract class that is a specialization of the abstract class.

connection. A formal, explicit relationship between parts. Making connections is the basic technique for building any visual application because that defines the way in which parts communicate with one another. The visual builder generates the code that then implements these connections.

See also *attribute-to-action connection, attribute-to-attribute connection, attribute-to-member function connection, parameter connection, custom logic connection, event-to-action connection, event-to-attribute connection,* and *event-to-member function connection*.

const. An attribute of a data object that declares that the object cannot be changed.

construction from parts. A software development technology in which applications are assembled from existing and reusable software components, known as parts.

constructor. A special class member function that has the same name as the class and is used to construct and possibly initialize class objects.

CUA. See *Common User Access*.

cursored emphasis. When the selection cursor is on a choice, that choice has cursored emphasis.

custom logic connection. A connection that causes your customized C or C++ code to be run. This connection can be triggered either when an attribute's value changes or an event occurs.

D

data abstraction. A data type with a private representation and a public set of operations. The C++ language uses the concept of classes to implement data abstraction.

data member. Private data that belongs to a given object and is hidden from direct access by all other objects. Data members can only be accessed by the member functions of the defining class and its subclasses.

data model. A combination of the base classes and parts shipped with the product and the classes and parts you save and create. They are saved in a file named vbbase.vbb.

data object. A storage area used to hold a value.

declaration. A description that makes an external object or function available to a function or a block.

DEF file. See *module definition file*.

derivation. The creation of a new or abstract class from an existing or base class.

destructor. A special class member function that has the same name as the class and is used to destruct class objects.

DLL. See *dynamic link library*.

dynamic link library (DLL). In OS/2, a library containing data and code objects that can be used by programs or applications during loading or at run time. Although they are not part of the program's executable (.exe) file, they are sometimes required for an .exe file to run properly.

E

encapsulation. The hiding of a software object's internal representation. The object provides an interface that queries and manipulates the data without exposing its underlying structure.

event. A specification of a notification from a part.

Compare to *action, attribute,* and *part event*.

event-to-action connection. A connection that causes an action to be performed when an event occurs.

See also *connection*.

event-to-attribute connection. A connection that changes the value of an attribute when a certain event occurs.

See also *connection*.

event-to-member function connection. A connection from an event of a part to a member function. When the connected event occurs, the member function is executed.

See also *connection*.

expansion area. The section of a multicell canvas between the current cell grid and the outer edge of the canvas. Visually, this area is bounded by the rightmost column gridline and the bottommost row gridline.

F

feature. (1) A major component of a software product that can be installed separately. (2) In Visual Builder, an action, attribute, or event that is available from a part's part interface and that other parts can connect to.

full attribute. An attribute that has all of the behaviors and characteristics that an attribute can have: a data member, a get member function, a set member function, and an event identifier.

free-form surface. The large open area of the Composition Editor window. The free-form surface holds the visual parts contained in the views you build and representations of the nonvisual parts (models) that your application includes.

G

graphical user interface (GUI). A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands. Typically, a graphical user interface includes a combination of graphics, pointing devices, menu bars and other menus, overlapping windows, and icons.

GUI. See *graphical user interface*.

H

handles. Small squares that appear on the corners of a selected visual part in the visual builder. Handles are used to resize parts.

Compare to *primary selection*.

header file. A file that contains system-defined control information that precedes user data.

I

inheritance. (1) A mechanism by which an object class can use the attributes, relationships, and member functions defined in more abstract classes related to it (its base classes). (2) An object-oriented programming technique that allows you to use existing classes as bases for creating other classes.

instance. Synonym for *object*, a particular instantiation of a data type.

L

legacy code. Existing code that a user might have. Legacy applications often have character-based, nongraphical user interfaces; usually they are written in a nonobject-oriented language, such as C or COBOL.

loaded. The state of the mouse pointer between the time you select a part from the parts palette and deposit the part on the free-form surface.

M

main part. The part that users see when they start an application. This is the part from which the main() function C++ code for the application is generated.

The main part is a special kind of composite part.

See also *part* and *subpart*.

member. (1) A data object in a structure or a union. (2) In C++, classes and structures can also contain functions and types as members.

member function. An operator or function that is declared as a member of a class. A member function has access to the private and protected data members and member functions of objects of its class.

member function call. A communication from one object to another that requests the receiving object to execute a member function.

A member function call consists of a member function name that indicates the requested member function and the arguments to be used in executing the member function. The member function call always returns some object to the requesting object as the result of performing the member function.

Synonym for *message*.

member function name. The component of a member function call that specifies the requested operation.

message. A request from one object that the receiving object implement a member function. Because data is encapsulated and not directly accessible, a message is the only way to send data from one object to another. Each message specifies the name of the receiving object, the member function to be implemented, and any arguments the member function needs for implementation.

Synonym for *member function call*.

model. A nonvisual part that represents the state and behavior of a object, such as a customer or an account.

Contrast with *view*.

module definition file. A file that describes the code segments within a load module.

Synonym for *DEF file*.

N

nested class. A class defined within the scope of another class.

nonvisual part. A part that has no visual representation at run time. A nonvisual part typically represents some real-world object that exists in the business environment.

Compare to *model*. Contrast with *view* and *visual part*.

no-event attribute. An attribute that does not have an event identifier.

no-set attribute. An attribute that does not have a set member function.

notebook part. A visual part that resembles a bound notebook containing pages separated into sections by tabbed divider pages. A user can turn the pages of a notebook or select the tabs to move from one section to another.

O

object. (1) A computer representation of something that a user can work with to perform a task. An object can appear as text or an icon. (2) A collection of data and member functions that operate on that data, which together represent a logical entity in the system. In object-oriented programming, objects are grouped into classes that share common data definitions and member functions. Each object in the class is said to be an instance of the class. (3) An instance of an object class consisting of attributes, a data structure, and operational member functions. It can represent a person, place, thing, event, or concept. Each instance has the same properties, attributes, and member functions as other instances of the object class, though it has unique values assigned to its attributes.

object class. A template for defining the attributes and member functions of an object. An object class can contain other object classes. An individual representation of an object class is called an object.

object factory. A nonvisual part capable of dynamically creating new instances of a specified part. For example, during the execution of an application, an object factory can create instances of a new class to collect the data being generated.

object-oriented programming. A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that comprise

the problem and how they are manipulated, not on how something is accomplished.

observer. An object that receives notification from a notifier object.

operation. A member function or service that can be requested of an object.

overloading. An object-oriented programming technique that allows you to redefine functions and most standard C++ operators when the functions and operators are used with class types.

P

palette. See *parts palette*.

parameter connection. A connection that satisfies a parameter of an action or member function by supplying either an attribute's value or the return value of an action, member function, or custom logic. The parameter is always the source of the connection.

See also *connection*.

parent class. The class from which another part or class inherits data, member functions, or both.

part. A self-contained software object with a standardized public interface, consisting of a set of external features that allow the part to interact with other parts. A part is implemented as a class that supports the INotifier protocol and has a part interface defined.

The parts on the palette can be used as templates to create instances or objects.

part event. A representation of a change that occurs to a part. The events on a part's interface enable other interested parts to receive notification when something about the part changes. For example, a push button generates an event signaling that it has been clicked, which might cause another part to display a window.

part event ID. The name of a part static-data member used to identify which notification is being signaled.

part interface. A set of external features that allows a part to interact with other parts. A part's interface is made up of three characteristics: attributes, actions, and events.

Part Interface Editor. An editor that the application developer uses to create and modify attributes, actions, and events, which together make up the interface of a part.

Compare to *Class Editor* and *Composition Editor*.

parts palette. The parts palette holds a collection of visual and nonvisual parts used in building additional parts for an application. The parts palette is organized into *categories*. Application developers can add parts to the palette for use in defining applications or other parts.

preferred features. A subset of the part's features that appear in a pop-up connection menu. Generally, they are the features used most often.

primary selection. In the Composition Editor, the part used as a base for an action that affects several parts. For example, an alignment tool will align all selected parts with the primary selection. Primary selection is indicated by closed (solid) selection handles, while the other selected parts have open selection handles.

See also *selection handles*.

private. Pertaining to a class member that is accessible only to member functions and friends of that class.

process. A program running under OS/2, along with the resources associated with it (memory, threads, file system resources, and so on).

program. (1) One or more files containing a set of instructions conforming to a particular programming language syntax. (2) A self-contained, executable module. Multiple copies of the same program can be run in different processes.

protected. Pertaining to a class member that is only accessible to member functions and friends of that class, or to member functions and friends of classes derived from that class.

prototype. A function declaration or definition that includes both the return type of the function and the types of its arguments.

primitive part. A basic building block of other parts. A primitive part can be relatively complex in terms of the function it provides.

process. A collection of code, data, and other system resources, including at least one thread of execution, that performs a data processing task.

property. A unique characteristic of a part.

pure virtual function. A virtual function that has a function definition of = 0;.

R

receiver. The object that receives a member function call.

Contrast with *caller*.

resource file. A file that contains data used by an application, such as text strings and icons.

S

selection handles. In the Composition Editor, small squares that appear on the corners of a selected visual part. Selection handles are used to resize parts.

See also *primary selection*.

server. A computer that provides services to multiple users or workstations in a network; for example, a file server, a print server, or a mail server.

service. A specific behavior that an object is responsible for exhibiting.

settings view. A view of a part that provides a way to display and set the attributes and options associated with the part.

sticky. In the Composition Editor, the mode that enables you to add multiple parts of the same class (for example, three push buttons) without going back and forth between the parts palette and the free-form surface.

structure. A construct that contains an ordered group of data objects. Unlike an array, the data objects within a structure can have varied data types.

subpart. A part that is used to create another part.

See also *nonvisual part, part,* and *visual part*.

superclass. See *abstract class* and *base class*.

T

tear-off attribute. An attribute that an application developer has exposed to work with as though it were a stand-alone part.

template. A family of classes or functions with variable types.

thread. A unit of execution within a process.

tool bar. The strip of icons along the top of the free-form surface. The tool bar contains tools to help you construct composite parts.

U

UI. See *user interface*.

unloaded. The state of the mouse pointer before you select a part from the parts palette and after you deposit a part on the free-form surface. In addition, you can unload the mouse pointer by pressing the Esc key.

user interface (UI). (1) The hardware, software, or both that enable a user to interact with a computer. (2) The term *user interface* normally refers to the visual presentation and its underlying software with which a user interacts.

V

variable. (1) A storage place within an object for a data feature. The data feature is an object, such as number or date, stored as an attribute of the containing object. (2) A part that receives an identity at run time. A variable by itself contains no data or program logic; it must be connected such that it receives runtime identity from a part elsewhere in the application.

view. (1) A visual part, such as a window, push button, or entry field. (2) A visual representation that can display and change the underlying model objects of an application. Views are both the end result of developing an application and the basic unit of composition of user interfaces.

Compare to *visual part*. Contrast with *model*.

virtual function. A function of a class that is declared with the keyword virtual. The implementation that is executed when you make a call to a virtual function depends on the type of the object for which it is called. This is determined at run time.

visual part. A part that has a visual representation at run time. Visual parts, such as windows, push buttons, and entry fields, make up the user interface of an application.

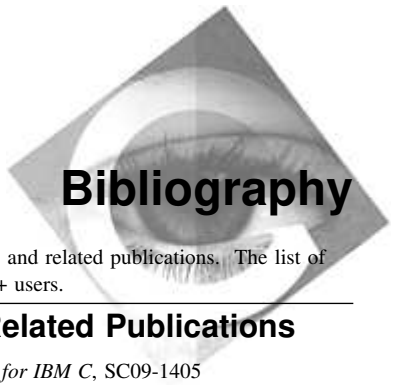
Compare to *view*. Contrast with *nonvisual part*.

visual programming tool. A tool that provides a means for specifying programs graphically. Application programmers write applications by manipulating graphical representations of components.

W

white space. Space characters, tab characters, form-feed characters, and new-line characters.

window. (1) A rectangular area of the screen with visible boundaries in which information is displayed. Windows can overlap on the screen, giving it the appearance of one window being on top of another. (2) In the Composition Editor, a window is a part that can be used as a container for other visual parts, such as push buttons.



Bibliography

This bibliography lists the publications that make up the IBM VisualAge for C++ library and related publications. The list of related publications is not exhaustive but should be adequate for most VisualAge for C++ users.

The IBM VisualAge for C++ Library

The following books are part of the IBM VisualAge for C++ library.

- Installation Guide & Product Overview, S33H-5030
- User's Guide, S33H-5031
- Programming Guide, S33H-5032
- Visual Builder User's Guide, S33H-5034
- Visual Builder Parts Reference, S33H-5035
- Building VisualAge for C++ Parts for Fun and Profit, S33H-5036
- Open Class Library User's Guide, S33H-5033
- Open Class Library Reference, S33H-5039
- Language Reference, S33H-5037-00
- C Library Reference, S33H-5038
- SOM Programming Guide, S33H-5044
- SOM Programming Reference, &dnsomrf.

C and C++ Related Publications

- *Portability Guide for IBM C*, SC09-1405
- *American National Standard for Information Systems / International Standards Organization — Programming Language C (ANSI/ISO 9899-1990[1992])*

Non-IBM Publications

Many books have been written about the C++ language and related programming topics. The authors use varying approaches and emphasis. The following is a sample of some non-IBM C++ publications that are generally available. This sample is not an exhaustive list. IBM does not specifically recommend any of these books, and other C++ books may be available in your locality.

- *The Annotated C++ Reference Manual* by Margaret A. Ellis and Bjarne Stroustrup, Addison-Wesley Publishing Company.
- *C++ Primer* by Stanley B. Lippman, Addison-Wesley Publishing Company.
- *Object-Oriented Design with Applications* by Grady Booch, Benjamin/Cummings.
- *Object-Oriented Programming Using SOM and DSOM* by Christina Lau, Van Nostrand Reinhold.

© Copyright IBM Corp. 1995, 1996

Index

Special Characters

__eof 810
__parmdwords 810
_alloca 802
_atold 803
_cabs 803
_chsize 803
_clear87 803
_control87 803
_crotl 803
_debug_calloc 803
_debug_free 803
_debug_heapmin 804
_debug_malloc 804
_debug_realloc 804
_debug_tcalloc 804
_debug_tfree 804
_debug_theapmin 804
_debug_tmalloc 804
_debug_trealloc 804
_dump_allocated_delta 804
_ecvt 805
_endthread 805
_exit 805
_fcloseall 805
_fcvt 805
_fgetchar 805
_filelength 805
_flushall 805
_fpreset 806
_fputchar 806
_freemod 806
_fullpath 806
_gcvt 806
_heap_check 806
_heapmin 806
_itoa 807
_j0 807
_loadmod 807
_lrotl 807
_ltoa 807
_makepath 807
_matherr 807
_msize 807
_onexit 808
_rmtmp 808
_rotl 808
_searchenv 808
_set_crt_msg_handle 808
_setmode 808
_sopen 808
_splitpath 809
_status87 809
_tcalloc 809
_tdump_allocated_delta 809
_tell 809
_tfree 809
_threadstore 809
_tmalloc 809
_trealloc 810
_ultoa 810

A

abort 653
abs 653
access 653
acos 653
acquire 653
acquired 811
actionType 811
activateEvent 959
activeColor 811
activeTextBackgroundColor 811
activeTextForegroundColor 812
add 653, 654, 655
addAllFrom 655
addAscending 656
addAsFirst 656, 657
addAsLast 657, 658
addAsNext 658
addAsPrevious 658
addAtOffset 658
addAtPosition 658, 659
addBorder 659
addColumn 659
addCustomer 659
addDescending 659
addDetent 659

© Copyright IBM Corp. 1995, 1996

addDifference 659, 660
addDrive 660
addedEvent 959, 960
addEntryAsFirst 660
addEvent 960
addExtension 660
addFileType 660
addFirstPage 661
addIntersection 661
addKey 661
addLastPage 661
addLibraries 662
addLine 662
addLineAsLast 662
addOrReplaceElementWithKey 662
addOrReplaceKey 662
addPageAfter 662
addPageBefore 663
address 812
addToCell 663
addToWindowList 663
addUnion 663
addValue 664
aliasName 812
alignment 812, 813
allElementsDo 664, 665, 666
allowsDragDelete 813
allowsDragDrop 813
allowsMouseClickFocus 813
alphabetic 813
alphanumeric 813
andValue 666
animatedWhenLatched 814
animationRate 814
animationStarted 814
anyElement 814, 815
anyEvent 960, 961
appendText 666
applyBidiSettings 666
area 815
areDetailsViewTitlesVisible 815
areHeadphonesEnabled 815
areSpeakersEnabled 816
armPixelOffset 816
armRange 816
armSize 816
armTickOffset 817
armTrackEvent 961
arrangeIconView 666
asACCEL 817
asDebugInfo 817
asDouble 817
asGUIStyle 667
asin 667
asInt 817
asMMTime 817
asPOINTL 817
asRECTL 818
asRGBLong 818
asSeconds 818
assignTextToCmdLineParm0 667
assignTextToCmdLineParm1 667
assignTextToDateToday 667
assignTextToDefault 667
assignTextToEmpty 667
assignTextToTimeNow 667
assignValueToDefault 667, 668
assignValueToE 668
assignValueToFalse 668
assignValueToHighLimit 668, 669
assignValueToLowLimit 669
assignValueToOne 669, 670
assignValueToPI 670
assignValueToRandom 670
assignValueToTrue 670
assignValueToZero 670, 671
asSIZEL 818
asString 818
asUnsigned 818
atan 671
atof 671
atoi 671
atol 671
audioEnabled 819
autoDeleteObject 819
autoDestroyWindow 819
autoLatchEnabled 819
autoPlayEnabled 819
autoScroll 820
autoSelect 820
autoTab 820
available 820, 821
avgCharWidth 821

B

b2c 671
b2d 672

b2x 672
backgroundColor 821
balance 821
bass 821
beginEdit 672
beginFlashing 672
beginUsingFont 672
bidiSupported 821
binaryDigits 822
binding 822
bitmap 822, 823
bitmapCount 823
bitmapOnly 823
bitmapSize 823
bitmapVisible 823
bitsPerSample 823
blockAlignment 824
blueMix 824
bold 824
border 824
borderColor 824
borderHeight 825
borderSize 825
borderWidth 825
bottom 825
bottomCenter 825
bottomLeft 825
bottomRight 825
bounded 826, 827
button2CnrItems 827
button2CollectionPosition 827
button2CollectionPositions 827
button2Item 827
button2Items 827
button2Point 828
buttonClickEvent 961
buttonPressedId 828
buttonsPosition 828
buttonView 828
bytesPerSecond 828

C

c2b 672
c2d 672
c2x 672
cachingEnabled 829
calloc 672
capturePointer 673
ceil 673
ceilValue 673
center 673, 829
centerAt 673
centeredAt 673
centerXCenterY 829
centerXMaxY 829
centerXMinY 829
change 673, 674
changed 829
changeTextToLowerCase 674
changeTextToUpperCase 674
channels 830
character 830
characterSize 830
characterTypeEvent 961
charType 674, 830
charWidth 674
checked 830
chmod 674
city 830
clear 674
clearDefaultTransparentColor 674
click 675
client 831
clientHandle 831
clientRectFor 675
clipboardHasTextFormat 831
close 675
closeDoor 675
closeEdit 676
closeEvent 961
closeOnDestroy 831
collapse 676
collapseTree 676
columnAt 676
columnCount 831
columnUnderPoint 676
columnWidth 676
command 831
commandEvent 962
commandId 832
commandNotifyEvent 962
commandType 832
communicationWindow 832
compare 676, 677
connectedDeviceId 677
consistent 832, 833
containerFromHandle 678

contains 678
containsAllFrom 678
containsApplication 679
containsKeyLike 679
containsKeyName 679
containsObject 679
contents 833
contentsWindow 833
continuousPlayEnabled 833
control 833
convertToGUIStyle 679
convertToWorkspace 679
coord1 833
coord2 834
copy 680
copyObjectTo 681
copyValueToDefault 681
cos 681
cosh 682
count 834
coverPageWindow 834
creat 682
createdEvent 962
csid 682
cueForPlayback 682
cueForRecording 682
cuePointEvent 962
currentBitmapIndex 834
currentEditColumn 834
currentEditMLE 834
currentEditObject 834
currentGraphicType 835
cursoredCollectionPosition 835
cursoredItem 835
cursoredObject 835
cursorLinePosition 835
cursorPosition 835, 836
cursorSelect 836
customerAddedEvent 962
customerList 836
cut 682

D

d2b 683
d2c 683
d2x 683
data 683
dataAsDate 683
dataAsIcon 683
dataAsNumber 683
dataAsString 683
dataAsTime 683
dataIString 836
date 836, 837
dayName 684
dayOfMonth 837
dayOfWeek 837
dayOfYear 837
daysInMonth 684
daysInYear 684
deactivateEvent 962
deckCount 837
deckOrientation 837
default 838
defaultApplicationName 838
defaultCell 838
defaultGroupPad 838
defaultInput 838
defaultMargin 838
defaultMisfitWidth 839
defaultOrdering 839
defaultPad 839
defaultPushButton 839
defaultText 839, 840
defaultTransparentColor 840
defaultTransparentColorSet 840
defaultValue 840, 841
delayTime 841
deleteElementWithApplication 684
deleteElementWithKey 684
deletePendingEvents 684
deleteSelection 684
dequeue 684
descendentsOf 685
description 841
deselect 685
deselectAll 685, 686
desktopWindow 686
destinationRectangle 842
detailsObjectRectangle 686
detailsTitleRectangle 842
detailsView 842
detailsViewPortOnWindow 842
detailsViewPortOnWorkspace 842
detailsViewSplit 842
detailsViewTitlesEvent 963
detentPosition 686

deviceEvent 963
deviceId 843
deviceName 843
deviceType 843
differenceWith 686
digits 843
digitsEvent 963
disable 687
disableAnimateWhenLatched 687
disableAudio 687
disableAutoLatch 687
disableAutoPlay 687
disableAutoScroll 687
disableAutoSelect 688
disableAutoTab 688
disableCaching 688
disableCommand 688
disableConnector 688
disableContinuousPlay 688
disableCursorSelect 688
disabled 843
disableDataUpdate 689
disabledBackgroundColor 844
disableDefault 689
disabledForegroundColor 844
disableDragDelete 689
disableDragDrop 689
disableDragLines 690
disableDrawBackground 690
disableDrawItem 690
disableDrop 690
disabledText 844
disableExtendedSelect 690
disableFastSpin 690
disableFillBackground 691
disableFlyOverHelp 691
disableGridLines 691
disableGroup 691
disableHalftone 691
disableHeadingUpdate 691
disableHeadphones 691
disableHelp 691
disableInsertMode 691
disableLatching 692
disableMargin 692
disableMisfitFiltering 692
disableMonitoring 692
disableMouseClickFocus 692
disableMultipleSelect 692
disableNoAdjustPosition 692
disableNotification 692, 693
disableRibbonStrip 694
disableSizeToGraphic 694
disableSnapToTick 694
disableSpeakers 694
disableStrikeout 694
disableSystemCommand 694
disableTabStop 694
disableTitleUpdate 694
disableTrace 694
disableUnderscore 695
disableUpdate 695
disableWordWrap 695
disableWriteLineNumber 695
disableWritePrefix 695
discard 695
discId 844
discTitle 844
dismiss 695
dismissedEvent 963
dispatchRemainingHandlers 695
displaySize 844
displayWidth 845
distanceFrom 696
div 696
divideValue 696
dotProduct 696
dragLines 845
drawBackgroundEnabled 845
drawItem 845
drawItemEnabled 845, 846
dropOnAble 846
dup 696
dup2 697

E

editColumnTitle 697
editContainerTitle 697
editObject 697
editRegionHeight 846
editRegionWidth 846
elementAt 697
elementAtPosition 697, 698
elementWithKey 698
empty 698, 846, 847, 848
enable 698, 699
enableConnector 699

enabled 848
enableDataUpdate 699
enabledForNotification 848, 849, 850
enableDrop 699
enableHeadphones 699
enableSpeakers 699
enableTrace 699
enableUpdate 699
enableWriteLineNumber 700
enableWritePrefix 700
endEdit 700
endFlashing 700
endUsingFont 700
enqueue 700
enterEvent 963
erf 700
errorId 850, 851
exit 700
exp 700
expand 701
expandBy 701
expanded 851
expandedBy 701
expandTree 701
exportSelectedTextToFile 701
exportToFile 701
extendedSelect 851
extendedSelectChangedEvent 964
extendedSelection 851
externalLeading 851

F

fabs 701
fastForwardButton 851
fastSpinEnabled 851
fclose 702
fdopen 702
feof 702
ferror 702
fflush 702
fgetpos 702
fgetws 702
fileDialog 851
fileDialogSettings 852
filename 852
fileno 702
fileNormalSpeed 852
fillBackground 853
fillColor 853
filter 703
firstElement 853, 854
firstPage 854
fixed 854
flashing 854
floatingFrame 854
floatingPosition 854
floatingTitle 855
floatSamples 553
floor 703
floorValue 703
flowed 855
flowedNameView 855
flowedTextView 855
flyHelpText 703
flyOverHelp 855
flyOverHelpHandler 855
flyTextControl 855
flyTextStringTableOffset 856
fmod 703
focus 856
font 856
fontDialog 856
fontDialogSettings 856
fopen 703
foregroundColor 857
format 703, 857
formatAsHandle 703
formatCount 857
fputc 704
fputs 704
fputwc 704
fputws 704
framed 857
frameRectFor 704
frames 857, 858
framesPerSecond 858
frameWindow 858
fread 704
free 704
freopen 705
frexp 705
fseek 705
fsetpos 705
ftell 705
full 858, 859
fwrite 705

G

gain 859
gamma 705
getc 705
getenv 706
gets 706
getwc 706
gotFocusEvent 964
goToEntry 706
graphicContext 860
graphicList 860
graphics 860
graphicWindow 860
greenMix 860
gridLines 860
group 860
groupPad 861

H

halftone 861
handle 861
handleCursoredChange 706
handleException 706
handleFor 706
handleInuseChange 707
handleOpen 707
handleSelectedChange 707
handleTreeCollapse 707
handleTreeExpand 707
hasBitmap 861
hasData 707
hasSelectedText 862
hasText 862
hasTransparentColor 862
headingIcon 862
headingIconHandle 862
headingString 862
headingText 863
headingWriteable 863
height 863
help 863
helpId 863, 864
helpWindow 707
hexDigits 864
hide 708
hideDetailsViewTitles 708
hideList 708
hideObject 708
hidePanelIds 708
hideSeparators 708
hideSourceEmphasis 708
hideSplitBar 709
hideTitle 709
hideTitleSeparator 709
hideTreeLine 709
highlight 709
highlighted 864
highLimit 864, 865
hiliteBackgroundColor 865
hiliteForegroundColor 866
homePhone 866
homePosition 866
horizontal 866
horizontalDataAlignment 866
horizontalHeadingAlignment 866
horizontalScroll 866, 867
horizontalScrollBar 867
horizontalSeparator 867
hours 867
hundredths 867
hypot 709

I

I0String 59
I3StateCheckBox 61
IAccelerator 67
IAcceleratorKey 69
IAcceleratorTable 71
IAddress 555
IAnimatedButton 73
IBase
  IAccelerator 67
  IAcceleratorKey 69
  IAcceleratorTable 71
  IDate 143
  IMenuItem 215
  IMMAudioBuffer 499
  IPair 253
  IPointArray 257
  IRectangle 281
  IResourceId 283
  IString 321
  ITime 335
IBaseComboBox
  ICollectionViewComboBox 105
  IComboBox 123

IBaseListBox
ICollectionViewListBox 113
IListBox 199
IBaseSpinButton
INumericSpinButton 243
ITextSpinButton 329
IBitmapControl 79
IIconControl 195
IButton
ICustomButton 137
IPushButton 267
ICanvas 83
IDrawingCanvas 145
IMultiCellCanvas 225
ISetCanvas 293
ISplitCanvas 307
IViewPort 399
ICheckBox 89
ICircularSlider 95
ICLibErrorInfo 101
IClipboard 103
ICollectionViewComboBox 105
ICollectionViewListBox 113
IColor 121
IComboBox 123
ICompany 557
icon 867, 868
iconHandle 868
iconRectangle 709
iconSize 868
IContainerColumn 131
IContainerControl
IVBContainerControl 359
IContainerObject 135
iconText 868
IControl
ICanvas 83
INotebook 237
IOutlineBox 249
IProgressIndicator 261
IScrollBar 287
iconView 868
ICustomButton 137
IAnimatedButton 73
IToolBarButton 349
ICustomer 559
id 868, 869
IDate 143
IDrawingCanvas 145
IDynamicLinkLibrary 151
IElemPointer 411
IEntryField 153
IErrorInfo
ICLibErrorInfo 101
IGUIErrorInfo 187
IMMErrorInfo 515
ISystemErrorInfo 327
IFlyOverHelpHandler 159
IFlyText 161
IVBFlyText 377
IFont 165
IFrameWindow 167
IGraphicPushButton 175
IGroupBox 181
IGUIErrorInfo 187
IHandler
IFlyOverHelpHandler 159
IVBDragDropHandler 369
IHelpWindow 189
IIconControl 195
IListBox 199
IMenu 205
IMenuCascade 211
IMenuHandler
IVBCheckMenuHandler 357
IMenuItem 215
IMenuCascade 211
IMenuSeparator 219
IMenuSeparator 219
IMessageBox 221
IMM24FramesPerSecondTime 489
IMM25FramesPerSecondTime 491
IMM30FramesPerSecondTime 493
IMMAmpMixer 495
IMMAudioBuffer 499
IMMAudioCD 501
IMMAudioCDContents 507
IMMConfigurableAudio
IMMDigitalVideo 509
IMMWaveAudio 543
IMMDevice
IMMAmpMixer 495
IMMDigitalVideo 509
immediateDescendentsOf 709
IMMErrorInfo 515
IMMFileMedia
IMMSequencer 533
IMMHourMinSecFrameTime 517
IMM24FramesPerSecondTime 489

IMMHourMinSecFrameTime *(continued)*
IMM25FramesPerSecondTime 491
IMM30FramesPerSecondTime 493
IMMHourMinSecTime 519
IMMMasterAudio 521
IMMMillisecondTime 525
IMMMinSecFrameTime 527
IMMPlayerPanel 529
IMMRemovableMedia
IMMAudioCD 501
IMMSequencer 533
IMMSpeed 537
IMMTime 539
IMMHourMinSecFrameTime 517
IMMHourMinSecTime 519
IMMMillisecondTime 525
IMMMinSecFrameTime 527
IMMTrackMinSecFrameTime 541
IMMTrackMinSecFrameTime 541
IMMWaveAudio 543
IMngPointer 413
importFromFile 710
IMultiCellCanvas 225
IMMPlayerPanel 529
IMultiLineEdit 231
inactiveColor 869
inactiveText 869
inactiveTextBackgroundColor 869
inactiveTextForegroundColor 870
includes 710
includesDBCS 870
includesMBCS 870
includesSBCS 870
index 870
indexOf 710
indexOfAnyBut 710
indexOfAnyOf 710
indexOfPhrase 710
indexOfWord 711
indexWindow 870
initialize 870
INotebook 237
INotifier
IStandardNotifier 313
input1 871
input10 871
input2 871, 872
input3 872
input4 872
input5 873
input6 873
input7 873, 874
input8 874
input9 874
inputDisabledEvent 964
inputEnabledEvent 964
inputStringIsValid 964
inputStringNotValid 964
insert 711
insertMode 875
integerWithKey 711
internalLeading 875
intersectionWith 711, 712
intersects 712
INumericSpinButton 243
inUse 875
IOrderedRecord 561
ioSamples 563
IOutlineBox 249
IPair 253
IPoint 255
IRange 279
ISize 299
IPoint 255
IPointArray 257
IProfile 259
IProgressIndicator 261
ISlider 301
IPushButton 267
IGraphicPushButton 175
IRadioButton 273
IRange 279
IRecord 565
IOrderedRecord 561
IRectangle 281
IResourceId 283
IResourceLibrary 285
IDynamicLinkLibrary 151
isAbbreviationFor 712
isAnExtension 712
isASCII 875
isatty 712
isBitmap 875
isCollapsed 712
isColumnExpandable 712
isColumnRight 712
isConnectionSupported 713
isConnectorEnabled 713

IScrollBar 287
isCursored 713
isDBCS 875
isDigits 875, 876
isDropOnAble 713
isEntryPoint32Bit 713
ISetCanvas 293
  IToolBar 343
ISettingButton
  I3StateCheckBox 61
  ICheckBox 89
  IRadioButton 273
isExpanded 713
isInUse 713
ISize 299
isLayoutDistorted 713
isLeapYear 714
ISlider 301
isLike 714
isLowerCase 876
isMBCS 876
isMoveValid 714
isOpen 876
ISplitCanvas 307
isRowExpandable 714
isSBCS 876
isSelected 714
isSet 876
isSource 715
IStandardNotifier 313
  IAddress 555
  ICompany 557
  ICustomer 559
  IMMMasterAudio 521
  IVBFileDialog 373
  IVBFontDialog 381
  IVBNotebookPage 391
isTarget 715
IStaticText 315
  IBitmapControl 79
  IVBInfoArea 385
isText 876
IString 321
  I0String 59
IStringGenerator 325
isUpperCase 876
isValid 715
isWriteable 715
ISystemErrorInfo 327
italic 877
itemChangedEvent 964, 965
itemHandle 715
itemProvider 877
items 877
itemText 715
ITextControl
  ICircularSlider 95
  IEntryField 153
  IFlyText 161
  IGroupBox 181
  IMultiLineEdit 231
  IStaticText 315
  ITitle 337
ITextSpinButton 329
ITime 335
ITitle 337
IToolBar 343
IToolBarButton 349
ITrace 355
IVBag 415
IVBagOnBSTKeySortedSet 417
IVBagOnHashKeySet 419
IVBagOnSortedDilutedSequence 421
IVBagOnSortedLinkedSequence 423
IVBagOnSortedTabularSequence 425
IVBase
  IClipboard 103
  IColor 121
  IContainerColumn 131
  IContainerObject 135
  IFont 165
  IMessageBox 221
  IMMAudioCDContents 507
  IMMSpeed 537
  IMMTime 539
  IProfile 259
  IRecord 565
  IResourceLibrary 285
  ITrace 355
IVBBooleanPart 567
IVBCheckMenuHandler 357
IVBContainerControl 359
IVBDataTypePart
  IVBBooleanPart 567
  IVBDoublePart 569
  IVBLogicalAndPart 571
  IVBLogicalOrPart 573
  IVBLongPart 575
  IVBShortPart 577

IVBDataTypePart *(continued)*
IVBStringPart 579
IVBUnsignedLongPart 581
IVBUnsignedShortPart 583
IVBDoublePart 569
IVBDragDropHandler 369
IVBFactory 371
IVBFileDialog 373
IVBFlyText 377
IVBFontDialog 381
IVBInfoArea 385
IVBLogicalAndPart 571
IVBLogicalOrPart 573
IVBLongPart 575
IVBNotebookPage 391
IVBShortPart 577
IVBStringPart 579
IVBUnsignedLongPart 581
IVBUnsignedShortPart 583
IVBVariable 395
IVDeque 427
IVDequeOnDilutedSequence 429
IVDilutedSequence 431
IVEqualitySequence 433
IVEqualitySequenceOnDilutedSequence 435
IVEqualitySequenceOnTabularSequence 437
IVGBag
IVBag 415
IVGBagOnBSTKeySortedSet
IVBagOnBSTKeySortedSet 417
IVGBagOnHashKeySet
IVBagOnHashKeySet 419
IVGBagOnSortedDilutedSequence
IVBagOnSortedDilutedSequence 421
IVGBagOnSortedLinkedSequence
IVBagOnSortedLinkedSequence 423
IVGBagOnSortedTabularSequence
IVBagOnSortedTabularSequence 425
IVGDeque
IVDeque 427
IVGDequeOnDilutedSequence
IVDequeOnDilutedSequence 429
IVGDilutedSequence
IVDilutedSequence 431
IVGEqualitySequence
IVEqualitySequence 433
IVGEqualitySequenceOnDilutedSequence
IVEqualitySequenceOnDilutedSequence 435
IVGEqualitySequenceOnTabularSequence
IVEqualitySequenceOnTabularSequence 437
IVGHeap
IVHeap 439
IVGHeapOnDilutedSequence
IVHeapOnDilutedSequence 441
IVGLinkedSequence
IVLinkedSequence 443
IVGQueue
IVQueue 445
IVGQueueOnDilutedSequence
IVQueueOnDilutedSequence 447
IVGQueueOnTabularSequence
IVQueueOnTabularSequence 449
IVGSequence
IVSequence 403
IVGSet
IVSet 451
IVGSetOnBSTKeySortedSet
IVSetOnBSTKeySortedSet 453
IVGSetOnHashKeySet
IVSetOnHashKeySet 455
IVGSetOnSortedDilutedSequence
IVSetOnSortedDilutedSequence 457
IVGSetOnSortedLinkedSequence
IVSetOnSortedLinkedSequence 459
IVGSetOnSortedTabularSequence
IVSetOnSortedTabularSequence 461
IVGSortedBag
IVSortedBag 463
IVGSortedBagOnSortedDilutedSequence
IVSortedBagOnSortedDilutedSequence 465
IVGSortedBagOnSortedLinkedSequence
IVSortedBagOnSortedLinkedSequence 467
IVGSortedBagOnSortedTabularSequence
IVSortedBagOnSortedTabularSequence 469
IVGSortedSet
IVSortedSet 471
IVGSortedSetOnBSTKeySortedSet
IVSortedSetOnBSTKeySortedSet 473
IVGSortedSetOnSortedLinkedSequence
IVSortedSetOnSortedLinkedSequence 475
IVGSortedSetOnSortedTabularSequence
IVSortedSetOnSortedTabularSequence 477
IVGStack
IVStack 479
IVGStackOnTabularSequence
IVStackOnTabularSequence 481
IVGTabularSequence
IVTabularSequence 483
IVHeap 439

IVHeapOnDilutedSequence 441
IViewPort 399
IVLinkedSequence 443
IVQueue 445
IVQueueOnDilutedSequence 447
IVQueueOnTabularSequence 449
IVSequence 403
IVSet 451
IVSetOnBSTKeySortedSet 453
IVSetOnHashKeySet 455
IVSetOnSortedDilutedSequence 457
IVSetOnSortedLinkedSequence 459
IVSetOnSortedTabularSequence 461
IVSortedBag 463
IVSortedBagOnSortedDilutedSequence 465
IVSortedBagOnSortedLinkedSequence 467
IVSortedBagOnSortedTabularSequence 469
IVSortedSet 471
IVSortedSetOnBSTKeySortedSet 473
IVSortedSetOnSortedLinkedSequence 475
IVSortedSetOnSortedTabularSequence 477
IVStack 479
IVStackOnTabularSequence 481
IVTabularSequence 483
IWindow
  IFrameWindow 167
  IHelpWindow 189
  IMenu 205

J

julianDate 877
justifyData 715
justifyHeading 716

K

keyCount 877
keyModifier 878

L

labs 716
lastElement 878
lastIndexOf 716
lastIndexOfAnyBut 716
lastIndexOfAnyOf 716
lastPage 878
latched 879
latchedBackgroundColor 879
latchedBackgroundColorHalftone 879
latchedBitmap 879
latchedForegroundColor 879
latchingEnabled 880
layoutAdjustment 880
layoutType 880
ldexp 716
ldiv 716
left 880
leftCenter 880
leftIndex 880
leftJustify 717
length 880, 881
lengthOfWord 717
limit 881
lineCount 881
lineFrom 717
lineSpacing 882
listShowing 882
loadAccelTable 717
loadBitmap 717
loadDialog 717
loadHelpTable 717
loadIcon 717
loadMenu 718
loadMessage 718
loadOnThread 718
loadPointer 718
loadString 718
locateOrAdd 718, 719
locateText 719
location 882
log 719
log10 719
logicalAndValue 719
logicalNotValue 720
logicalOrValue 720
longHelpText 720
longStringTableOffset 882
longTextControl 882
lostFocusEvent 965
lowerBound 883
lowerCase 720, 883
lowerCaseEvent 965
lowLimit 883, 884
lseek 720

M

majorTabBackgroundColor 884
majorTabForegroundColor 884
malloc 720
margin 884, 885
marginSize 885
master 885
matchForMnemonic 720
mathSamples 585
maxAscender 885
maxCharHeight 885
maxDescender 885
maximize 720
maximized 885
maximizeRect 885
maximum 721
maximumSpeed 886
maximumWindows 886
maxNumberOfElements 886, 887
maxSize 887
maxUppercaseSize 887
maxX 887
maxXCenterY 888
maxXMaxY 888
maxXMinY 888
maxY 888
mblen 721
mbstowcs 721
mbtowc 721
mediaLoadedEvent 965
mediaPresent 888
menu 888
menuSelected 721
minimize 721
minimized 888
minimizeRect 889
minimum 721
minimumCharacters 889
minimumRows 889
minimumSize 889
minimumSpeed 889
minorTabBackgroundColor 890
minorTabForegroundColor 890
minScrollIncrement 890
minTextWidth 721
minutes 890
minX 890
minXCenterY 890
minXMaxY 891
minXMinY 891
minY 891
misfitFilteringEnabled 891
missingText 891
mixedTargetEmphasis 891
mleHandler 891
modal 892
mode 892
modeless 892
modf 722
modifiedEvent 965, 966
monitoringEnabled 892
monthName 722
monthOfYear 892
mousePointer 892
moveAfter 722
moveBefore 722
moveBy 722
movedBy 722
movedTo 722
moveIconTo 722
moveObjectTo 723
moveSizeToClient 723
moveToFirst 723
moveToLast 723
multiLineEdit 723
multipleSelect 893
multipleSelection 893
multiplyValue 723, 724

N

name 893
nameView 893
nativeRect 893
nextPage 724
nextShellRect 894
nlsCompare 724
noAdjustPosition 894
noDismiss 894
nonPropOnly 894
normalSpeed 894
normalTargetEmphasis 894
notebookSize 724
notifyObservers 724
notifyOwner 725
notInput1 895
notInput10 895

notInput2 895
notInput3 895, 896
notInput4 896
notInput5 896
notInput6 896
notInput7 897
notInput8 897
notInput9 897
notValue 898
notValueAsText 898
now 898
number 898
numberOfApplications 898
numberOfButton2Items 899
numberOfColumnChanges 899
numberOfDifferentElements 899
numberOfElements 899, 900
numberOfEntries 900
numberOfKeys 900
numberOfLines 900
numberOfObjectChanges 900
numberOfOccurrences 725
numberOfSelections 901
numberOfTicks 725
numberOfTracks 901
numWords 901

O

objectAt 725
objectCopy 901
objectCount 901
objectList 902
objectText 902
objectUnderPoint 725
objectWindow 725
occurrencesOf 726
open 726, 902
openDoor 726
openOnThread 726
operator 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742
operator= 742
ordered 902
orderedTargetEmphasis 902
ordinal 902
orientation 903
origDefaultButtonHandle 903
orValue 742, 743
outline 903
outlineType 903
overlayWith 743
owner 903, 904

P

packType 904
pad 904
pageBackgroundColor 904
pageScrollIncrement 904
pageSettings 743
pageSize 904
pagesToEnd 743
pagesToMajorTab 743
pagesToMinorTab 743
parent 905
parentObject 744
parentSize 905
Part List 37
passDeviceEvent 966
paste 744
pause 744
pauseButton 905
perror 744
phone 905
pitch 905
play 744
playableDevice 905
playAt 745
playButton 905
playFast 745
playingForward 906
playScan 745
playSlow 745
pointerCaptured 906
pointSize 906
pop 745
Portability 7
portability publications 981
position 906
positionBehindSibling 745
positionBehindSiblings 746
positionChangeEvent 966
positionOnSiblings 746
positionTimerEvent 966
postEvent 746
pow 746
preappendText 746

pressedOK 906, 907
pressedOkEvent 966, 967
presSpace 907
previousPage 746
primaryFormat 907
primaryScale 907
printable 907
procAddress 746
profile 907
publications, portability 981
publications, related 981
punctuation 907
push 746
putc 747
putchar 747
putenv 747
puts 747
putwc 747

R

radius 907
rand 747
range 908
read 747
readOnly 908
realloc 747
reallocateString 748
record 748
rect 908
redMix 908
refresh 748
refreshAllContainers 748
refreshOn 908
refreshTabs 748
registerFormat 748
related publications, VisualAge for C++ 981
relativeWindowRect 909
release 749
releasePointer 749
releasePresSpace 749
remove 749, 750
removeAll 751, 752
removeAllItems 752
removeAllKeys 752
removeAllOccurrences 752, 753
removeAllPages 753
removeAtPosition 753
removeBorder 753, 754
removeButton2Items 754
removeColumn 754
removeDetent 754
removedEvent 967, 968
removeEvent 968
removeExtension 754
removeFirst 754, 755
removeFromCell 755
removeFromWindowList 755
removeHelpText 755
removeInUse 755
removeKeyLike 755
removeLast 756
removeLine 756
removePage 756
removeSelectedItems 756
removeTabSection 757
removeWords 757
rename 757
replacedEvent 968, 969
requiresFiles 909
reset 757
resetActiveColor 757
resetActiveTextBackgroundColor 757
resetActiveTextForegroundColor 757
resetBackgroundColor 757
resetBorderColor 757
resetChangedFlag 758
resetDisabledBackgroundColor 758
resetDisabledForegroundColor 758
resetFillColor 758
resetFont 758
resetForegroundColor 758
resetHiliteBackgroundColor 758
resetHiliteForegroundColor 758
resetInactiveColor 758
resetInactiveTextBackgroundColor 759
resetInactiveTextForegroundColor 759
resetLatchedBackgroundColor 759
resetLatchedForegroundColor 759
resetMajorTabBackgroundColor 759
resetMajorTabForegroundColor 759
resetMinimumSize 759
resetMinorTabBackgroundColor 759
resetMinorTabForegroundColor 760
resetPageBackgroundColor 760
resetShadowColor 760
resetSplitBarEdgeColor 760
resetSplitBarMiddleColor 760

resetTransparentColor 760
resize 760
resourceLibrary 909
restore 760
restoreRect 909
result 910
resume 761
returnValue 910
reverse 761
reversed 910
reverseText 761
rewind 761
rewindButton 910
ribbonStripEnabled 910
right 910
rightCenter 910
rightJustify 761
rotationIncrement 911
rowHeight 761
rpmatch 761

S

samplesPerSecond 911
save 762
saveAs 762
saveHeadphonesSetting 762
saveSpeakersSetting 762
saveVolume 762
scaleBy 762
scaledBy 762, 763
scaleEvent 969
scroll 763
scrollableRange 911
scrollBoxPosition 911
scrollBoxRange 911
scrollDetailsHorizontally 763
scrollHorizontally 763
scrollToObject 763
scrollVertically 763
scrollViewHorizontallyTo 763
scrollViewVerticallyTo 763
searchListWindow 911
seconds 911, 912
seek 764
seekToEnd 764
seekToStart 764
select 764
selectable 912
selectAll 764
selected 912
selectedCnrElement 912
selectedCnrElements 764
selectedCnrItem 912
selectedCnrItems 912
selectedCollectionPosition 913
selectedCollectionPositions 913
selectedElement 914
selectedElements 764
selectedFileCount 914
selectedIndex 914
selectedItem 914
selectedItems 915
selectedRange 915
selectedText 915
selectedTextLength 915
selectEvent 969
selectHalftone 765
selection 916
selectionText 916
selectRange 765
sendEvent 765
separator 916
servant 916
set 765
setActiveWindow 765
setAddressToDefault 765
setAllEmphasis 766
setAssociatedWindow 766
setBitmaps 766
setBorderSize 766
setbuf 766
setChangedFlag 766
setCharHeight 766
setCharSize 767
setCharWidth 767
setCityToDefault 767
setClosed 767
setColumnWidth 767
setCursor 767
setCustomerListToDefault 767
setData 767
setDataOffset 768
setDecrementBitmaps 768
setDeleteColumnsOnClose 768
setDeleteObjectsOnClose 768
setDestroyOnClose 768
setDialogTemplate 768

setDirection 769
setDisplayPS 769
setEditColumn 769
setEditMLE 769
setEditObject 769
setEditRegion 769
setExtendedSelection 769
setExtensionSize 769
setFamily 770
setFocus 770
setFontAngle 770
setFontFromFontModally 770
setFontFromWindowModally 770
setFontShear 770
setHandle 770
setHelpKey 771
setHelpTable 771
setHelpText 771
setHomePhoneToDefault 771
setIncrementBitmaps 771
setInitialDrive 771
setInitialFileType 771
setInput1 772
setInput10 772
setInput2 772
setInput3 772
setInput4 772
setInput5 773
setInput6 773
setInput7 773
setInput8 773
setInput9 773
setInUse 774
setItemHandle 774
setItemHeight 774
setItemText 774
setKey 774
setLayoutDistorted 774
setMajorTabSize 775
setMaster 775
setMinorTabSize 775
setMixedTargetEmphasis 775
setMLEHandler 775
setMultipleSelection 775
setNameToDefault 775
setNormalTargetEmphasis 775
setNotInput1 776
setNotInput10 776
setNotInput2 776
setNotInput3 776
setNotInput4 776
setNotInput5 777
setNotInput6 777
setNotInput7 777
setNotInput8 777
setNotInput9 777
setOKButtonText 778
setOpenDialog 778
setOrderedTargetEmphasis 778
setPageButtonSize 778
setPhoneToDefault 778
setPosition 778
setPreviewText 778
setPrinterPS 778
setProgram 779
setRefreshOff 779
setRowHeight 779
setSaveAsDialog 779
setScrollBar 779
setSelected 779
setSeparator 779
setShaftBreadth 780
setSingleSelection 780
setSizeList 780
setSplitBarThickness 780
setSplitWindowPercentage 780
setStateToDefault 780
setStatusText 780
setStreetToDefault 780
setTab 780
setTabBitmap 781
setTabText 781
setTickLength 781
setTicks 781
setTickText 781
setTitle 781, 782
setTitleAlignment 782
setTitleText 782
setToDefault 782
setTrackTitle 782
setTreeExpandIconSize 782
setTreeItemIcons 782
setTreeViewIndent 782
setUserData 783
setUsingHelp 783
setvbuf 783
setWindow 783
setWindowFont 783

setWorkPhoneToDefault 783
setZipToDefault 783
shadowColor 917
shaftPosition 917
shaftSize 917
show 784
showContentsHelp 784
showCustomized 784
showDetailsView 784
showErrorInfo 785
showException 785
showFlowedNameView 785
showFlowedTextView 785
showFromFont 785
showGeneralHelp 785
showHelpPanel 785
showIconView 785
showIndexHelp 785
showing 917
showingMiniIcons 917
showKeysHelp 786
showList 786
showMiniIcons 786
showModally 786
showNameView 786
showObject 786
showPanelIds 787
showSeparators 787
showSourceEmphasis 787
showSplitBar 787
showTextView 787
showTreeIconView 787
showTreeLine 787
showTreeNameView 787
showTreeTextView 788
showUsingHelp 788
shrinkBy 788
shrunkBy 788
sin 788
singleSelection 917
sinh 788
size 917, 918
sizeBy 788
sizedBy 788
sizedTo 789
sizeToGraphic 918
snapToTickEnabled 918
sort 789
sortByIconText 789
sourceRectangle 918
space 789
special notices xxv
speed 918, 919
speedFormat 919
spinDown 789
spinFieldValid 919
spinTo 789
spinUp 790
splitBarEdgeColor 919
splitBarMiddleColor 919
splitBarOffset 919
splitBarThickness 790
splitWindowPercentage 790
sqrt 790
squareValue 790
srand 791
standardBitmapSize 920
standardFormat 920
standardTextLines 920
standardTextWidth 920
start 791
startAnimation 791
startPositionTracking 791
startScanningBackward 791
startScanningForward 791
state 920
staticWindowSamples 587
statusTextAlignment 920
stdioSamples 589
stdlibSamples 591
stepBackwardButton 921
stepForwardButton 921
stepFrame 792
stop 792
stopAnimation 792
stopButton 921
stopPositionTracking 792
street 921
strikeout 921
string 921
stringGenerator 922
stringTableOffset 922
strip 792
stripBlanks 792
stripBlanksOnText 793
stripLeading 793
stripLeadingBlanks 793
stripTrailing 793

stripTrailingBlanks 793
strtod 793
strtol 793
strtold 793
submenu 922
submenuHandle 922
subString 793
subtractValue 793, 794
supportsAudio 922
supportsCommand 794
supportsDigitalTransfer 922
supportsDisableEject 923
supportsEject 923
supportsOverlayGraphics 923
supportsPlay 923
supportsRecord 923
supportsRecordInsertion 923
supportsReverse 923
supportsSave 923
supportsSizing 924
supportsStreaming 924
supportsStretchToFit 924
supportsVideo 924
supportsVolumeAdjustment 924
swab 794
system 794
systemCommand 924, 925
systemCommandEvent 969
systemProfile 925
systemScrollBarWidth 925
systemScrollBoxLength 925
systemScrollButtonLength 925

T

tabShape 925
tabStop 926
tabTextAlignment 926
tan 794
tanh 794
tempnam 795
text 926, 927
textAsLowerCase 927
textAsUpperCase 928
textEqualDefault 928
textEqualDefaultEvent 969
textLength 928
textLengthEvent 969
textLines 795
textNotEqualDefault 928
textNotEqualDefaultEvent 969
textRectangle 795
textView 928
textVisible 928
textWidth 795
thousandths 929
throwCLibError 795
throwError 795
throwGUIError 795
throwMMError 796
throwSystemError 796
tickLength 796
tickPosition 796
tickSpacing 796, 929
tickText 796
time 929
timeFormat 929
title 930
titleRectangle 930
titleSeparatorVisible 930
titleVisible 930
titleWriteable 930
tmpfile 796
tmpnam 796
today 930
toolBarContainer 930
toolBarList 931
top 931
topCenter 931
topLeft 932
topPage 932
topRight 932
totalPages 932
traceDestination 932
traceEnabled 932
track 932
trackBackward 797
trackForward 797
trackStartedEvent 969
trackTitle 797
trademarks xxv
translate 797
transparentColor 933
transpose 797
treble 933
treeIconView 933
treeNameView 933
treeTextView 933

treeView 933
tryToLoadBitmap 797
tryToLoadIcon 797
tryToLoadMessage 797
tryToLoadString 798
turnToPage 798
type 933

U

umask 798
underscore 934
undo 798
undoable 934
ungetc 798
ungetwc 798
unhighlight 798
unionWith 799
unlatch 799
unlink 799
upc 934
update 799
upperBound 934
upperCase 800, 934
upperCaseEvent 970
useBitmapOnly 800
useDefaultWindow 800
useExtensionMinimumSize 800
useNonPropOnly 800
userData 935
userProfile 935
usesDialogBackground 935
useVectorOnly 800

V

valid 935
validDBCS 935
validMBCS 935
value 936, 937
valueAboveHighLimit 937, 938
valueAboveHighLimitEvent 970
valueAs1Based 938, 939
valueAsDouble 939, 940
valueAsInt 940
valueAsText 940, 941, 942
valueAsUnsigned 942
valueBelowLowLimit 942, 943
valueBelowLowLimitEvent 970
valueEqualDefault 943, 944
valueEqualDefaultEvent 970
valueEqualHighLimit 944
valueEqualHighLimitEvent 970
valueEqualLowLimit 944, 945
valueEqualLowLimitEvent 970
valueFalseEvent 970
valueNegative 945
valueNegativeEvent 970
valueNotEqualDefault 946
valueNotEqualDefaultEvent 970
valueNotZero 946, 947
valueNotZeroEvent 970
valueOutsideLimits 947, 948
valueOutsideLimitsEvent 971
valuePositive 948
valuePositiveEvent 971
valueTrueEvent 971
valueWithinLimits 948, 949
valueWithinLimitsEvent 971
valueZero 949, 950
valueZeroEvent 971
vbbase.vbb 47
vbcc.vbb 407
vbmm.vbb 485
vbsample.vbb 549
vectorOnly 950
vertical 950
verticalDataAlignment 950
verticalHeadingAlignment 950
verticalScrollBar 950
verticalSeparator 950
videoFileHeight 951
videoFileName 951
videoFileWidth 951
videoHeight 951
videoWidth 951
view 951
viewedPagesWindow 951
viewNumber 952
viewPortOnWindow 952
viewPortOnWorkspace 952
viewText 952
viewWindow 952
viewWindowDrawRectangle 952
viewWindowSize 953
virtualKey 953
visibilityDisabledEvent 971
visibilityEnabledEvent 971

visible 953, 954
visibleCount 954
visibleLines 954
visibleRectangle 954
VisualAge for C++ publications 981
volume 954, 955

W

wcsid 800
wcstombs 801
wctob 801
wctomb 801
whiteSpace 955
width 955
willDeleteColumnsOnClose 955
willDeleteObjectsOnClose 955
willDestroyOnClose 955
window 801
windowInCell 801
word 801
wordIndexOfPhrase 801
words 801
wordWrap 956
workPhone 956
write 802
writeable 956, 957
writeLineNumberEnabled 957
writePrefixEnabled 957
writeToQueue 802
writeToStandardError 802
writeToStandardOutput 802

X

x 957
x2b 802
x2c 802
x2d 802

Y

y 957
year 957

Z

zip 958

Communicating Your Comments to IBM

IBM VisualAge for C++ for Windows
Visual Builder Parts Reference
Version 3.5

Publication No. S33H-5035-00

If there is something you like—or dislike—about this book, please let us know. You can use one of the methods listed below to send your comments to IBM. If you want a reply, include your name, address, and telephone number. If you are communicating electronically, include the book title, publication number, page number, or topic you are commenting on.

The comments you send should only pertain to the information in this book and its presentation. To request additional publications or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give it to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - United States and Canada: 416-448-6161
  - Other countries: (+1)-416-448-6161
- If you prefer to send comments electronically, use the network ID listed below. Be sure to include your entire network address if you wish a reply.
  - Internet: torrcf@vnet.ibm.com
  - IBMLink: toribm(torrcf)
  - IBM/PROFS: torolab4(torrcf)
  - IBMMAIL: ibmmail(caibmwt9)

Readers' Comments — We'd Like to Hear from You

IBM VisualAge for C++ for Windows
Visual Builder Parts Reference
Version 3.5

Publication No. S33H-5035-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
S33H-5035-00

IBM®

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

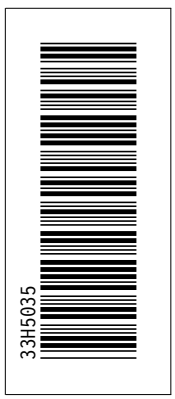
IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 EGLINTON AVENUE EAST
NORTH YORK ONTARIO CANADA M3C 1H7

Fold and Tape | **Please do not staple** | Fold and Tape

S33H-5035-00

Cut or Fold
Along Line

IBM®

Part Number: 33H5035

Printed in U.S.A.